On Physics-Inspired Generative Models

By

Yilun Xu

S.B., Peking University (2020)

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 2024

Authored by:	Yilun Xu Department of Electrical Engineering and Computer Science May 17, 2024
Certified by:	Tommi S. Jaakkola Professor of Electrical Engineering and Computer Science Thesis Supervisor
Accepted by:	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee on Graduate Students

On Physics-Inspired Generative Models

By

Yilun Xu

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Physics-inspired generative models such as diffusion models constitute a powerful family of generative models. The advantages of models in this family come from relatively stable training process and high capacity. A number of possible improvements remain possible. In the thesis, we will first delve into the improved techniques for training and sampling in diffusion models. The training objectives of diffusion models exhibit high variance when the data distribution is multi-modal. To mitigate this, we propose a training objective that generalizes conventional denoising score-matching and significantly reduces variance in training targets. Alternatively, we introduce a training framework that integrates learnable discrete latents into continuous diffusion models. These latents simplify the learning of diffusion models' complex noise-to-data mapping. On the other hand, the sampling process of diffusion models generally involves solving differential equations. To expedite the sampling process, we propose a new sampling algorithm that combines the best of previous ODE and SDE samplers, greatly boosting the performance of pre-trained diffusion models. Additionally, our research explores methods to promote diversity in finite samples by introducing mutual repulsion forces in the generative process.

In the realm of physics-inspired generative models, many physical processes could be used to develop generative models. We will introduce a new family of generative models arising from electrostatic theory, termed Poisson Flow Generative Models (PFGM). PFGM rivals leading diffusion models while showcasing improved sampling robustness. The extended version, PFGM++, places diffusion models and PFGM under the same framework and introduces new, better models. We will further present a principled approach to convert physical processes into generative models.

Thesis supervisor: Tommi S. Jaakkola
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I am incredibly grateful to my advisor, Tommi Jaakkola, for his constant guidance and support throughout my four-year PhD journey. His advice was invaluable in every aspect of my research: from identifying research directions to providing constructive criticism and encouragement for my sometimes whimsical ideas, and from presentation skills to paper writing, teaching, and mentoring junior students. More importantly, his guidance extends beyond research, helping me to grow as a person. I have learned so much from him in many areas, of which I can only mention a few due to space: Tommi is an optimistic advisor with his own distinct tastes and a readiness to listen. Every time I discussed any idea with him, good or bad, he always listened patiently and analyzed the strengths and weaknesses of the ideas with professional and sometimes creative feedback. Tommi is also an advisor of great integrity, with high academic ethics and a fair and just approach to his students, always prioritizing the career success of his students. Over these four years, I have gradually learned these virtues from him, although I know I still have much to learn to catch up with him.

My peer colleagues have greatly enriched my life at MIT on campus. I was fortunate to have Ziming Liu as my collaborator at MIT. We first met during a discussion about optimal transport at the end of 2021. Although that discussion didn't yield results, we reconnected over an idea on "physics-inspired generative models" a month later and subsequently collaborated on a series of projects. Much of the work in this thesis is related to Ziming. Throughout our collaboration, I often found myself learning from Ziming: his creativity, witty comments, and knack for simplifying complex concepts. Shangyuan Tong was a reliable collaborator; I generally sought his opinion first whenever I had a new idea, as he always provided excellent advice. Yonglong Tian was like an elder brother, always willing to share his wisdom and life experiences, and to support our projects with whatever resources he could offer. Xiang Chen

is a collaborator who can always patiently help me with theoretical questions at any time and a brother who listens well when I face personal challenges. I am grateful for the assistance and insights provided by Gabriele Corso, particularly for his profound understanding of AI for Science. I enjoyed the casual chats with Hao He and am grateful for his help on my first paper at MIT, which greatly helped me get up to speed in my PhD journey. Di Luo and Tianxiao Shen were excellent research collaborators, distinguished by their diligence, ingenuity, and creativity. Amit Schechter's excellent communication skills and empathy made my first experience as a TA at MIT extremely positive, contributing to the course receiving high ratings. I have learned a lot from Timur Garipov about coding, paper writing, and filing my tax return. I will never forget the quality time with many other (former) group members: Guang-He Lee, Wengong Jin, Vikas Garg, Tian Xie, Bracha Laufer, Benson Chen, Bowen Jing, Xiang Fu, Abhi Gupta, Felix Faltings, Peter Holderrieth, Chenyu Wang, Jason Yim, Hannes Stärk and Octavian Ganea. The sudden passing of Octavian Ganea deeply saddens me — it is an enormous loss for his family and the academy. I also thank junior students Mingyang Deng and Ziyu Xiong for placing their trust in me during their internship in Tommi's group — it has been a joy to watch them grow as researchers!

I am also very fortunate to have worked with Karsten Kreis and Arash Vahdat during my first—and only—industrial internship in my PhD. They offered unrestrained research suggestions and insights, helped me improve my presentations and writings, provided advice on career paths in the industry, and supported me with their extensive industrial backgrounds.

I would like to thank Phillip Isola and Karsten Kreis for agreeing to serve on my PhD thesis committee and providing valuable feedback. I am also much obliged to Dina Katabi and Vincent Sitzmann for their service to my RQE committee. I want to thank Max Tegmark for his support in generously providing a recommendation letter for me and helping with the paper writing. The graduate officers, Leslie Kolodziejski, Janet Fisher, and Meredith Bittrich, also provide invaluable support throughout my PhD journey. I am also thankful to the lab assistant Teresa Cataldo, for her timely and thoughtful response to many requests. During my job search process, I receive generous help from Yang Song, Ji Lin, Ruiqi Gao, Jim Fan, Yilun Du, Durk Kingma, Shengjia Zhao, Jiaming Song, Karsten Kreis and Arash Vahdat. It is my honor to receive support from the MIT-DSTA Singapore collaboration, the National

Science Foundation, and the MIT-IBM Grand Challenge project for my thesis research.

I also receive enormous support and company from my close friends during my time in Boston. I would like to thank my long-time friends, Tianyuan Zhang and Victor Chen. I always know they will be there to listen, whether in times of up or down. Friends like Veronica Liao, Nicole Han, Tianwei Yin, Vivian Chiang, Matt Hong, Tianchen Yu, Ke Li, Yuanqi Du, and Weiyi Zhang have provided great conversations. I am grateful to be part of the MIT Table Tennis team. I will always memorize the games with Jie Xu, Tonghang Han, Jingting Lin, Anastasia Nikolakopoulou, Fabian Mohr, IIya Smirnov and Maria Castillo. Thank you, Maria, for trusting me as your doubles partner. We are undoubtedly the strongest mixed doubles team in the Boston area!

Finally, I would like to thank my parents for their unwavering love and support. Words cannot express my gratitude!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Generative models have significantly transformed the way people work, create, and learn in recent years. Prominent applications include ChatGPT [1], text-to-image models [2]–[4], text-to-3D models [5], [6], and text-to-video models [7], [8]. Such capabilities can greatly stimulate creativity and enhance work efficiency across numerous sectors, including education, the gaming industry, social media, and professional editing software. The training of generative models in machine learning is based on the assumption that training data are sampled from an unknown data distribution [9]. Modern generative models commonly use deep neural networks to approximate complex data distributions based on finite training data. They can generate novel data points by sampling from these modeled distributions.

Among the various data types utilized in generative modeling, high-dimensional data poses a significant challenge primarily due to the curse of dimensionality. As the dimensionality increases, the volume of data space expands exponentially. This phenomenon makes it difficult to capture and model the data distribution effectively with limited training data in high-dimensional space. Additionally, the data distributions of interest are often highly complex and multi-modal, further complicating the task of generative modeling. Recently, diffusion models [10]–[12], along with broader physics-inspired generative models [13], have emerged as strong frameworks and achieved impressive results on a wide spectrum of generative tasks for high-dimensional data. Before diffusion models, the primary approaches included: (i) Generative Adversarial Networks (GANs [14]) that utilize an adversarial training objective; (ii) models trained with maximum likelihood objectives, such as PixelCNN [15] and normal-

izing flow models [16], [17]; (iii) Variational Autoencoders [18], [19] and (iv) Energy-based models [20], [21]. Each method, however, involves its own set of drawbacks: (i) can result in unstable training and low diversity in generated samples; (ii) necessitates specific architectural designs that may limit model capacity; (iii) demands careful coordination between various neural networks; and (iv) suffers from slow training and sampling speeds. Utilizing **natural physical processes** as encoders to transform data into noise, diffusion models learn to perform generative tasks by **reversing the physical processes**. This approach allows them to bypass many of the limitations associated with earlier generative models.

1.1 Generative Modeling by Reversing Physical Processes

Drawing on principles from thermodynamics [10], diffusion models involve two opposing processes: a forward process that transforms the data distribution into a simpler prior smoothly over time, and a reverse process that iteratively denoises a sample from this noisy prior. The forward process in diffusion models is a simple Brownian motion that degrades the data by gradually adding Gaussian noise. To reverse this process, it is sufficient to learn a time-dependent vector field, known as the *score function*, and iteratively solve a differential equation [22]. Unlike GANs and VAEs, the training of diffusion models does not require synchronization between multiple neural networks, resulting in a more stable training procedure. Additionally, they are less constrained by architectural requirements and employ an iterative process similar to the concatenation of neural networks, enhancing their overall capacity. This stability and enhanced capacity allow diffusion models to scale effectively to large datasets.

Despite their advantages, diffusion models encounter several challenges, including a high-variance training process, especially when handling multi-modal data, and a slow iterative sampling process. Additionally, the independent and identically distributed (i.i.d.) sampling procedure often results in repetitive samples. These issues underscore the need for **approaches to stabilize and improve the training of diffusion models on complex datasets** and demand for **new techniques aimed at accelerating the sampling process and promoting diversity within mini-batch samples**. Furthermore, diffusion models are

just one of many physics-inspired generative models. Numerous physical processes beyond Brownian motion remain untapped and can be utilized to construct generative models. This leads us to an important question: **Can we discover other physics-inspired generative models that exhibit even better properties?** In the following sections, we will briefly summarize the improved training and sampling techniques for diffusion models and discuss our research on developing other physics-inspired generative models, as detailed in later chapters.

1.1.1 Improved Training Techniques for Diffusion Models

The training of diffusion models utilizes a perturbation-denoising approach to estimate the vector fields. This starts by perturbing clean data using Gaussian noise; then, the network reconstructs the original from the perturbed samples [12]. However, with complex multi-modal data, many clean data points can be perturbed to similar noisy samples, creating ambiguous training targets and causing instability.

In [23], we tackle this problem by implementing a weighted summation of multiple clean data points to estimate the true target, pinpointing the direction of the true vector field from the perturbed sample. This novel training objective generalizes conventional single-point estimate, and **significantly reduces variance in training targets**. Consequently, this results in **improved sample quality, enhanced stability, and accelerated training speed in various variants of diffusion models**.

Another challenge facing diffusion models is the requirement to learn a non-linear and highly complex mapping from a uni-modal Gaussian distribution to a multi-modal data distribution. This complexity makes training more difficult and results in generative ordinary differential equation (ODE) [24] trajectories with strong curvature. To address this issue, we augment diffusion models with discrete latent variables. These discrete latents help to capture different modes within the data distribution, and the task for diffusion models then becomes capturing continuous variations within each mode based on the given discrete latent. The separated modeling of discrete and continuous variation significantly **simplifies the learning of the model's complex noise-to-data mapping**. This approach effectively reduces the curvature of the diffusion model's generative ODE and makes the overall training

loss smaller, especially at larger diffusion times.

1.1.2 Improved Sampling Techniques for Diffusion Models

Solving differential equations in the sampling process of diffusion models often involves speed and quality trade-offs. Deterministic samplers (ODE-based) [25]–[27] are fast but plateau in performance, whereas stochastic samplers (SDE-based) [27], [28] provide better sample quality but are slower. Our analysis attributes this difference to sampling errors: ODE-samplers have smaller discretization errors, while stochasticity in SDE contracts errors accumulated during the sampling process [29].

Based on these insights, in [29], we propose a novel sampling algorithm called *Restart*, which **combines the advantages of ODE and SDE**. The method alternates between adding substantial noise during additional forward steps and strictly adhering to a reverse ODE process. The inclusion of substantial forward noise enhances the contraction effect of the stochasticity, while adherence to the reverse ODE allows for expedited sampling. This separation of stochasticity and the deterministic sampling process proves highly beneficial, as Restart **surpasses SDE and ODE samplers in speed and quality on standard benchmarks** (CIFAR-10 and ImageNet-64), and **exhibits a superior balance of text-image alignment, visual quality, and diversity on large-scale text-to-image Stable Diffusion model.**

Conventionally, diffusion models generate independent and identically distributed samples from the model distribution. In practice, however, the model often needs to be sampled multiple times to achieve a diverse set of mini-batch samples, which incurs costs orthogonal to sampling time. We propose moving beyond the typical assumption of independent samples to enhance sample diversity and efficiency. Our approach introduces an extension of diffusion-based generative sampling called *particle guidance*. In this method, **a joint-particle time-evolving potential enforces diversity by adding mutual repulsive forces between samples (particles)**. Empirically, our framework improves diversity and mitigates memorization in applications such as text-to-image generation and molecular conformer generation.

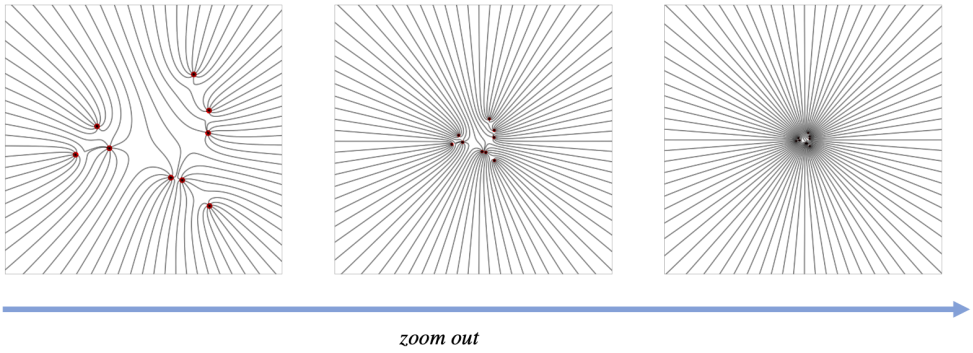


Figure 1.1: Gradual zooming-out views of the 2D electric field in a 3D augmented space. As the distance from the data support increases, the charge distribution progressively resembles a point charge. This indicates that the charge distribution effectively “collapses" to a single point when we are sufficiently far away.

1.1.3 Generative Models from Alternative Physical Processes

With diffusion models as a notable example, physics-inspired generative models encompass a forward process that simplifies complex data distributions into prior distributions over time, and an associated reverse process, or sampling process, that gradually reverts these prior distributions back to their original data distribution. Consequently, **in order to define new physics-inspired generative models, it is necessary to identify an appropriate forward process**. This process should naturally simplify the data distribution over time and be reversible, and its associated vector field should be easy to learn by neural networks.

Drawing on electrostatics, we chart a novel path for physics-inspired generative models and introduce the Poisson Flow Generative Models (PFGM) [30] and its extended version, PFGM++ [31]. PFGM interprets data as electrical charges in an augmented space. As shown in Figure 1.1, when we move sufficiently far from the data support, the charge distribution collapses into a point charge, and the electric field appears radial in every direction. Consequently, it can be demonstrated that the electric field lines emitted by these charges define a bijection between the data distribution and a uniform distribution on a large hemisphere. Empirically, this new family of models surpasses diffusion models in terms of sample quality, sampling speed, and robustness. Additionally, we explore the duality between physical processes and generative models with an aim to conceptualize and design additional new physics-inspired generative models [13].

1.2 Summary of Thesis

This thesis is structured into three thematic parts. Below, we provide brief summary of each part.

Part I focuses on developing new techniques aimed at stabilizing the training in diffusion models, and straightening their generative trajectories, particularly when dealing with complex multi-modal datasets.

- In Chapter 3, we remedy the high-variance problem in diffusion models' objective, by incorporating a reference batch, which we use to calculate weighted conditional scores as more stable training targets. We show that the procedure indeed helps in the challenging intermediate regime by reducing (the trace of) the covariance of training targets. This chapter is based on [23].

- In Chapter 4, we augment diffusion models with learnable discrete latents, inferred with an encoder, and train DM and encoder end-to-end. The discrete latents significantly simplify learning the DM's complex noise-to-data mapping by reducing the curvature of the DM's generative ODE, and improving sample quality on various datasets with ODE samplers. This chapter is based on [32].

Part II is about accelerating the sampling process of diffusion models, as well as promoting diversity by exerting mutually repulsive forces among samples. All the techniques discussed are training-free and can be readily applied to any pre-trained diffusion models.

- In Chapter 5, we propose a novel sampling algorithm called *Restart* to combine the best of previous ODE and SDE samplers. Restart alternates between adding substantial noise in additional forward steps and strictly following a backward ODE. Empirically, the Restart sampler surpasses previous SDE and ODE samplers in both speed and accuracy. The chapter is based on [29].

- In Chapter 6, we propose particle guidance, an extension of diffusion-based generative sampling where a joint-particle time-evolving potential enforces diversity. We test the framework in both conditional image generation, where we can increase diversity

without affecting quality, and molecular conformer generation, where we improve the median error over previous methods. This chapter is built upon [33].

Part III investigates a new family of generative models inspired by electrostatics theory and its unification with diffusion models in an extended view. This part also provides a forward-looking perspective on the methodologies for constructing generative models given any physical process.

- In Chapter 7, we introduce a new type of generative model — Poisson Flow Generative Model (PFGM) — based on electrostatic theory. We interpret the data points as electrical charges on the $z = 0$ hyperplane in a space augmented with an additional dimension z, generating a high-dimensional electric field (the gradient of the solution to Poisson equation). We prove that if these charges flow upward along electric field lines, their initial distribution in the $z = 0$ plane transforms into a distribution on the hemisphere of radius r that becomes *uniform* in the $r \to \infty$ limit. We demonstrate that PFGM outperforms previous state-of-the-art diffusion models while offering faster image generation speeds.

- In Chapter 8, we expand the theory of electrostatics used in PFGM, unifying diffusion models and PFGM. More intriguingly, interpolation between the two models reveals a sweet spot with new state-of-the-art performance in image generation. We provide a theoretical explanation for why both PFGM and diffusion models are sub-optimal solutions for practitioners. This chapter is built on [31].

- In Chapter 9, we present a unifying framework and algorithm that transforms physical processes into smooth density flow generative models. Additionally, we introduce a classification criterion based on the dispersion relations of the underlying physical partial differential equations (PDEs). This theoretical approach can be applied to various physical PDEs, leading to the discovery of new families of generative models. This chapter is based on [13].

We conclude the thesis and discuss current limitations in Chapter 10.

Chapter 2

Background on Diffusion Models

In this chapter, we formulate the problem of generative modeling and explain how diffusion models tackle this task. Since other physics-inspired generative models share similar training and sampling techniques with diffusion models, we will use diffusion models as an anchor to examine the training and sampling processes of physics-inspired generative models.

2.1 Problem Formulation

A generative model is a statistical model designed to approximate an unknown data distribution $p(\mathbf{x})$. It parameterizes a *model distribution* p_θ using a learnable function, such as a deep neural network, characterized by parameters θ. In a typical problem setting, we have a finite number of N-dimensional data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, each sampled *independently and identically* from an unknown data distribution $p(\mathbf{x})$, as the training set. The primary objective is to closely approximate the true data distribution by estimating θ from the available n data points:

$$p_\theta(\mathbf{x}) \approx p(\mathbf{x})$$

Upon determining the model parameters θ, it is possible to *generate novel data* by sampling from the model distribution $p_\theta(\mathbf{x})$. Therefore, a valid generative model must fulfill two key requirements: **(1) the ability to approximate the data distribution accurately and**

(2) the capability to allow sampling from the model distribution.

2.2 Constructing Generative Models with Thermodynamical Theory

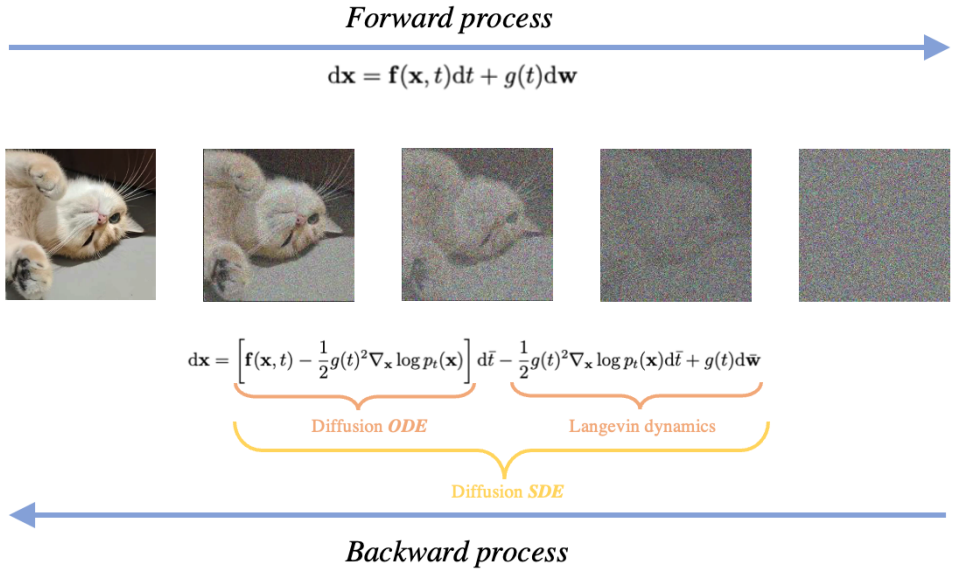


Figure 2.1: The forward SDE and backward SDE/ODE in diffusion models.

Inspired by the Brownian motion in thermodynamics [10], diffusion models gradually degrade the complex data distribution into an equilibrium state through a forward process[1]. This thesis adopts the continuous perspective on such degradation, as discussed in [34]. An intuitive analogy can be drawn from a drop of ink diffusing in water. The forward process is an SDE with no learned parameter in the form of:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ with $\mathbf{x}(0) \sim p_0$ being the data distribution, $t \in [0, 1]$, $\mathbf{f}\colon \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$, $g\colon [0, 1] \to \mathbb{R}$, and $\mathbf{w} \in \mathbb{R}^d$ is the standard Wiener process. It gradually transforms the data distribution to a known prior (*e.g.,* Gaussian distribution) as time goes from 0 to 1.

[1]For simplicity, we focus on the version where the diffusion coefficient $g(t)$ is independent of $\mathbf{x}(t)$.

We denote the time-dependent intermediate marginal distribution in the forward process as p_t. The sampling of diffusion models is done via a corresponding backward (reverse-time) SDE [22]:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}\bar{t} + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{2.2}$$

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \mathbf{s}_\theta(\mathbf{x}, t)\right] \mathrm{d}\bar{t} + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{2.3}$$

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \mathbf{s}_\theta(\mathbf{x}, t)\right] \mathrm{d}\bar{t} - \frac{1}{2} g(t)^2 \mathbf{s}_\theta(\mathbf{x}, t)\mathrm{d}\bar{t} + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{2.4}$$

where $\bar{\cdot}$ denotes time traveling backward from 1 to 0, and $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ in the drift term is the *score function* of intermediate distribution at time t. The backward SDE induces exactly the same intermediate distribution p_t as the forward SDE. Intuitively, the forward process describes a natural degradation from data to Gaussian distribution, and the backward process iteratively reverses such process to reconstruct clean data. Further, the backward SDE has a marginally equivalent probability flow ODE [35]:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}\bar{t}, \tag{2.5}$$

Both backward SDE (Equation 2.4) and ODE (Equation 2.5) progressively recover p_0 from the prior p_1. Figure 2.1 summarizes the forward process and the corresponding backward SDE/ODE in diffusion models. To construct a generative model from this framework, estimating the score function by neural network $\mathbf{s}_\theta$ suffices. Let's denote the model distribution $p_\theta(\mathbf{x})$ as the distribution at time $t = 0$ in the following SDE, which substitutes the score function with neural functions in Equation 2.4:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \mathbf{s}_\theta(\mathbf{x}, t)\right] \mathrm{d}\bar{t} + g(t)\mathrm{d}\bar{\mathbf{w}}$$

If one has an accurate approximation of the score function, given by

$$\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}),$$

then it can be demonstrated that the model distribution closely approximates the data distribution under certain mild conditions [36]: $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$. Consequently, diffusion models meet the two criteria outlined in Section 2.1: the ability to accurately approximate the data distribution and to allow sampling from the model distribution. There are two types of commonly used SDEs in the literature, *i.e.,* Variance Exploding (VE) and Variance Preserving (VP), considered in [37]. For VE, $\mathbf{f}(\mathbf{x}, t) = 0, g(t) = \sigma(t)\sqrt{2\log \sigma_{\max}/\sigma_{\min}}$, where $\sigma(t) = \sigma_{\min}(\sigma_{\max}/\sigma_{\min})^t$; for VP, $\mathbf{f}(\mathbf{x}, t) = -\frac{1}{2}\beta(t), g(t) = \sqrt{\beta(t)}$, where $\beta(t) = (\beta_{\max} - \beta_{\min})t + \beta_{\min}$.

Some chapters in this thesis are built on a simplified version of the diffusion framework, termed EDM, proposed in [27]. The forward process in EDM has the simple form as follows:

$$\mathrm{d}\mathbf{x} = \sqrt{2\dot{\sigma}(t)\sigma(t)}\mathrm{d}\mathbf{w}, \tag{2.6}$$

where $\sigma(t)$ is typically set to t, and is ranging from 0 to some large values. It can be regarded as a special case of Equation 2.1, with $\mathbf{f}(\mathbf{x}, t) = 0, g(t) = \sqrt{2\dot{\sigma}(t)\sigma(t)}$. The forward process leads to a simple perturbation kernel, which will be discussed in the next section. The corresponding backward SDE and ODE are $\mathrm{d}\mathbf{x} = -2\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\mathrm{d}t + \sqrt{2\dot{\sigma}(t)\sigma(t)}\mathrm{d}\mathbf{w}$ and $\mathrm{d}\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\mathrm{d}t$, respectively.

2.3 Training

In the previous section, we have shown that by estimating the score of the transformed data distribution at time t, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, via a neural network $\mathbf{s}_\theta(\mathbf{x}, t)$, we are able to construct a valid generative model. A popular approach to learning the score function is to minimize a

weighted sum of the denoising score-matching objective [38]:

$$\min_{\theta} \mathbb{E}_{t\sim q_t(t)}\lambda(t)\mathbb{E}_{\mathbf{x}\sim p_0}\mathbb{E}_{\tilde{\mathbf{x}}\sim p_{t|0}(\cdot|\mathbf{x})}\left[\|\mathbf{s}_\theta(\tilde{\mathbf{x}},t)-\nabla_{\tilde{\mathbf{x}}}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2\right], \tag{2.7}$$

where q_t is the distribution for time variable, *e.g.*, $\mathcal{U}[0,1]$ for VE/VP [35] and a log-normal distribution for EDM [27], $\lambda(t)$ is a positive weighting function to keep the time-dependent loss around the same magnitude [35], p_0 is the data distribution and $p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})$ is the transition kernel denoting the conditional distribution of $\tilde{\mathbf{x}}$ given $\mathbf{x}$[2]. Specifically, diffusion models "destroy" data according to a diffusion process utilizing Gaussian transition kernels $p_{t|0}$. For example, the perturbation kernel pertaining to the EDM SDE (Equation 2.6) is Gaussian with varying time-dependent standard deviations: $p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x},\sigma(t)^2\boldsymbol{I})$

Although the training target in the objective is the conditional score $\nabla_{\tilde{\mathbf{x}}}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})$, the optimal model $\mathbf{s}_{\theta^*}(\tilde{\mathbf{x}},t)$ matches $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ almost everywhere given sufficient data and model capacity [38]. There are several equivalent objectives that can obtained by reparameterizing the neural network, such as $\mathbf{x}_0$-prediction, $\mathbf{v}$-prediction [39] and the preconditioning technique in [27], to reduce the variance or amplify the learning signal in the training target.

2.4 Sampling

After the estimation of the score function, we can sample from diffusion models via a reverse-time SDE [22] or a marginally-equivalent ODE [37], with the initial samples sampled from the known prior $\mathbf{x}\sim p_1(\mathbf{x})$:

$$\text{(SDE)} \qquad \mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x},t) - g(t)^2\mathbf{s}_\theta(\mathbf{x},t)\right]\mathrm{d}\bar{t} + g(t)\mathrm{d}\bar{\mathbf{w}} \tag{2.8}$$

$$\text{(ODE)} \qquad \mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x},t) - \frac{1}{2}g(t)^2\mathbf{s}_\theta(\mathbf{x},t)\right]\mathrm{d}\bar{t} \tag{2.9}$$

Prior works employ the Euler-Maruyama method, Predictor-Corrector solver [37] or adaptive step size method [28] to solve the diffusion SDE (Equation 2.8). To solve the diffusion ODE (Equation 2.9), previous papers utilize first-order (*e.g.,* forward Euler method)

[2]We omit "(0)" from $\mathbf{x}(0)$ when there is no ambiguity.

or higher-order (*e.g.,* Heun's method, RK45) numerical solvers [25], [27]. [26], [40] further utilize the multistep method to stabilize the sampling process. Diffusion ODE is generally faster to solve than diffusion SDE and is more commonly used in practical scenarios due to speed considerations.

Part I

Improved Training Techniques for Diffusion Models

When handling complex multi-modal data distributions, diffusion models often exhibit a high-variance training process, and their backward ODE trajectories are highly curved. In Part I, we focus on developing new training techniques aimed at stabilizing the training of diffusion models (Chapter 3) and reducing the curvature of generative trajectories (Chapter 4). We have observed that these enhanced training techniques consistently improve the models' performance across datasets and modalities, and result in higher-quality samples during inference.

Chapter 3

Reducing the Variance in the Score Estimation

Despite providing impressive empirical results, the training algorithms of diffusion models can be further improved by reducing the variance of the training targets in their objective for score estimation. We argue that the source of such variance lies in the handling of intermediate noise-variance scales, where multiple modes in the data affect the direction of reverse paths. In this chapter, we propose to remedy the problem by incorporating a reference batch into the training objective of diffusion models, to calculate weighted conditional scores. We show that the procedure indeed helps in the challenging intermediate regime by reducing (the trace of) the covariance of training targets. The new stable targets can be seen as trading bias for reduced variance, where the bias vanishes with increasing reference batch size. Empirically, we show that the new objective improves the image quality, stability, and training speed of various popular diffusion models across datasets with both general ODE and SDE solvers.

This chapter was previously published as [23].

3.1 Introduction

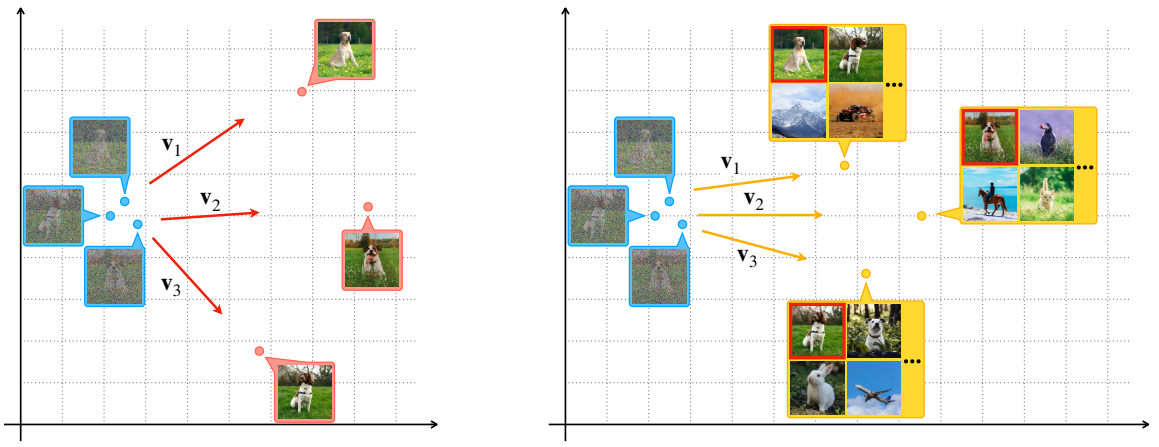
Diffusion models have recently achieved impressive results on a wide spectrum of generative tasks. However, we argue that, despite achieving impressive empirical results, the current training scheme of diffusion models can be further improved. In particular, the variance of

training targets in the denoising score-matching (**DSM**) objective can be large and lead to suboptimal performance. To better understand the origin of this instability, we decompose the score field into three regimes. Our analysis shows that the phenomenon arises primarily in the intermediate regime, which is characterized by multiple modes or data points exerting comparable influences on the scores. In other words, in this regime, the sources of the noisy examples generated in the course of the forward process become ambiguous. We illustrate the problem in Figure 3.1a, where each stochastic update of the score model is based on disparate targets.

In this chapter, we propose a generalized version of the denoising score-matching objective, termed the *Stable Target Field* (**STF**) objective. The idea is to include an additional *reference batch* of examples that are used to calculate weighted conditional scores as targets. We apply self-normalized importance sampling to aggregate the contribution of each example in the reference batch. Although this process can substantially reduce the variance of training targets (Figure 3.1b), especially in the intermediate regime, it does introduce some bias. However, we show that the bias together with the trace-of-covariance of the STF training targets shrinks to zero as we increase the size of the reference batch.

Experimentally, we show that our STF objective achieves new state-of-the-art performance on CIFAR-10 unconditional generation when incorporated into EDM [27]. The resulting FID score [41] is 1.90 with 35 network evaluations. STF also improves the FID/Inception scores for other variants of score-based models, *i.e.*, VE and VP SDEs [35], in most cases. In addition, it enhances the stability of converged score-based models on CIFAR-10 and CelebA 64^2 across random seeds, and helps avoid generating noisy images in VE. STF accelerates the training of score-based models (3.6× speed-up for VE on CIFAR-10) while obtaining comparable or better FID scores. To the best of our knowledge, STF is the first technique to accelerate the training process of diffusion models. We further demonstrate the performance gain with increasing reference batch size, highlighting the negative effect of large variance.

Our contributions are summarized as follows: **(1)** We detail the instability of the current diffusion models training objective in a principled and quantitative manner, characterizing a region in the forward process, termed *the intermediate phase*, where the score-learning targets are most variable. **(2)** We propose a generalized score-matching objective, *stable*



(a) DSM (b) STF

Figure 3.1: Illustration of differences between the DSM objective and our proposed STF objective. The "destroyed" images (in blue box) are close to each other while their sources (in red box) are not. Although the true score in expectation is the weighted average of $\mathbf{v}_i$, the individual training updates of the DSM objective have a high variance, which our STF objective reduces significantly by including a large reference batch (yellow box).

target field, which provides more stable training targets. **(3)** We analyze the behavior of the new objective and prove that it is asymptotically unbiased and reduces the trace-of-covariance of the training targets by a factor pertaining to the reference batch size in the intermediate phase under mild conditions. **(4)** We illustrate the theoretical arguments empirically and show that the proposed STF objective improves the performance, stability, and training speed of score-based methods. In particular, it achieves the current state-of-the-art FID score on the CIFAR-10 benchmark when combined with EDM.

3.2 Understanding the Training Target in Score-Matching Objective

The vanilla denoising score-matching objective at time t is:

$$\ell_{\text{DSM}}(\theta, t) = \mathbb{E}_{p_0(\mathbf{x})}\mathbb{E}_{p_{t|0}(\mathbf{x}(t)|\mathbf{x})}[\|\mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x})\|_2^2], \tag{3.1}$$

where the network is trained to fit the individual targets $\nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x})$ at $(\tilde{\mathbf{x}}, t)$ – the "influence" exerted by clean data $\mathbf{x}$ on $\mathbf{x}(t)$. We can swap the order of the sampling process

by first sampling $\tilde{\mathbf{x}}$ from p_t and then $\mathbf{x}$ from $p_{0|t}(\cdot|\tilde{\mathbf{x}})$. Thus, $\mathbf{s}_\theta$ has a closed form minimizer:

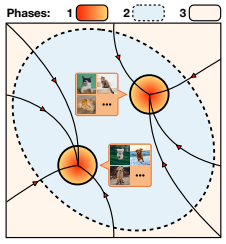
$$\mathbf{s}^*_{\mathrm{DSM}}(\mathbf{x}(t), t) = \mathbb{E}_{p_{0|t}(\mathbf{x}|\mathbf{x}(t))}[\nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x})] = \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t)). \tag{3.2}$$

The score field is a conditional expectation of $\nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})$ with respect to the posterior distribution $p_{0|t}$. In practice, a Monte Carlo estimate of this target can have high variance [42], [43]. In particular, when multiple modes of the data distribution have comparable influences on $\mathbf{x}(t)$, $p_{0|t}(\cdot|\mathbf{x}(t))$ is a multi-mode distribution, as also observed in [44]. Thus the targets $\nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x})$ vary considerably across different $\mathbf{x}$ and this can strongly affect the estimated score at $(\tilde{\mathbf{x}}, t)$, resulting in slower convergence and worse performance in practical stochastic gradient optimization [45].

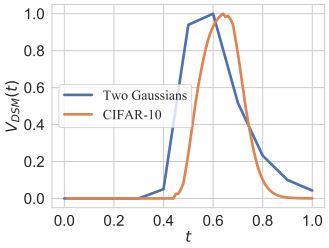
To quantitatively characterize the variations of individual targets at different time, we propose a metric – the average trace-of-covariance of training targets at time t:

$$\begin{aligned} V_{\mathrm{DSM}}(t) &= \mathbb{E}_{p_t(\tilde{\mathbf{x}})}\left[\mathrm{Tr}(\mathrm{Cov}_{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}(\nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})))\right] \\ &= \mathbb{E}_{p_t(\tilde{\mathbf{x}})}\mathbb{E}_{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\left[\|\nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})) - \nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}})\|_2^2\right]. \end{aligned} \tag{3.3}$$

We use $V_{\mathrm{DSM}}(t)$ to define three successive phases relating to the behavior of training targets. As shown in Figure 3.2a, the three phases partition the score field into near, intermediate, and far regimes (Phase 1∼3 respectively). Intuitively, $V_{\mathrm{DSM}}(t)$ peaks in the intermediate phase (Phase 2), where multiple distant modes in the data distribution have comparable influences on the same noisy perturbations, resulting in unstable targets. In Phase 1, the posterior $p_{0|t}$ concentrates around one single mode, thus low variation. In Phase 3, the targets remain similar across modes since $\lim_{t\to 1} p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}) \approx p_1$ for commonly used transition kernels.



(a) ODE Sampling



(b) $V_{\mathrm{DSM}}(t)$ versus t

Figure 3.2: **(a)**: Illustration of the three phases in a two-mode distribution. **(b)**: Estimated $V_{\mathrm{DSM}}(t)$ for two distributions. We normalize the maximum value to 1 for illustration purposes.

We validate this argument empirically in Figure 3.2b, which shows the estimated $V_{\text{DSM}}(t)$ for a mixture of two Gaussians as well as a subset of CIFAR-10 dataset [46] for a more realistic setting. Here we use VE SDE, *i.e.*, $p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}\left(\mathbf{x}, \sigma_m^2(\frac{\sigma_M}{\sigma_m})^{2t}\boldsymbol{I}\right)$ for some σ_m and σ_M [35]. $V_{\text{DSM}}(t)$ exhibits similar phase behavior across t in both toy and realistic cases. Moreover, $V_{\text{DSM}}(t)$ reaches its maximum value in the intermediate phase, demonstrating the large variations of individual targets. We defer more details to Appendix B.1.1.

3.3 Variance Reduction with Stable Target Field

The vanilla denoising score-matching approach (Equation 3.2) can be viewed as a Monte Carlo estimator, *i.e.*, $\nabla_{\mathbf{x}(t)} \log p_t(\tilde{\mathbf{x}}) = \mathbb{E}_{p_{0|t}(\mathbf{x}|\mathbf{x}(t))}[\nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x})] \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_i)$ where $\mathbf{x}_i$ is sampled from $p_{0|t}(\cdot|\tilde{\mathbf{x}})$ and $n = 1$. The variance of a Monte Carlo estimator is proportional to $\frac{1}{n}$, so we propose to use a larger batch (n) to counter the high variance problem described in Section 3.2. Since sampling directly from the posterior $p_{0|t}$ is not practical, we first apply importance sampling with the proposal distribution p_0. Specifically, we sample a large reference batch $\mathcal{B}_L = \{\mathbf{x}_i\}_{i=1}^n \sim p_0^n$ and get the following approximation:

$$\nabla_{\mathbf{x}(t)} \log p_t(\tilde{\mathbf{x}}) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{p_{0|t}(\mathbf{x}_i|\mathbf{x}(t))}{p_0(\mathbf{x}_i)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\mathbf{x}(t)|\mathbf{x}_i).$$

The importance weights can be rewritten as $p_{0|t}(\mathbf{x}|\mathbf{x}(t))/p_0(\mathbf{x}) = p_{t|0}(\mathbf{x}(t)|\mathbf{x})/p_t(\mathbf{x}(t))$. However, this basic importance sampling estimator has two issues. The weights now involve an unknown normalization factor $p_t(\mathbf{x}(t))$ and the ratio between the prior and posterior distribution can be large in high dimensional spaces. To remedy these problems, we appeal to self-normalization techniques [47] to further stabilize the training targets:

$$\nabla_{\mathbf{x}(t)} \log p_t(\tilde{\mathbf{x}}) \approx \sum_{i=1}^{n} \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_i)}{\sum_{j=1}^{n} p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_i). \tag{3.4}$$

We term this new training target in Equation 3.4 as *Stable Target Field (STF)*. In practice, we sample the reference batch $\mathcal{B}_L = \{\mathbf{x}_i\}_{i=1}^n$ from p_0^n and obtain $\mathbf{x}(t)$ by applying the transition

kernel to the "first" training data $\mathbf{x}_1$. Taken together, the new STF objective becomes:

$$\ell_{\mathrm{STF}}(\theta, t) = \mathbb{E}_{\{\mathbf{x}_i\}_{i=1}^n \sim p_0^n} \mathbb{E}_{\mathbf{x}(t) \sim p_{t|0}(\cdot|\mathbf{x}_1)} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right\|_2^2 \right]. \quad (3.5)$$

When $n = 1$, STF reduces to the vanilla denoising score-matching (Equation 3.1). When $n > 1$, STF incorporates a reference batch to stabilize training targets. Intuitively, the new weighted target assigns larger weights to clean data with higher influence on $\tilde{\mathbf{x}}$, *i.e.*, higher transition probability $p_{t|0}(\mathbf{x}(t)|\mathbf{x})$.

Similar to our analysis in Section 3.2, we can again swap the sampling process in Equation 3.5 so that, for a perturbation $\tilde{\mathbf{x}}$, we sample the reference batch $\mathcal{B}_L = \{\mathbf{x}_i\}_{i=1}^n$ from $p_{0|t}(\cdot|\mathbf{x}(t))p_0^{n-1}$, where the first element involves the posterior, and the rest follow the data distribution. Thus, the minimizer of the new objective (Equation 3.5) is (derivation can be found in Appendix A.1.1)

$$\mathbf{s}^*_{\mathrm{STF}}(\tilde{\mathbf{x}}, t) = \mathbb{E}_{\mathbf{x}_1 \sim p_{0|t}(\cdot|\tilde{\mathbf{x}})} \mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}} \left[\sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_j p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right]. \quad (3.6)$$

Note that although STF significantly reduces the variance, it introduces bias: the minimizer is no longer the true score. Nevertheless, in Section 3.4, we show that the bias converges to 0 as $n \to \infty$, while reducing the trace-of-covariance of the training targets by a factor of n when $p_{0|t} \approx p_0$. We further instantiate the STF objective (Equation 3.5) with transition kernels in the form of $p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma_t^2 \boldsymbol{I})$, which includes EDM [27], VP (through reparameterization) and VE [35]:

$$\mathbb{E}_{\mathbf{x}_1 \sim p_{0|t}(\cdot|\tilde{\mathbf{x}})} \mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \frac{1}{\sigma_t^2} \sum_{k=1}^n \frac{\exp\left(-\frac{\|\mathbf{x}(t) - \mathbf{x}_k\|_2^2}{2\sigma_t^2}\right)}{\sum_j \exp\left(-\frac{\|\mathbf{x}(t) - \mathbf{x}_j\|_2^2}{2\sigma_t^2}\right)} (\mathbf{x}_k - \mathbf{x}(t)) \right\|_2^2 \right].$$

To aggregate the time-dependent STF objective over t, we sample the time variable t from the training distribution q_t and apply the weighting function $\lambda(t)$. Together, the final training objective for STF is $\mathbb{E}_{t \sim q_t(t)} [\lambda(t) \ell_{\mathrm{STF}}(\theta, t)]$. We summarize the training process in Algorithm 1.

The small batch size $|\mathcal{B}|$ is the same as the normal batch size in the vanilla training process. We defer specific use cases of STF objectives combined with various popular diffusion models to Appendix A.1.4.

Algorithm 1 Learning the stable target field

Input: Training iteration T, Initial model $\mathbf{s}_\theta$, dataset $\mathcal{D}$, learning rate η.
for $t = 1 \dots T$ **do**
  Sample a large reference batch $\mathcal{B}_L$ from $\mathcal{D}$, and subsample a small batch $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{B}|}$ from $\mathcal{B}_L$
  Uniformly sample the time $\{t_i\}_{i=1}^{|\mathcal{B}|} \sim q_t(t)^{|\mathcal{B}|}$
  Obtain the batch of perturbed samples $\{\mathbf{x}_i(t_i)\}_{i=1}^{|\mathcal{B}|}$ by applying the transition kernel $p_{t|0}$ on $\mathcal{B}$
  Calculate the stable target field of $\mathcal{B}_L$ for all $\mathbf{x}_i(t_i)$:
    $\mathbf{v}_{\mathcal{B}_L}(\mathbf{x}_i(t_i)) = \sum_{\mathbf{x}\in\mathcal{B}_L} \frac{p_{t_i|0}(\mathbf{x}_i(t_i)|\mathbf{x})}{\sum_{\mathbf{y}\in\mathcal{B}_L} p_{t_i|0}(\mathbf{x}_i(t_i)|\mathbf{y})} \nabla_{\mathbf{x}_i(t_i)} \log p_{t_i|0}(\mathbf{x}_i(t_i)|\mathbf{x})$
  Calculate the loss: $\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \lambda(t_i) \|\mathbf{s}_\theta(\mathbf{x}_i(t_i), t_i) - \mathbf{v}_{\mathcal{B}_L}(\mathbf{x}_i(t_i))\|_2^2$
  Update the model parameter: $\theta = \theta - \eta \nabla \mathcal{L}(\theta)$
end for
return $\mathbf{s}_\theta$

3.4 Theoretical Analysis of Stable Target Field

In this section, we analyze the theoretical properties of our approach. In particular, we show that the new minimizer $\mathbf{s}^*_{\text{STF}}(\tilde{\mathbf{x}}, t)$ (Equation 3.6) converges to the true score asymptotically (Section 3.4.1). Then, we show that the proposed STF reduces the trace-of-covariance of training targets propositional to the reference batch size in the intermediate phase, with mild conditions (Section 3.4.2).

3.4.1 Asymptotic Behavior

Although in general $\mathbf{s}^*_{\text{STF}}(\tilde{\mathbf{x}}, t) \neq \nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}})$, the bias shrinks toward 0 with a increasing n. In the following theorem we show that the minimizer of STF objective at $(\tilde{\mathbf{x}}, t)$, *i.e.*, $\mathbf{s}^*_{\text{STF}}(\tilde{\mathbf{x}}, t)$, is asymptotically normal when $n \to \infty$.

Theorem 1. *Suppose* $\forall t \in [0,1], 0 < \sigma_t < \infty$, *then*

$$\sqrt{n}\left(\mathbf{s}^*_{STF}(\tilde{\mathbf{x}},t) - \nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}})\right) \xrightarrow{d} \mathcal{N}\left(\mathbf{0}, \frac{\mathrm{Cov}(\nabla_{\tilde{\mathbf{x}}} p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}))}{p_t(\tilde{\mathbf{x}})^2}\right) \tag{3.7}$$

We defer the proof to Appendix A.1.2. The theorem states that, for commonly used transition kernels, $\mathbf{s}^*_{\mathrm{STF}}(\tilde{\mathbf{x}},t) - \nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}})$ converges to a zero mean normal, and larger reference batch size (n) will lead to smaller asymptotic variance. As can be seen in Equation 3.7, when $n \to \infty$, $\mathbf{s}^*_{\mathrm{STF}}(\tilde{\mathbf{x}},t)$ highly concentrates around the true score $\nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}})$.

3.4.2 Trace of Covariance

We now highlight the small variations of the training targets in the STF objective compared to the DSM. As done in Section 3.2, we study the trace-of-covariance of training targets in STF:

$$V_{\mathrm{STF}}(t) = \mathbb{E}_{p_t(\tilde{\mathbf{x}})}\left[\mathrm{Tr}\left(\mathrm{Cov}_{p_{0|t}(\cdot|\tilde{\mathbf{x}})p_0^{n-1}}\left(\sum_{k=1}^{n} \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_j p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)\right)\right)\right].$$

In the following theorem we compare V_{STF} with V_{DSM}. In particular, we can upper bound $V_{\mathrm{STF}}(t)$ by

Theorem 2. *Suppose* $\forall t \in [0,1], 0 < \sigma_t < \infty$, *then*

$$V_{STF}(t) \leq \frac{1}{n-1}\left(V_{DSM}(t) + \frac{\sqrt{3}d}{\sigma_t^2}\sqrt{\mathbb{E}_{p_t(\tilde{\mathbf{x}})} D_f\left(p_0(\mathbf{x}) \parallel p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})\right)}\right) + O\left(\frac{1}{n^2}\right),$$

where D_f *is an f-divergence with* $f(y) = \begin{cases} (1/y-1)^2 & (y < 1.5) \\ 8y/27 - 1/3 & (y \geq 1.5) \end{cases}$. *Further, when* $n \gg d$ *and* $p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}}) \approx p_0(\mathbf{x})$ *for all* $\tilde{\mathbf{x}}$, $V_{STF}(t) \lessapprox \frac{V_{DSM}(t)}{n-1}$.

We defer the proof to Appendix A.1.3. The second term that involves f-divergence D_f is necessary to capture how the coefficients, *i.e.*, $p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)/\sum_j p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)$ used to calculate the weighted score target, vary across different samples $\tilde{\mathbf{x}}$. This term decreases monotonically as a function of t. In Phase 1, $p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})$ differs substantially from $p_0(\mathbf{x})$ and the divergence

term D_f dominates. In contrast to the upper bound, both $V_{\text{STF}}(t)$ and $V_{\text{DSM}}(t)$ have minimal variance at small values of t since the training target is always dominated by one $\mathbf{x}$. The theorem has more relevance in Phase 2, where the divergence term decreases to a value comparable to $V_{\text{DSM}}(t)$. In this phase, we empirically observe that the ratio of the two terms in the upper bound ranges from 10 to 100. Thus, when we use a large reference batch size (in thousands), the theorem implies that STF offers a considerably lower variance (by a factor of 10 or more) relative to the DSM objective. In Phase 3, the second term vanishes to 0, as $p_t \approx p_{t|0}$ with large σ_t for commonly used transition kernels. As a result, STF reduces the average trace-of-covariance of the training targets by at least $n-1$ times in the far field.

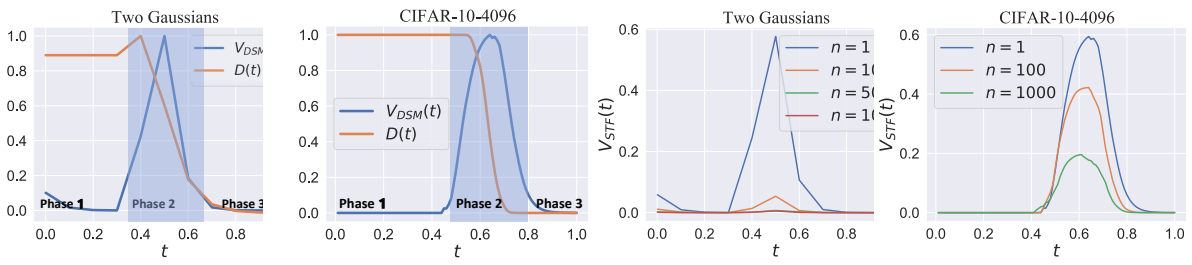
Together, we demonstrate that the STF targets have diminishing bias (Theorem 1) and are much more stable during training (Theorem 2). These properties make the STF objective more favorable for diffusion models training with stochastic gradient optimization.

3.5 Experiments

In this section, we first empirically validate our theoretical analysis in Section 3.4, especially for variance reduction in the intermediate phase (Section 3.5.1). Next, we show that the STF objective improves various diffusion models on image generation tasks in terms of image quality (Section 3.5.2). In particular, STF achieves state-of-the-art performance on top of EDM. In addition, we demonstrate that STF accelerates the training of diffusion models (Section 3.5.3), and improves the convergence speed and final performance with an increasing reference batch size (Section 3.5.3).

3.5.1 Variance Reduction in the Intermediate Phase

The proposed Algorithm 1 utilizes a large reference batch to calculate the stable target field instead of the individual target. In addition to the theoretical analysis in Section 3.4, we provide further empirical study to characterize the intermediate phase and verify the variance reduction effects by STF. Apart from $V(t)$, we also quantify the average divergence between the posterior $p_{0|t}(\cdot|\tilde{\mathbf{x}})$ and the data distribution p_0 at time t (introduced in Theorem 2): $D(t) = \mathbb{E}_{p_t(\tilde{\mathbf{x}})}\left[D_f\left(p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}}) \parallel p_0(\mathbf{x})\right)\right]$. Intuitively, the number of high-density modes in



(a) (b) (c) (d)

Figure 3.3: **(a, b)**: $V_{\mathrm{DSM}}(t)$ and $D(t)$ versus t. We normalize the maximum values to 1 for illustration purposes. **(c, d)**: $V_{\mathrm{STF}}(t)$ with a varying reference batch size n.

$p_{0|t}(\cdot|\tilde{\mathbf{x}})$ grows as $D(t)$ decreases. To investigate their behaviors, we construct two synthetic datasets: (1) a 64-dimensional mixture of two Gaussian components (**Two Gaussians**), and (2) a subset of 1024 images of CIFAR-10 (**CIFAR-10-4096**).

Figure 3.3a and Figure 3.3b show the behaviors of $V_{\mathrm{DSM}}(t)$ and $D(t)$ on Two Gaussian and CIFAR-10-4096. In both settings, $V_{\mathrm{DSM}}(t)$ reaches its peak in the intermediate phase (Phase 2), while $D(t)$ gradually decreases over time. These results agree with our theoretical understanding from Section 3.2. In Phase 2 and 3, several modes of the data distribution have noticeable influences on the scores, but only in Phase 2 are the influences much more distinct, leading to high variations of the individual target $\nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}), \mathbf{x} \sim p_{0|t}(\cdot|\tilde{\mathbf{x}})$.

Figure 3.3c and Figure 3.3d further show the relationship between $V_{\mathrm{STF}}(t)$ and the reference batch size n. Recall that when $n = 1$, STF degenerates to individual target and $V_{\mathrm{STF}}(t) = V_{\mathrm{DSM}}(t)$. We observe that $V_{\mathrm{STF}}(t)$ decreases when enlarging n. In particular, the predicted relation $V_{\mathrm{STF}}(t) \lessapprox V_{\mathrm{DSM}}(t)/(n-1)$ in Theorem 2 holds for the two Gaussian datasets where D_f is small. On the high dimensional dataset CIFAR-10-4096, the stable target field can still greatly reduce the training target variance with large reference batch sizes n.

3.5.2 Image Generation

We demonstrate the effectiveness of the new objective on image generation tasks. We consider CIFAR-10 [46] and CelebA 64×64 [51] datasets. We set the reference batch size n to 4096 (CIFAR-10) and 1024 (CelebA 64^2). We choose the current state-of-the-art score-based method EDM [27] as the baseline, and replace the DSM objective with our

Table 3.1: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE).

Methods	Inception ↑	FID ↓	NFE ↓
StyleGAN2-ADA [48]	9.83	2.92	1
DDPM [12]	9.46	3.17	1000
NCSNv2 [49]	8.40	10.87	1161
PFGM [50]	9.68	2.48	104
VE [35]			
DSM - RK45	9.27	8.90	264
STF (ours) - RK45	9.52 ↑	5.51 ↓	200
DSM - PC	9.68	2.75	2000
STF (ours) - PC	9.86 ↑	2.66 ↓	2000
VP [35]			
DSM - DDIM	9.20	5.16	100
STF (ours) - DDIM	9.28 ↑	5.06 ↓	100
DSM - RK45	9.46	2.90	140
STF (ours) - RK45	9.43 ↓	2.99 ↑	140
EDM [27]			
DSM - Heun, NCSN++	9.82	1.98	35
STF (ours) - Heun, NCSN++	**9.93** ↑	**1.90** ↓	35
DSM - Heun, DDPM++	9.78	1.97	35
STF (ours) - Heun, DDPM++	9.79 ↑	1.92 ↓	35

STF objective during training. We also apply STF to two other popular diffusion models, VE/VP SDEs [35]. For a fair comparison, we directly adopt the architectures and the hyper-parameters in [52] and [35] for EDM and VE/VP respectively. In particular, we use the improved NCSN++/DDPM++ models [27] in the EDM scheme. To highlight the stability issue, we train three models with different seeds for VE on CIFAR-10. We provide more experimental details in Appendix B.1.1.

Numerical Solver. The reverse-time ODE and SDE in scored-based models are compatible with any general-purpose solvers. We use the adaptive solver RK45 method [35], [53] (RK45) for VE/VP and the popular DDIM solver [54] for VP. We adopt Heun's 2nd order method (Heun) and the time discretization proposed by [27] for EDM. For SDEs, we apply the predictor-corrector (PC) sampler used in [35]. We denote the methods in a objective-sampler format, *i.e.*, **A**-**B**, where **A** $\in$ {DSM, STF} and **B** $\in$ {RK45, PC, DDIM,

Heun}. We defer more details to Appendix B.1.1.

Results. For quantitative evaluation of the generated samples, we report the FID scores [41] (lower is better) and Inception [55] (higher is better). We measure the sampling speed by the average NFE (number of function evaluations). We also include the results of several popular generative models [11], [12], [48], [50] for reference.

Table 3.1 and Table 3.2 report the sample quality and the sampling speed on unconditional generation of CIFAR-10 and CelebA 64^2. Our main findings are: **(1) STF achieves new state-of-the-art FID scores for unconditional generation on CIFAR-10 benchmark.** As shown in Table 3.1, The STF objective obtains a FID of 1.90 when incorporated with the EDM scheme. To the best of our knowledge, this is the lowest FID score on the unconditional CIFAR-10 generation task. In addition, the STF objective consistently improves the EDM across the two architectures. **(2) The STF objective improves the performance of different diffusion models.** We observe that the STF objective improves the FID/Inception scores of VE/VP/EDM on CIFAR-10, for most ODE and SDE samplers. STF consistently provides performance gains for VE across datasets. Remarkably, our objective achieves much better sample quality using ODE samplers for VE, with an FID score gain of 3.39 on CIFAR-10, and 2.22 on Celeba 64^2. For VP, STF provides better results on the popular DDIM sampler, while suffering from a slight performance drop when using the RK45 sampler. **(3) The STF objective stabilizes the converged VE model with the RK45 sampler.** In Appendix B.1.2, we report the standard deviations of performance metrics for converged models with different seeds on CIFAR-10 with VE. We observe that models trained with the STF objective give more consistent results, with a smaller standard deviation of used metrics.

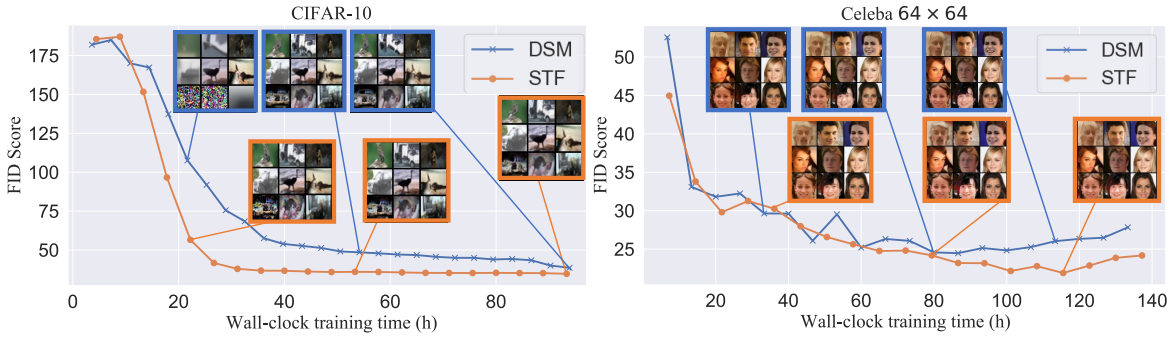
Table 3.2: FID and NFE on CelebA 64^2

Methods/NFEs	FID ↓	NFE ↓
CelebA 64^2 **- *RK45***		
VE (DSM)	7.56	260
VE (STF)	**5.34**	266
CelebA 64^2 **- *PC***		
VE (DSM)	9.13	2000
VE (STF)	**8.28**	2000

We further provide generated samples in Appendix B.1.3. One interesting observation is that when using the RK45 sampler for VE on CIFAR-10, the generated samples from the STF objective do not contain noisy images, unlike the vanilla DSM objective.

Effects of the Reference Batch Size According to our theory (Theorem 2), the upper bound of the trace-of-covariance of the STF target decreases proportionally to the reference batch size. Here we study the effects of the reference batch size (n) on model performances during training. The FID scores are evaluated on $1k$ samples using the RK45 sampler. As shown in Figure 3.5, models converge faster and produce better samples when increasing n. It suggests that smaller variations of the training targets can indeed speed up training and improve the final performances of diffusion models.

3.5.3 Accelerating Training of Diffusion Models



(a) CIFAR-10 (b) CelebA 64×64

Figure 3.4: FID and generated samples throughout training on (a) CIFAR-10 and (b) CelebA 64^2.

The variance-reduction techniques in neural network training can help to find better optima and achieve faster convergence rate [45], [56], [57]. In Figure 3.4, we demonstrate the FID scores every 50k iterations during the course of training. Since our goal is to investigate relative performance during the training process, and because the FID scores computed on 1k samples are strongly correlated with the full FID scores on 50k sample [49], we report FID scores on 1k samples for faster evaluations. We apply ODE samplers for FID evaluation, and measure the training time on two NVIDIA A100 GPUs. For a fair comparison, we report the average FID scores of models trained by the DSM and STF objective on VE versus the wall-clock training time (h).

The STF objective achieves better FID scores with the same training time, although the calculation of the target field by the reference batch introduces slight overhead (Algorithm 1). In Figure 3.4a, we show that the STF objective drastically accelerates the training of diffusion models on CIFAR-10. The STF objective achieves comparable FID scores with 3.6× less training time (25h versus 90h). The training time improvement for CelebA 64^2 datasets is less significant than on CIFAR-10. Our hypothesis is that the STF objective is more effective when there are multiple well-separated modes in data distribution, *e.g.*, the ten classes in CIFAR-10, where the DSM objective suffers from relatively larger variations in the intermediate phase. In addition, the converged models have better final performance when paired with the STF on both datasets.

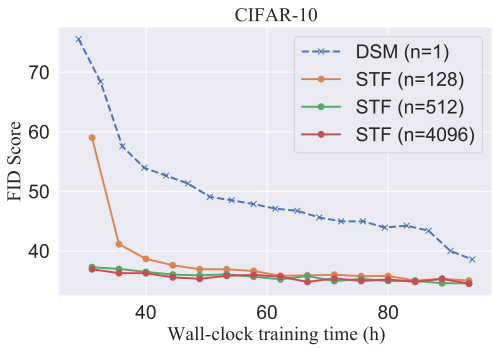


Figure 3.5: FID scores in the training with varying reference batch size.

3.6 Related Works

Different phases of diffusion models. The idea of diffusion models having different phases has been explored in prior works though the motivations and definitions vary [27], [58]. [27] argues that the training targets are difficult and unnecessary to learn in the very near field (small t in our Phase 1), whereas the training targets are always dissimilar to the true targets in the intermediate and far field (our Phase 2 and Phase 3). As a result, their solution is sampling t with a log-normal distribution to emphasize the relevant region (relatively large t in our Phase 1). In contrast, we focus on reducing large training target variance in the intermediate and far field, and propose STF to better estimate the true target (cf. [27]). [58] identifies a key region where the model learns perceptually rich contents, and determines the training weights $\lambda(t)$ based on the signal-to-noise ratio (SNR) at different t. As SNR is monotonically decreasing over time, the resulting up-weighted region does not match our Phase 2 characterization. In general, our proposed STF method reduces the training target variance in the intermediate field and is complementary to previous improvements of diffusion

models.

Importance sampling. The technique of importance sampling has been widely adopted in machine learning community, such as debiasing generative models [59], counterfactual learning [60] and reinforcement learning [61]. Prior works using importance sampling to improve generative model training include reweighted wake-sleep (RWS) [62] and importance weighted autoencoders (IWAE) [63]. RWS views the original wake-sleep algorithm [64] as importance sampling with one latent variable, and proposes to sample multiple latents to obtain gradient estimates with lower bias and variance. IWAE utilizes importance sampling with multiple latents to achieve greater flexibility of encoder training and tighter log-likelihood lower bound compared to the standard variational autoencoder [65], [66].

Variance reduction for Fisher divergence. One popular approach to score-matching is to minimize the Fisher divergence between true and predicted scores [67]. [68] links the Fisher divergence to denoising score-matching [38] and studies the large variance problem (in $O(1/\sigma_t^4)$) of the Fisher divergence when $t \to 0$. They utilize a control variate to reduce the variance. However, this is typically not a concern for current diffusion models as the time-dependent objective can be viewed as multiplying the Fisher divergence by $\lambda(t) = \sigma_t^2$, resulting in a finite-variance objective even when $t \to 0$.

3.7 Conclusion

We identify large target variance as a significant training issue affecting diffusion models. We define three phases with distinct behaviors, and show that the high-variance targets appear in the intermediate phase. As a remedy, we present a generalized score-matching objective, *Stable Target Field* (STF), whose formulation is analogous to the self-normalized importance sampling via a large reference batch. Albeit no longer an unbiased estimator, our proposed objective is asymptotically unbiased and reduces the trace-of-covariance of the training targets, which we demonstrate theoretically and empirically. We show the effectiveness of our method on image generation tasks, and show that STF improves the performance, stability, and training speed over various state-of-the-art diffusion models. Future directions include a principled study on the effect of different reference batch sampling procedures. Our presented

approach is uniformly sampling from the whole dataset $\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}$, so we expect that training diffusion models with a reference batch of more samples in the neighborhood of $\mathbf{x}_1$ (the sample from which $\tilde{\mathbf{x}}$ is perturbed) would lead to an even better estimation of the score field. Moreover, the three-phase analysis can effectively capture the behaviors of other physics-inspired generative models, such as PFGM [50] or the more advanced PFGM++ [31]. Therefore, we anticipate that STF can enhance the performance and stability of these models further.

Chapter 4

Towards Straighter Diffusion Trajectories with Discrete Latents

Another challenge in the training of diffusion models (DMs) lies in their complex noise-to-data mapping. The generative ODE in diffusion models defines a highly non-linear mapping from a uni-modal Gaussian noise to the multi-modal data distribution, which has strong curvature for their ODE trajectories and arguably represents an unnecessarily challenging learning problem. How can we simplify the mapping? This chapter proposes Discrete-Continuous Latent Variable Diffusion Models (DisCo-Diff) to simplify this task by introducing complementary discrete latent variables. We augment DMs with learnable discrete latents, inferred with an encoder, and train DM and the encoder end-to-end. DisCo-Diff does not rely on pre-trained networks, making the framework universally applicable. The discrete latents significantly simplify learning the DM's complex noise-to-data mapping by reducing the curvature of the DM's generative ODE. An additional autoregressive transformer models the distribution of the discrete latents, a simple step because DisCo-Diff requires only few discrete variables with small codebooks. We validate DisCo-Diff on toy data, several image synthesis tasks as well as molecular docking, and find that introducing discrete latents consistently improves model performance.

This chapter is based on the paper [32].

4.1 Introduction

The generation of diffusion models (DMs) can be formulated either as a stochastic (diffusion SDE) or, more conveniently, as a deterministic process (diffusion ODE) that takes as input random noise from the Gaussian prior and transforms it into data through a generative ordinary differential equation (ODE) [24]. The Gaussian prior corresponds to the DM's *continuous latent variables*, where the data is uniquely encoded through the ODE-defined mapping.

However, realistic data distributions are typically high-dimensional, complex and often multimodal. Directly encoding such data into a single unimodal Gaussian distribution and learning a corresponding reverse noise-to-data mapping is challenging. The mapping, or generative ODE, necessarily needs to be highly complex, with strong curvature, and one may consider it unnatural to map an entire data distribution to a single Gaussian distribution. In practice, conditioning information, such as class labels or text prompts, often helps to simplify the complex mapping by offering the DM's denoiser additional cues for more accurate denoising. However, such conditioning information is typically of a semantic nature and, even given a class or text prompt, the mapping remains highly complex. For instance, in the case of images, even within a class we find images with vastly different styles and color patterns, which corresponds to large distances in pixel space.

In this chapter, we propose ***Dis****crete-****Co****ntinuous Latent Variable* ***Diff****usion Models (DisCo-Diff)*, DMs augmented with additional *discrete* latent variables that encode additional high-level information about the data and can be used by the main DM to simplify its denoising task (Figure 4.1). These discrete latents are inferred through an encoder network and learnt end-to-end together with the DM. Thereby, the discrete latents directly learn to encode information that is beneficial for reducing the DM's score matching objective and making the DM's hard task of mapping simple noise to complex data easier. Indeed, in practice, we find that they significantly reduce the curvature of the DM's generative ODE and reduce the DM training loss in particular for large diffusion times, where denoising is most ambiguous and challenging. In contrast to previous work [70]–[72], we do not rely on domain-specific pre-trained encoder networks, making our framework general and universally applicable. To facilitate

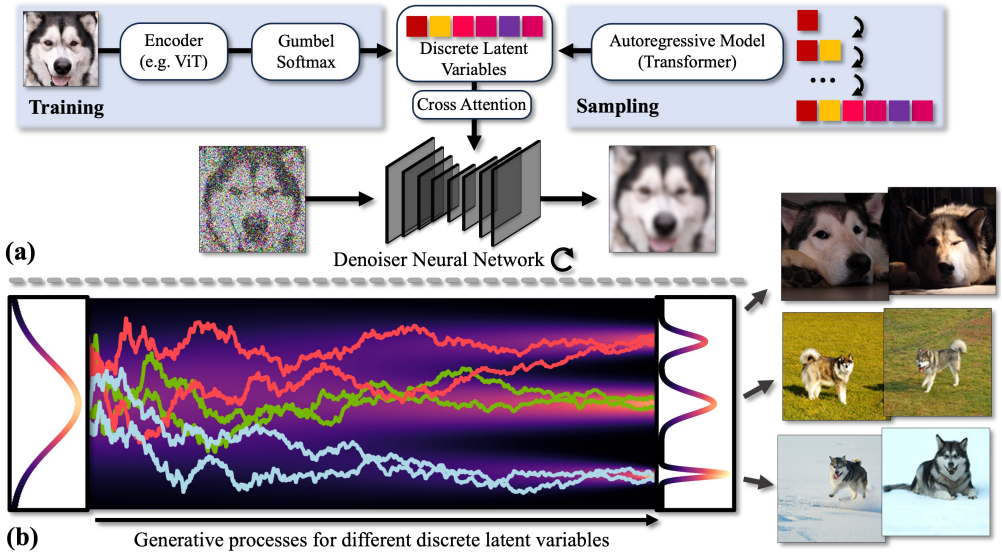


Figure 4.1: **Dis**crete-**Co**ntinuous Latent Variable **Diff**usion Models (DisCo-Diff) augment DMs with additional *discrete* latent variables that capture global appearance patterns, here shown for images of huskies. **(a)** During training, discrete latents are inferred through an encoder, for images a vision transformer [69], and fed to the DM via cross-attention. Backpropagation is facilitated by continuous relaxation with a Gumbel-Softmax distribution. To sample novel images, an additional autoregressive model is learnt over the distribution of discrete latents. **(b)** Schematic visualization of generative denoising diffusion trajectories. Different colors indicate different discrete latent variables, pushing the trajectories toward different modes.

sampling of discrete latent variables during inference, we learn an autoregressive model over the discrete latents in a second step. We only use a small set of *discrete* latents with relatively small codebooks, which makes the additional training of the autoregressive model easy. We specifically advocate for the use of auxiliary discrete instead of continuous latents; see 4.2.2.

While previous works [73]–[78] use fully discrete latent variable-based approaches to model images, this typically requires large sets of spatially arranged latents with large codebooks, which makes learning their distribution challenging. DisCo-Diff, in contrast, carefully combines its discrete latents with the continuous latents (Gaussian prior) of the DM and effectively separates the modeling of discrete and continuous variations within the data. It requires only a few discrete latents.

To demonstrate its universality, we validate the DisCo-Diff framework on several different tasks. As a motivating example, we study 2D toy distributions, where the discrete latents learn to capture different modes with smaller curvature during sampling. We then tackle image synthesis, where the discrete latents learn large-scale appearance, often associated with global style and color patterns. Thereby, they offer complementary benefits to semantic conditioning information. Quantitatively, DisCo-Diff universally boosts output quality and achieves state-of-the-art performance on several ImageNet generation benchmarks. In addition, we experimentally validate that auxiliary *discrete* latents are superior to *continuous* latents in our setup, and study different network architectures for injecting the discrete latents into the DM network. A careful hierarchical design can encourage different discrete latents to encode different image characteristics, such as shape vs. color, reminiscent of observations from the literature on generative adversarial networks [79], [80]. We also apply DisCo-Diff to molecular docking, a critical task in drug discovery, where the discrete latents again improve performance by learning to indicate critical atoms in the interaction and, in this way, deconvolving the multimodal uncertainty given by different possible poses from continuous variability of each pose. Moreover, we augment Poisson Flow Generative Models [81], [82] with discrete latent variables to showcase that the framework can also be applied to other "iterative" generative models, other than regular DMs, observing similar benefits.

Contributions. *(i)* We propose DisCo-Diff, a novel framework for combining discrete and continuous latent variables in DMs in a universal manner. *(ii)* We extensively validate

(a) 128×128 (b) Shared discrete latents

Figure 4.2: Samples generated from DisCo-Diff trained on the ImageNet dataset: (a) randomly sampled discrete latents and class labels; (b) samples in each grid sharing the same discrete latent. The class label for the top/bottom row is fixed to coffeepot/malamute.

DisCo-Diff, significantly boosting model quality in all experiments, and achieving state-of-the-art performance on several image synthesis tasks. *(iii)* We present detailed analyses as well as ablation and architecture design studies that demonstrate the unique benefits of discrete latent variables and how they can be fed to the main denoiser network. *(iv)* Overall, we provide insights for designing performant generative models. We make the case for discrete latents by showing that real-world data is best modeled with generative frameworks that leverage *both* discrete and continuous latents. We intentionally developed a simple and universal framework that does not rely on pre-trained encoders to offer a broadly applicable modeling approach to the community.

4.2 Augmenting Diffusion Models with Discrete Latents

In Section 4.2.1, we first formally define DisCo-Diff's generative model and training framework, before discussing and carefully motivating our approach in detail in Section 4.2.2. In Section 4.2.3, we highlight critical architecture considerations.

4.2.1 Two-Stage Training Procedure

In our DisCo-Diff framework (Figure 4.1), we augment a DM's learning process with an m-dimensional discrete latent $\mathbf{z} \in \mathbb{N}^m$, where each dimension is a random variable from a categorical distribution of codebook size k. There are three learnable components: the denoiser neural network $\boldsymbol{D}_\theta \colon \mathbb{R}^d \times \mathbb{R} \times \mathbb{N}^m \to \mathbb{R}^d$, corresponding to DisCo-Diff's DM, which predicts denoised images conditioned on diffusion time t and discrete latent $\mathbf{z}$; an encoder $\boldsymbol{E}_\phi \colon \mathbb{R}^d \to \mathbb{N}^m$, used to infer discrete latents given clean images $\mathbf{y}$. It outputs a categorical distribution over the k categories for each discrete latent; and a post-hoc auto-regressive model $\boldsymbol{A}_\psi$, which approximates the distribution of the learned discrete latents $\mathbf{z}$ by $\prod_{i=1}^m p_\psi(\mathbf{z}_i|\mathbf{z}_{<i})$. DisCo-Diff's training process is divided into two stages. In the first stage, the denoiser $\boldsymbol{D}_\theta$ and the encoder $\boldsymbol{E}_\phi$ are co-optimized in an end-to-end fashion. This is achieved by extending the denoising score matching objective (as expressed in Eq. 2.7) to include learnable discrete latents $\mathbf{z}$ associated with each data $\mathbf{y}$:

$$\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{z}\sim\boldsymbol{E}_\phi(\mathbf{y})}\mathbb{E}_{t,\mathbf{n}}\left[\lambda(t)||\boldsymbol{D}_\theta(\mathbf{y}+\mathbf{n},t,\mathbf{z})-\mathbf{y}||^2\right], \tag{4.1}$$

where $\mathbf{y}$ is sampled from the data distribution $p_0(\mathbf{y})$. In contrast to the standard objective in Equation 2.7, which focuses on learning the reparameterization of the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, the denoiser in our approach is essentially learning the reparameterization of the conditional score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{z})$ by $\mathbf{x}_0$-prediction: the score function in Equation 2.7 can be expressed as $\mathbf{s}^*_\theta(\mathbf{x},\mathbf{z},t) = (\boldsymbol{D}_\theta(\mathbf{x},t,\mathbf{z})-\mathbf{x})/t^2$. The convolution of the probability density functions $p_t(\cdot|\mathbf{z}) = p(\cdot|\mathbf{z}) * \mathcal{N}(\mathbf{0}, t^2\boldsymbol{I})$. This conditional score originates from conditioning the DM on the discrete latents $\mathbf{z}$, which are inferred by the encoder $\boldsymbol{E}_\phi$. The denoiser network $\boldsymbol{D}_\theta$ can better capture the time-dependent score (*i.e.,* achieving a reduced loss) if the score for each sub-distribution $p_t(\mathbf{x}|\mathbf{z})$ is simplified. Therefore, the encoder $\boldsymbol{E}_\phi$, which has access to clean input data, is encouraged to encode useful information into the discrete latents and help the denoiser to more accurately reconstruct the data. Naively backpropagating gradients into the encoder through the sampling of the discrete latent variables $\mathbf{z}$ is not possible. Hence, during training we rely on a continuous relaxation based on the Gumbel-Softmax distribution [83] (see B.2.2 for details).

When training the denoiser network, we randomly replace the discrete latent variables

with a non-informative null-embedding with probability 0.1. Thereby, the DM learns both a discrete latent variable-conditioned and a regular, unconditional score. During sampling, we can combine these scores for classifier-free guidance [84] with respect to the model's own discrete latents, and amplify their conditioning effect (details in 4.3.1).

We can interpret DisCo-Diff as a variational autoencoder (VAE) [66], [85]–[87] with discrete latents and a DM as decoder. VAEs often employ regularization on their latents. We did not find this to be necessary, as we use only very low-dimensional latent variables, *e.g.,* 10 in our ImageNet experiments, with relatively small codebooks. Moreover, we employ a strictly non-zero temperature in the Gumbel-Softmax relaxation, encouraging stochasticity.

In the second stage, we train the autoregressive model $\boldsymbol{A}_\psi$ to capture the distribution of the discrete latent variables $p_\phi(\mathbf{z})$ defined by pushing the clean data through the trained encoder. We use a maximum likelihood objective as follows:

$$\mathbb{E}_{\mathbf{y}\sim p_0(\mathbf{y}),\mathbf{z}\sim\boldsymbol{E}_\phi(\mathbf{y})}\left[\sum_{i=1}^{m}\log p_\psi(\mathbf{z}_i|\mathbf{z}_{<i})\right] \tag{4.2}$$

Since we set m to a relatively small number, it becomes very easy for the model to handle such short discrete vectors, which makes this second-stage training efficient. Also, the additional sampling overhead due to this autoregressive component on top of the DM becomes negligible. At inference time, when using DisCo-Diff to generate novel samples, we first sample a discrete latent variable from the autoregressive model, and then sample the DM with an ODE or SDE solver.

4.2.2 Reduced Curvature through End-to-end training

We will now critically discuss and motivate our design choices and also discuss the most relevant related works. *For an extended discussion of related work see Section 4.4.*

The curvature of diffusion models. DMs, in their simpler ODE-based formulation ($\beta(t) = 0$ in 2.4), learn a complex noise-to-data mapping. The noise is drawn from an analytically tractable, unimodal Gaussian distribution. As the data is encoded in this distribution, we can consider this high-dimensional Gaussian distribution the DM's *continuous* latent variables (DMs can generally be seen as deep latent variable models [88], [89]). However,

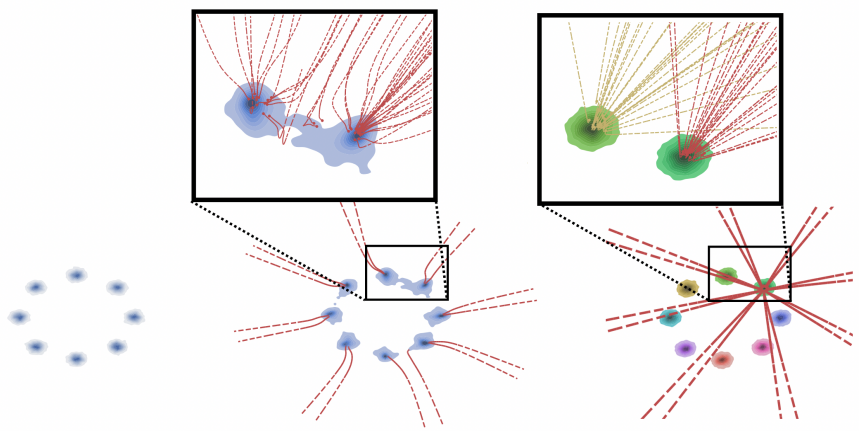
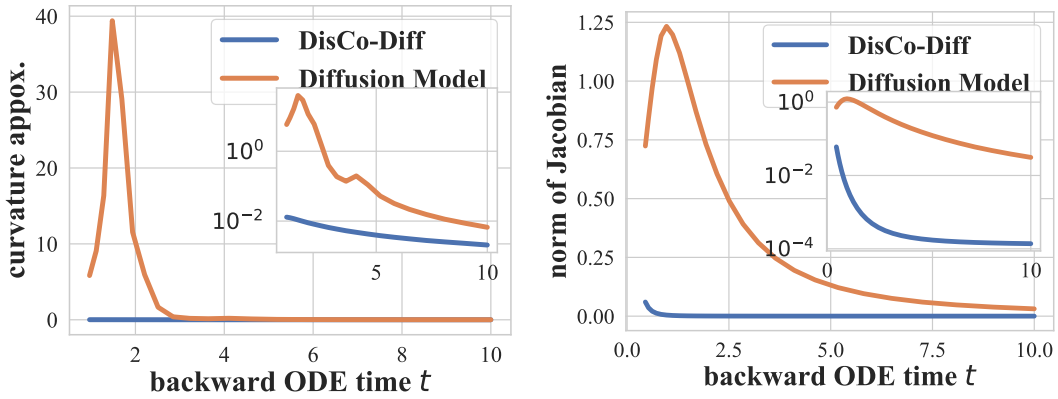


Figure 4.3: **Modeling 2D mixture of Gaussians.** *Left:* Data distribution. *Middle:* Generated data by regular DM. *Right:* Generated data by DisCo-Diff. We use different colors to distinguish data generated by different discrete latents. We further provide zoom-ins and visualize some ODE trajectories by dotted lines.



(a) (b)

Figure 4.4: **Modeling 2D mixture of Gaussians: analysis.** The mean curvature *(left)* and norm of the neural networks' Jacobians *(right)* along the reverse-time ODE trajectories as function of t.

the mapping from unstructured noise to a diverse, typically multimodal data distribution necessarily needs to be highly complex. This corresponds to a highly non-linear generative ODE with strong curvature, which is challenging to learn and also makes synthesis slow by requiring a fine discretization. To illustrate this point, we trained a DM on a simple 2D mixture of Gaussians, where we observe bent ODE trajectories near the data (Figure 4.3, *middle*). This effect is significantly stronger in high dimensions.

A simpler mapping with discrete latent variables. The role of the discrete latents in DisCo-Diff is to reduce this complexity and make the DM's learning task easier. The single

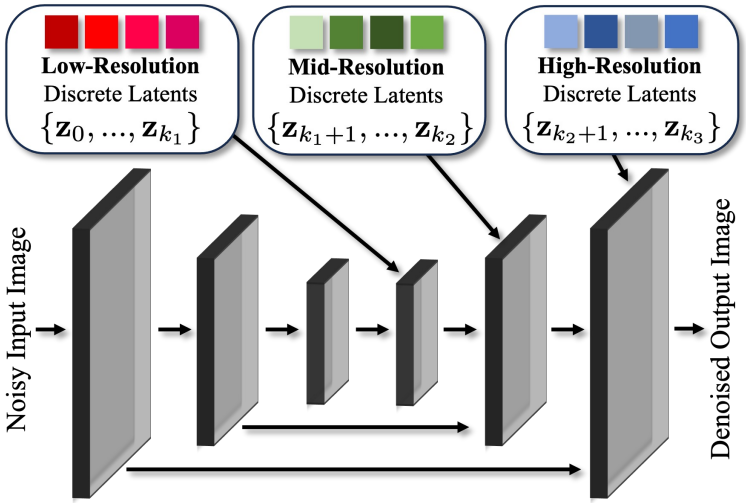


Figure 4.5: **Group hierarchical DisCo-Diff.** Different discrete latents are fed to the denoiser U-Net at different feature resolutions.

noise-to-data mapping is effectively partitioned into a set of simpler mappings, each with less curvature in its generative ODE. We argue that it is unnatural to map an entire multimodal complex data distribution to a single continuous Gaussian distribution. Instead, we believe that an ideal generative model should combine both discrete and continuous latent variables, where discrete latents capture global multimodal structure and the continuous latents model local continuous variability. With this in mind, we suggest to only use a moderate number of discrete latents with small codebooks. On the one hand, a few latents can already significantly simplify the DM's learning task. On the other hand, if we only have few latents with small codebooks, training a generative model—an autoregressive one in our case—over the discrete latent variable distribution itself, will be simple (which we observe, see Section 4.3).

Validation in 2D. To validate our reasoning, let us revisit the toy 2D mixture of Gaussians. In Figure 4.3, *right*, we show the DisCo-Diff model's synthesized data. The discrete latents learn to capture the different modes, and DisCo-Diff's DM component models the individual modes. The DM's ODE trajectories for different latents are now almost perfectly straight, indicating a simple conditional score function. In Figure 4.4, *left*, we quantitatively show strongly reduced curvature along the entire diffusion time t. In Figure 4.4, *right*, as a measure of network complexity we also show the norms of the Jacobians of the employed denoiser networks. We see significantly reduced norms for DisCo-Diff for all t, suggesting that the denoiser's task is indeed strongly simplified and less network capacity is required.

Using few, global latents with relatively small codebooks is important. DisCo-Diff is fundamentally different from most contemporary generative modeling frameworks using discrete latent variables [73]–[78], [86]. These works use autoencoders to encode images in its entirety into spatially-arranged discrete tokens, essentially a down-sampled version of the input. However, this is also unnatural: Encoding continuous variability, like smooth pose, shape or color variations in images, into discrete latents requires the use of very large codebooks and, on top of that, these models generally rely on very high-dimensional spatial grids of discrete latents (*e.g.* 32x32=1024 latents with codebooks $>1,000$ [73], while we use just 10 latents with a codebook size of 100 in our main image models). This makes learning the distribution over the discrete latents very challenging for these types of models, while it is simple in DisCo-Diff, where they just supplement the DM. In DisCo-Diff, we get the best from both continuous and discrete latent variables, using only few *global* latents.

End-to-end training is essential. DisCo-Diff's discrete latents are in spirit similar to leveraging non-learnt conditioning information. As pointed out by [70], this has been crucial to facilitate training high-performance generative models like strong class-conditional [90], [91] or text-to-image DMs [3], [4], [92]. However, DisCo-Diff aims to fundamentally address the problem, rather than relying on given conditioning data. Moreover, the data usually has significant variability even given, for instance, a class label. Our discrete latents can further reduce the complexity (as observed, see Section 4.3).

However, could we use pre-trained encoder networks, such as CLIP [93] or others [94], [95], to produce encodings to condition on and whose distribution could be modeled in a second stage? This is explored by previous works [70]–[72], [96], but has important disadvantages: *(i)* The most crucial downside is that such encoders are not universally available, but typically only for images. However, we seek to develop a universally applicable framework. For instance, we also apply DisCo-Diff to molecular docking (see Section 4.3.2), where no suitable pre-trained networks are available. *(ii)* In DisCo-Diff, the job of the discrete latents is to make the denoising task of the DM easier, which is especially ambiguous at large noise levels (in fact, we find that the latents help in particular to reduce the loss at these high noise levels, see Figure 4.7). It is not obvious what information about the data the latents should best encode for this. By learning them jointly with the DM objective itself, they are directly

trained to help the DM learn better denoisers and lower curvature generative ODEs. *(iii)* A generative model needs to be trained over the encodings in the second stage. In DisCo-Diff, we can freely choose an appropriate number of latents and codebook size to simplify the DM's denoising task, while also facilitating easy learning of the autoregressive model in the second stage. When using pre-trained encoders, one must work with the encodings by these methods, which were not developed for generative modeling. We attribute DisCo-Diff's strong generation performance to its end-to-end learning.

The latent variables must be discrete. Could we also use auxiliary continuous latent variables? Continuous latents are almost always based on underlying Gaussian distributions. Hence, if such continuous latents learnt multimodal structure in the data to simplify the main DM's denoising task, as DisCo-Diff's discrete latents do, then learning a distribution over them in the second stage would again require a highly non-linear difficult-to-learn mapping from Gaussian noise to the multimodal encodings. This is the problem DisCo-Diff aims to solve in the first place. [97] augment DMs with non-spatial continuous latent variables, but they only focus on semantic face image manipulation. InfoDiffusion [98] conditions DMs on discrete latent variables. However, it focuses on learning disentangled representations, also primarily for low-resolution face synthesis, and uses a mutual information-based objective. Contrary to DisCo-Diff, neither of these works tackles high-quality synthesis for challenging, diverse datasets.

In our ablation studies (Section 4.3.1), we further validate our design choices and motivations that we presented here.

4.2.3 Architecture

As discussed, DisCo-Diff enhances the training of continuous DMs by incorporating learnable discrete latent variables that are meant to capture the *global* underlying discrete structure of the data. To ensure that DisCo-Diff works as intended, suitable network architectures are necessary. Below, we summarize our design choices, focusing on DisCo-Diff for image synthesis. However, the framework is general, requiring only an encoder to infer discrete latents from clean input data and a conditioning mechanism that integrates these discrete latents into the denoiser network. In fact, we also apply our model to 2D toy data and

Table 4.1: FID score together with NFE on ImageNet-64.

	FID	NFE
without class-conditioning		
IC-GAN [99]	9.20	1
BigGAN [100]	16.90	1
iDDPM [101]	16.38	50
EDM [27]	6.20	50
SCDM [70]	3.94	50
DisCo-Diff (*ours*)	**3.70**	50
class-conditioned, ODE sampler		
EDM [27]	2.36	79
PFGM++ [82]	2.32	79
DisCo-PFGM++ (*ours*)	1.92	78
DisCo-Diff (*ours*)	**1.65**	78
class-conditioned, stochastic sampler		
iDDPM [101]	2.92	250
ADM [90]	2.07	250
CDM [102]	1.48	8000
VDM++ [91]	1.43	511
EDM (w/ Restart [29])	1.36	623
RIN [103]	1.23	1000
DisCo-Diff (*ours*; w/ Restart [29])	**1.22**	623
class-conditioned, w/ adversarial objective		
IC-GAN [99]	6.70	1
BigGAN-deep [100]	4.06	1
CTM [104]	1.92	1
StyleGAN-XL [105]	1.51	1

molecular docking.

Encoder. For image modeling, we utilize a ViT [69] as the backbone for the encoder. We extend the classification mechanism in ViTs, and treat each discrete token as a different classification token. Concretely, we add m extra classification tokens to the sequence of image patches. This architectural design naturally allows each discrete latent to effectively capture the global characteristic of the images, akin to performing data classification.

Discrete latent variable conditioning. For image experiments, DisCo-Diff's denoisers are U-Nets as widely used for DMs [27], [106]. For the discrete latent variable conditioning, we utilize cross-attention [4]. Drawing inspiration from text-to-image generation, DisCo-Diff's discrete latents function analogously to text, exerting a global influence on the denoiser's output. Specifically, image features act as queries and discrete latents are keys and values

Table 4.2: FID score and NFE on class-cond. ImageNet-128.

	FID	NFE
ADM [90]	5.91	250
ADM-G [90]	2.97	250
CDM (32, 64, 128) [102]	3.52	8100
RIN [103]	2.75	1000
VDM++, *w/ ODE sampler* [91]	2.29	115
DisCo-Diff, *w/ ODE sampler* (*ours*)	**2.08**	114

in the cross-attention layer, enabling discrete latents to globally shape the image features. We add a cross-attention layer after each self-attention layer within the U-Net. In our main models, all discrete latents are given to all cross-attention layers.

Group hierarchical models. To enhance the interpretability of discrete latents, we also explore the inductive bias inherent in the U-Net architecture and feed distinct latent groups into various resolution features in the up-sampling branch of the U-Net, as shown in Figure 4.5. This approach draws inspiration from StyleGAN [79], where distinct latents are introduced at different resolutions, enabling each to capture different image characteristics by the neural network's inductive bias. This design fosters a group hierarchy, where the groups associated with higher-resolution features offer supplementary information, conditioned upon the groups related to lower-resolution features. We refer to this refined model as the *group hierarchical* DisCo-Diff.

In the **molecular docking** task, existing denoisers operate through message passing in a permutation equivariant way over 3D point clouds representing molecular structures [107]. We build this property and architectural bias directly into the latent variables allowing them to take values indicating one node in the point cloud (therefore for every point cloud the codebook size equals the number of nodes). This latent design choice aligns with the intuition of the encoder determining the atoms playing key roles in the structure and allows for minimal modification of the score model where the latents simply represent additional features for every node. The encoder is also composed of a similar equivariant message passing, `e3nn` [108], network where for each node one logit per latent will be predicted. More details on the architecture for the molecular docking task can be found in Section B.2.2.

The **auto-regressive model** over the distribution of the discrete latents is implemented

Table 4.3: Ablations on class-cond. ImageNet-64.

	FID
EDM [27]	2.36
Oracle setting	
Continuous latent (KLD weight=0.1)	1.67
Continuous latent (KLD weight=1)	2.36
DisCo-Diff (cfg=0)	1.65
Generative prior on latent	
Continuous latent (KLD weight=0.1)	11.12
Continuous latent (KLD weight=1, cfg=0)	2.36
Continuous latent (KLD weight=1, cfg=1)	2.36
DisCo-Diff (cfg=0)	1.81
DisCo-Diff (cfg=1)	**1.65**
DisCo-Diff (cfg=2)	2.33

in image experiments using a standard Transformer decoder [109]. For molecular docking, it again uses an `e3nn` network that is fed the conditioning information of the protein structure and molecular graph. Generally, DisCo-Diff is compatible with other conditional inputs, *e.g.* class labels, which can be added as inputs to denoiser and auto-regressive model. We use an auto-regressive model for simplicity and expect DisCo-Diff's second stage to work equally well with other discrete data generative models, *e.g.* discrete state diffusion models [110], [111]. Architecture details, also for 2D toy data experiments, in B.2.2.

4.3 Experiments

4.3.1 Image Generation

We use the ImageNet [112] dataset and tackle both class-conditional (at varying resolutions 64×64 and 128×128) and unconditional generation. To measure sample quality, we follow the literature and use Fréchet Inception Distance (FID) [113] (lower is better). We also report the number of neural function evaluations (NFE).

In the class-conditional setting, the DisCo-Diff's denoiser is initialized using pre-trained ImageNet models, except for the new components: the cross-attention layers between discrete latents and images, and the encoder. We utilize the pre-trained U-Net in EDM [27] for

ImageNet-64, and for ImageNet-128, we implement the U-ViT in VDM++ [91]. We also adhere to their respective noise schedules and loss weightings during the training process. We use Heun's second-order method as ODE sampler, and a 12-layer Transformer as the auto-regressive model. We set the latent dimension to $m = 10$ and the codebook size to $k = 100$ in DisCo-Diff.

Results. See Table 4.1, Table 4.2. **(1) DisCo-Diff achieves the new state-of-the-art on class-conditioned ImageNet-64/ImageNet-128 when using ODE sampler.** Specifically, DisCo-Diff reduces the previous state-of-the-art FID score from 2.36 to 1.65 on ImageNet-64, and from 2.29 to 2.08 on ImageNet-128. This aligns with our analysis (4.2.2) that DisCo-Diff yields straighter ODE trajectories. **(2) DisCo-Diff outperforms all baselines in the unconditional setting, or when using stochastic sampler.** DisCo-Diff also surpasses the previous best method (SCDM [70]) in the unconditional setting, even though their method relies on pre-trained MoCo features. In addition, DisCo-Diff sets the new record ImageNet-64 FID of 1.22 when using Restart sampler [29]. Note that the competitive method RIN [103] employs a novel architecture distinct from conventional U-Nets/U-ViTs. **(3) Discrete latents capture variability complementary to class semantics.** Figure 4.2 (b) illustrates that samples sharing the same discrete latent exhibit similar characteristics, and there are noticeable distinctions for different discrete latents under the same class. It suggests that the discrete latents capture variations that are useful in simplifying the diffusion process defined in Euclidean space beyond class labels, underpinning the improvements of DisCo-Diff over the pre-trained class-conditioned DMs. **(4) Discrete latents boost the performance on PFGM++.** When applied to another ODE-based generative model PFGM++ [82], DisCo-PFGM++ also improves over the baseline version (see Table 4.1). More samples in B.2.4.

Ablations and Analyses. Table 4.3 shows that employing moderate classifier-free guidance with respect to the discrete latents (scale cfg=1) enhances the FID score (studied using ODE sampler), implying that the discrete latents effectively learn modes similar to the role of class labels and text. We further substituted the discrete latents with 1000-dim.

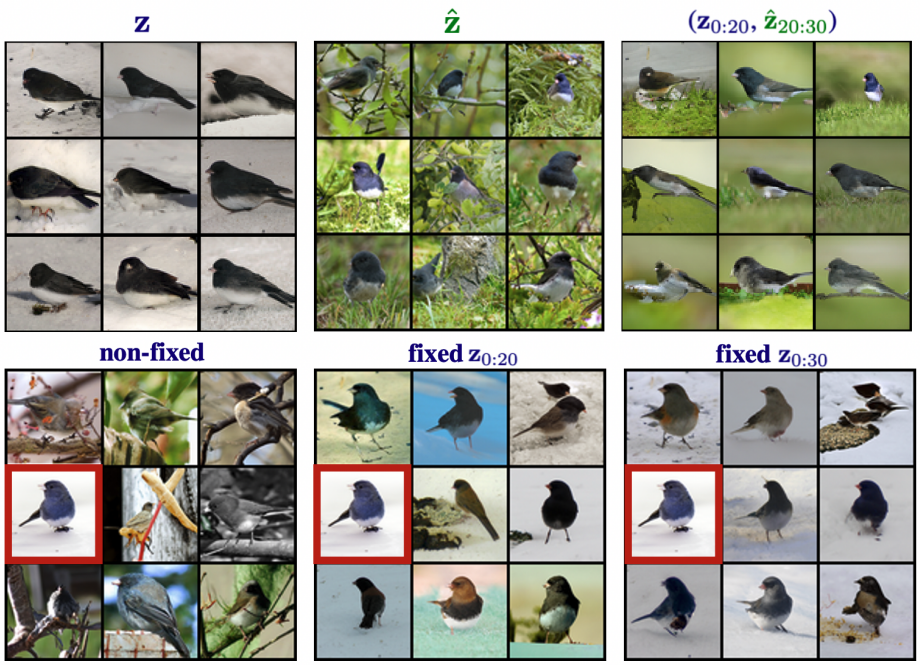


Figure 4.6: *Top:* Images created from two 30-dim discrete latents $\mathbf{z}$ and $\hat{\mathbf{z}}$, with the far-right column combining their sub-coordinates. *Bottom:* Variations in images by fixing portions of $\mathbf{z}$ (originating from the red-boxed image). We see that lower-resolution latents affect layout / shape; high-resolution latents alter color / texture.

continuous latents (1000 to offer capacity at least as high as with the m=10 and k=100 discrete latents), using Kullback-Leibler divergence-based (KLD) regularization as in VAEs to control the information retained. For fair comparison, we trained a DiT-based DM [114] on the continuous latents using the same Transformer architecture as in DisCo-Diff's auto-regressive model. Table 4.3 shows that with a low KLD weight (0.1), the continuous latents are under-regularized, challenging the DiT in modeling the complex encoding distribution and leading to a significant gap between oracle FID (latents predicted from training images) and generative FID (latents sampled from second-stage latent generative models). Conversely, a higher KLD weight (1) causes encoder collapse, and the continuous latents are not used (no latent (EDM), oracle latents and generative latents all produce same FIDs). In contrast, DisCo-Diff's generative FID shows only a minor degradation compared to the oracle FID, indicating the ease of modeling the discrete prior with a simple Transformer.

The DM training objective (2.7) has most variability at large diffusion times due to the multimodal posterior of clean data given noisy inputs [23]. Conditioning information can

(a) (b)

Figure 4.7: *Left:* Loss versus time. *Right:* Impact of discrete latent switching during the sampling process. The numbers represent the percentage of the total sampling steps. The blue/green arrows mean the sampling steps that utilize the discrete latent associated with the leftmost/rightmost grid in the figure.

reduce this ambiguity. For instance, [115] show that text conditioning primarily influences the denoiser at larger times. Figure 4.7 (a) shows that the learned discrete latents behave similarly to text conditioning, significantly lowering the training loss at higher time steps. Complementarily, Figure 4.7 (b) indicates that switching discrete latents towards the end of sampling barely affects the samples, implying they are not used at smaller times t.

In DisCo-Diff, the sampling time of the auto-regressive model is negligible compared to the DM's. For instance, for generating 32 images on ImageNet-128, the auto-regressive models requires only 0.44 seconds, while DisCo-Diff's DM component takes 78 seconds for

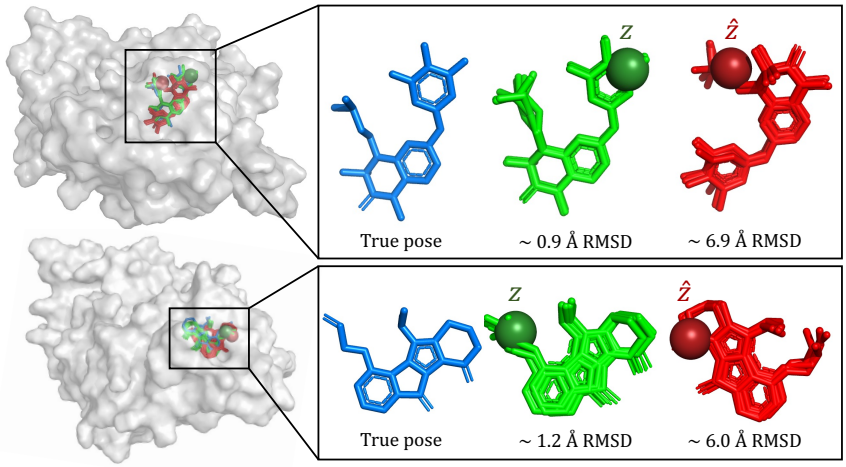


Figure 4.8: Examples of alternative docking poses modeled when conditioning on different discrete latents, the "correct" z (i.e. same as the encoder) and an incorrect $\hat{z}$. The DM maps them to two distinct sets of plausible orientations with which the ligand could fit in the pocket. Notably, the correct latent corresponds to poses within 2Å of the ground truth. The colored beads are set on the atoms corresponding to the first latent variable.

114 NFE, with an average of 0.68 second/NFE, all on a single NVIDIA A100 GPU.

Group Hierarchical DisCo-Diff. We evaluate the group hierarchical DisCo-Diff (Section 4.2.3), feeding three separate 10-dim. discrete latents into the U-Net at each level of resolution. Figure 4.6 shows that latents for lower-resolution features mainly govern overall shape and layout, while latents for higher-resolution control color and texture. For example, in the bottom figure, when gradually fixing groups in order, the images first converge in shape and then in color.

Discrete Latent Variable Classifier-Free Guidance Classifier-free guidance [84] (cfg) is a mode-seeking technique commonly used in diffusion literature, such as class-conditioned genreation [114] or text-to-image generation [4]. It generally guides the sampling trajectories toward higher-density regions. We can similarly apply classifier-free guidance in the DisCo-Diff, where we treat the discrete latent as conditional inputs. We follow the convention in [2], and the classifier-free guidance at time step t is as follows:

$$\tilde{\boldsymbol{D}}_\theta(\mathbf{x}, \sigma(t), \mathbf{z}) = w\boldsymbol{D}_\theta(\mathbf{x}, \sigma(t), \mathbf{z}) + (1-w)\boldsymbol{D}_\theta(\mathbf{x}, \sigma(t), \emptyset)$$

where $\boldsymbol{D}_\theta(\mathbf{x}, \sigma(t), \mathbf{z})/\boldsymbol{D}_\theta(\mathbf{x}, \sigma(t), \emptyset)$ is the conditional/unconditional models, sharing parameters. We drop the discrete latent with probability 0.1 during training, to train the unconditional model $\boldsymbol{D}_\theta(\mathbf{x}, \sigma(t), \emptyset)$. A mild w would usually lead to improvement in sample diversity [114]. Table 4.3 demonstrates that using a moderate guidance scale w=1 (we use $w = 1$ and cfg=1 interchangeably in the paper) improves the FID score, suggesting that the learned discrete latent in the DisCo-Diff framework has strong indications of mode of data distribution. We further explore varying the guidance scale on ImageNet-128. As shown in Figure 4.9, increasing the classifier-free guidance scale w would strengthen the effect of guidance.

(a) cfg=0 (b) cfg=1 (c) cfg=4 (d) cfg=8

Figure 4.9: Generated samples in DisCo-Diff with a cfg scale ranging from 0 to 8, under the class label “malamute" on ImageNet-128.

4.3.2 Molecular Docking

We test DisCo-Diff also on molecular docking, a fundamental task in drug discovery that consists of generating the 3D structure with which a small molecule will bind to a protein. We build on top of DiffDock [107], a DM that recently achieved state-of-the-art performance, integrating discrete latent variables (see Sec. 4.2 and App. B.2.2 for details). For computational reasons, we use the reduced DiffDock's architecture (referred to as DiffDock-S) from [116], which, although less accurate, is much faster for training and inference. For training and evaluation, we follow the standard from [117] using the PDBBind dataset [118] (see App. B.2.2).

Results. Table 4.4 reports performance of our (*DisCo-DiffDock-S*) and relevant baseline methods. We see that also in this domain discrete latents provide improvements, with the success rate on the full dataset increasing from 32.9% to 35.4% and from 13.9% to 18.5% when considering only test complexes with unseen proteins. This improvement is particularly strong on the harder component of the test set, where the baseline model is, likely, highly uncertain. This supports the intuition that DisCo-Diff boosts performance by more appropriately modeling discrete and continuous variations in the data. In Fig. 4.8, we visualize two examples from the test set which highlight how the model learns to associate distinct sets of poses with different latents, decomposing the multimodal components of the pose distribution from the continuous variations that each pose can have.

Table 4.4: Molecular docking performance on PDBBind. For each method, we report the percentage of top-1 predictions within 2Å of the ground truth for the *full* test set and the subset restricted to *unseen* proteins. Runtime in seconds (* refers to run on CPU).

	Full	Unseen	Runtime
GNINA [119]	22.9	14.0	127
SMINA [120]	18.7	14.0	126*
GLIDE [121]	21.8	19.6	1405*
EquiBind [117]	5.5	0.7	0.04
TankBind [122]	20.4	6.3	0.7
DiffDock-S [116]	32.9	13.9	8.1
DisCo-DiffDock-S (*ours*)	35.4	18.5	9.1
DiffDock [107]	38.2	20.8	40

4.4 Related Works

Our work builds on DMs [10], [24], [27], [123], which have been widely used not only for image generation [2]–[4], [90], [101], [124]–[126], but also for video [92], [127]–[129], 3D [130]–[135] and 4D [136]–[139] synthesis, as well as in various other domains, including, for instance, molecular docking and protein design [107], [140]–[142].

In the DM literature, latent variables have been most popular as part of latent diffusion models, where a DM is trained in a compressed, usually continuous, latent space [4], [143]. In contrast, DisCo-Diff leverages *discrete* latent variables and uses them to augment a DM. The first models using discrete latent variables for high-dimensional generative modeling tasks include Boltzmann machines [144], [145] and early discrete variational autoencoders [87], [146], [147]. More recently, a variety of works encode images into large 2D spatial grids of discrete tokens with vector quantization or similar techniques [73]–[78], [86]. As discussed, these models typically require a very large number of tokens and rely on large codebooks, which makes modeling their distribution challenging. DisCo-Diff, in contrast, leverages only few discrete latents with small codebooks that act in harmony with the additional continuous DM.

There are previous related works that also condition DMs on auxiliary encodings. [97] augment DMs with non-spatial latent variables, but their latents are continuous and high-dimensional, which makes training their latent DM more challenging. This is precisely what

we avoid by instead using low-dimensional and discrete latents. Moreover, they focus on semantic face image manipulation, not high-quality synthesis for challenging, diverse datasets.

[72] use the representations of a pre-trained CLIP image encoder [93] for conditioning a DM and learn another DM over the CLIP embeddings for sampling. Similarly, [70] and [71] use clustered MoCo-based [94] and clustered DINO-based [95] features, respectively, for conditioning. Hence, these three approaches are strictly limited to image synthesis, where such encoders, pre-trained on large-scale datasets, are available. In contrast, we purposefully avoid the use of pre-trained networks and learn the discrete latents jointly with the DM, making our framework universally applicable. Another related work is InfoDiffusion [98], which also conditions DMs on discrete latent variables. However, contrary to DisCo-Diff, this work focuses on learning disentangled representations, similar to β-VAEs [148], primarily for low-resolution face synthesis. It uses a mutual information-based objective and does not focus on diverse and high-quality synthesis of complex data such as ImageNet.

In contrast to the above works, we show how discrete latent variables boost generative performance itself and we significantly outperform these works in complex and diverse high-quality synthesis. Furthermore, we motivate DisCo-Diff fundamentally, with reduced ODE curvature and model complexity, providing a new and complementary perspective.

In the molecular docking literature, since DiffDock [107] introduced the use of diffusion models in the task, a number of works have proposed different modifications to its framework. In particular, some [149]–[151] have proposed to separate the blind docking task between pocket identification (i.e. identifying the region of the protein where the small molecule would bind) and pose prediction (i.e. predicting the specific pose with which the ligand would bind to the protein), as previously done in many traditional approaches [152]. One could see this as hand-crafting a (roughly discrete) latent variable in the pocket and using it to decompose the task. By allowing the encoder to learn arbitrary discrete latents through its interaction with the denoiser, DisCo-Diff largely includes the above-mentioned strategy as a particular case.

4.5 Conclusion

We have proposed ***Dis**crete-**Co**ntinuous Latent Variable **Diff**usion Models (DisCo-Diff)*, a novel and universal framework for combining discrete latent variables with continuous DMs. The approach significantly boosts performance by simplifying the DM's denoising task through the help of auxiliary discrete latent variables, while introducing negligible overhead. Extensive experiments and analyses demonstrate the unique benefits of global discrete latent variables that are learnt end-to-end with the denoiser. DisCo-Diff does not rely on any pre-trained encoder networks. As such, we validated our method not only on image synthesis, but also for molecular docking, demonstrating its universality. Future work includes applying DisCo-Diff at larger scale, for instance in text-to-image models or video generation.

Part II

Improved Sampling Techniques for Diffusion Models

In Part II, we address the challenges associated with the slow sampling speeds of pre-trained diffusion models. These challenges are primarily due to the need to solve differential equations and the repetitive generation of mini-batch samples for each user prompt. We focus on accelerating the sampling process by introducing a significantly larger amount of noise than typically used (Chapter 5), and avoiding repetitive samples by applying mutually repulsive forces among the samples (Chapter 6). The techniques presented are training-free and can be seamlessly integrated with any pre-trained diffusion models.

Chapter 5

Accelerating the Sampling Process by Optimized Noise Usage

The slow sampling processes hinder the real-world deployment of diffusion models. Previous efforts in acceleration of diffusion models frequently necessitate balancing speed and quality: ODE-based samplers are fast but plateau in performance, while SDE-based samplers deliver higher sample quality at the cost of increased sampling time. In this chapter, we attribute this difference to sampling errors: ODE-samplers involve smaller discretization errors while stochasticity in SDE contracts accumulated errors. Based on these findings, we propose a novel sampling algorithm called *Restart* in order to balance discretization errors and contraction better. The sampling method alternates between adding substantial noise in additional forward steps and strictly following a backward ODE. Empirically, the Restart sampler surpasses previous SDE and ODE samplers in both speed and accuracy. Restart not only outperforms the previous best SDE results but also accelerates the sampling speed by 10-fold / 2-fold on CIFAR-10 / ImageNet 64×64. In addition, it attains significantly better sample quality than ODE samplers within comparable sampling times. Moreover, Restart better balances text-image alignment/visual quality versus diversity than previous samplers in the large-scale text-to-image Stable Diffusion model.

This chapter was previously published as [29].

5.1 Introduction

Diffusion models involve iterative backward processes that gradually transform a simple distribution (*e.g.*, Gaussian in diffusion models) into a complex data distribution by solving differential equations. The associated vector fields (or drifts) driving the evolution of the differential equations are predicted by neural networks. The resulting sample quality can be often improved by enhanced simulation techniques but at the cost of longer sampling times.

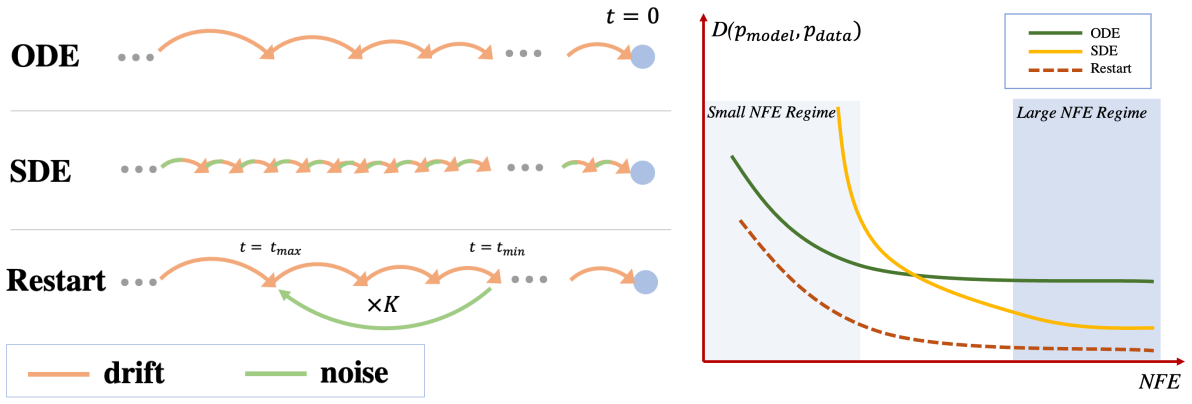
Prior samplers for simulating these backward processes can be categorized into two groups: ODE-samplers whose evolution beyond the initial randomization is deterministic, and SDE-samplers where the generation trajectories are stochastic. Several works [27], [28], [37] show that these samplers demonstrate their advantages in different regimes, as depicted in Figure 5.1b. ODE solvers [25]–[27] result in smaller discretization errors, allowing for decent sample quality even with larger step sizes (*i.e.*, fewer number of function evaluations (NFE)). However, their generation quality plateaus rapidly. In contrast, SDE achieves better quality in the large NFE regime, albeit at the expense of increased sampling time. To better understand these differences, we theoretically analyze SDE performance: the stochasticity in SDE contracts accumulated error, which consists of both the discretization error along the trajectories as well as the approximation error of the learned neural network relative to the ground truth drift (*e.g.*, score function in diffusion model [37]). The approximation error dominates when NFE is large (small discretization steps), explaining the SDE advantage in this regime. Intuitively, the stochastic nature of SDE helps "forget" accumulated errors from previous time steps.

Inspired by these findings, we propose a novel sampling algorithm called *Restart*, which combines the advantages of ODE and SDE. As illustrated in Figure 5.1a, the Restart sampling algorithm involves K repetitions of two subroutines in a pre-defined time interval: a *Restart forward process* that adds a substantial amount of noise, akin to "restarting" the original backward process, and a *Restart backward process* that runs the backward ODE. The Restart algorithm separates the stochasticity from the drifts, and the amount of added noise in the Restart forward process is significantly larger than the small single-step noise interleaving with drifts in previous SDEs such as [27], [37], thus amplifying the contraction effect on

accumulated errors. By repeating the forward-backward cycle K times, the contraction effect introduced in each Restart iteration is further strengthened. The deterministic backward processes allow Restart to reduce discretization errors, thereby enabling step sizes comparable to ODE. To maximize the contraction effects in practice, we typically position the Restart interval towards the end of the simulation, where the accumulated error is larger. Additionally, we apply multiple Restart intervals to further reduce the initial errors in more challenging tasks.

Experimentally, Restart consistently surpasses previous ODE and SDE solvers in both quality and speed over a range of NFEs, datasets, and pre-trained models. Specifically, Restart accelerates the previous best-performing SDEs by 10× fewer steps for the same FID score on CIFAR-10 using VP [37] (2× fewer steps on ImageNet 64×64 with EDM [27]), and outperforms fast ODE solvers (*e.g.*, DPM-solver [26]) even in the small NFE regime. When integrated into previous state-of-the-art pre-trained models, Restart further improves performance, achieving FID scores of 1.88 on unconditional CIFAR-10 with PFGM++ [31], and 1.36 on class-conditional ImageNet 64×64 with EDM. To the best of our knowledge, these are the best FID scores obtained on commonly used UNet architectures for diffusion models without additional training. We also apply Restart to the practical application of text-to-image Stable Diffusion model [153] pre-trained on LAION 512×512. Restart more effectively balances text-image alignment/visual quality (measured by CLIP/Aesthetic scores) and diversity (measured by FID score) with a varying classifier-free guidance strength, compared to previous samplers.

Our contributions can be summarized as follows: **(1)** We investigate ODE and SDE solvers and theoretically demonstrate the contraction effect of stochasticity via an upper bound on the Wasserstein distance between generated and data distributions (Section 5.2); **(2)** We introduce the Restart sampling, which better harnesses the contraction effect of stochasticity while allowing for fast sampling. The sampler results in a smaller Wasserstein upper bound (Section 5.3); **(3)** Our experiments are consistent with the theoretical bounds and highlight Restart's superior performance compared to previous samplers on standard benchmarks in terms of both quality and speed. Additionally, Restart improves the trade-off between key metrics on the Stable Diffusion model (Section 5.4).



(a) (b)

Figure 5.1: **(a)** Illustration of the implementation of drift and noise terms in ODE, SDE, and Restart. **(b)** Sample quality versus number of function evaluations (NFE) for different approaches. ODE (Green) provides fast speeds but attains only mediocre quality, even with a large NFE. SDE (Yellow) obtains good sample quality but necessitates substantial sampling time. In contrast to ODE and SDE, which have their own winning regions, Restart (Red) achieves the best quality across all NFEs.

5.2 Explaining SDE and ODE Performance Regimes

To sample from the aforementioned generative models, a prevalent approach employs general-purpose numerical solvers to simulate the corresponding differential equations. This includes Euler and Heun's 2nd method [154] for ODEs (e.g., Equation 2.5), and Euler-Maruyama for SDEs (e.g., Equation 2.4). Sampling algorithms typically balance two critical metrics: (1) the quality and diversity of generated samples, often assessed via the Fréchet Inception Distance (FID) between generated distribution and data distribution [41] (lower is better), and (2) the sampling time, measured by the number of function evaluations (NFE). Generally, as the NFE decreases, the FID score tends to deteriorate across all samplers. This is attributed to the increased discretization error caused by using a larger step size in numerical solvers.

However, as illustrated in Figure 5.1b and observed in previous works on diffusion models [25], [27], [37], the typical pattern of the quality vs time curves behaves differently between the two groups of samplers, ODE and SDE. When employing standard numerical solvers, ODE samplers attain a decent quality with limited NFEs, whereas SDE samplers struggle in the same small NFE regime. However, the performance of ODE samplers quickly reaches a plateau and fails to improve with an increase in NFE, whereas SDE samplers can

achieve noticeably better sample quality in the high NFE regime. This dilemma raises an intriguing question: *Why do ODE samplers outperform SDE samplers in the small NFE regime, yet fall short in the large NFE regime?*

The first part of the question is relatively straightforward to address: given the same order of numerical solvers, simulation of ODE has significantly smaller discretization error compared to the SDE. For example, the first-order Euler method for ODE results in a local error of $O(\delta^2)$, whereas the first-order Euler-Maruyama method for SDEs yields a local error of $O(\delta^{\frac{3}{2}})$ (see *e.g.*, Theorem 1 of [155]), where δ denotes the step size. As $O(\delta^{\frac{3}{2}}) \gg O(\delta^2)$, ODE simulations exhibit lower sampling errors than SDEs, likely causing the better sample quality with larger step sizes in the small NFE regime.

In the large NFE regime, the step size δ shrinks and discretization errors become less significant for both ODEs and SDEs. In this regime it is the *approximation error* — error arising from an inaccurate estimation of the ground-truth vector field by the neural network s_θ — starts to dominate the sampling error. We denote the discretized ODE and SDE using the learned field s_θ as ODE_θ and SDE_θ, respectively. In the following theorem, we evaluate the total errors from simulating ODE_θ and SDE_θ within the time interval $[t_{\min}, t_{\max}] \subset [0, T]$. This is done via an upper bound on the Wasserstein-1 distance between the generated and data distributions at time $t_{\min}$. We characterize the accumulated initial sampling errors up until $t_{\max}$ by total variation distances. Below we show that the inherent stochasticity of SDEs aids in contracting these initial errors at the cost of larger additional sampling error in $[t_{\min}, t_{\max}]$. Consequently, SDE results in a smaller upper bound as the step size δ nears 0 (pertaining to the high NFE regime).

Theorem 3 (Informal)**.** *Let t_{max} be the initial noise level and p_t denote the true distribution at noise level t. Let $p_t^{ODE_\theta}, p_t^{SDE_\theta}$ denote the distributions of simulating ODE_θ, SDE_θ respectively. Assume that $\forall t \in [t_{min}, t_{max}]$, $\|x_t\| < B/2$ for any $\mathbf{x}_t$ in the support of p_t, $p_t^{ODE_\theta}$ or $p_t^{SDE_\theta}$. Then*

$$W_1(p_{t_{min}}^{ODE_\theta}, p_{t_{min}}) \leq B \cdot TV\left(p_{t_{max}}^{ODE_\theta}, p_{t_{max}}\right) + O(\delta + \epsilon_{approx}) \cdot (t_{max} - t_{min})$$

$$\underbrace{W_1(p_{t_{min}}^{SDE_\theta}, p_{t_{min}})}_{\text{total error}} \leq \underbrace{\left(1 - \lambda e^{-U}\right) B \cdot TV(p_{t_{max}}^{SDE_\theta}, p_{t_{max}})}_{\text{upper bound on contracted error}} + \underbrace{O(\sqrt{\delta t_{max}} + \epsilon_{approx})\,(t_{max} - t_{min})}_{\text{upper bound on additional sampling error}}$$

In the above, $U = BL_1/t_{min} + L_1^2 t_{max}^2/t_{min}^2$, $\lambda < 1$ *is a contraction factor,* L_1 *and* ϵ_{approx} *are uniform bounds on* $\|t\mathbf{s}_\theta(\mathbf{x}_t, t)\|$ *and the approximation error* $\|t\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - t\mathbf{s}_\theta(\mathbf{x}, t)\|$ *for all* $\mathbf{x}_t, t$*, respectively.* $O()$ *hides polynomial dependency on various Lipschitz constants and dimension.*

We defer the formal version and proof of Theorem 3 to Appendix A.2.1. As shown in the theorem, the upper bound on the total error can be decomposed into upper bounds on the *contracted error* and *additional sampling error*. $TV(p_{t_{\max}}^{\mathrm{ODE}_\theta}, p_{t_{\max}})$ and $TV(p_{t_{\max}}^{\mathrm{SDE}_\theta}, p_{t_{\max}})$ correspond to the initial errors accumulated from both approximation and discretization errors during the simulation of the backward process, up until time $t_{\max}$. In the context of SDE, this accumulated error undergoes contraction by a factor of $1 - \lambda e^{-BL_1/t_{\min} - L_1^2 t_{\max}^2/t_{\min}^2}$ within $[t_{\min}, t_{\max}]$, due to the effect of adding noise. Essentially, the minor additive Gaussian noise in each step can drive the generated distribution and the true distribution towards each other, thereby neutralizing a portion of the initial accumulated error.

The other term related to additional sampling error includes the accumulation of discretization and approximation errors in $[t_{\min}, t_{\max}]$. Despite the fact that SDE incurs a higher discretization error than ODE ($O(\sqrt{\delta})$ versus $O(\delta)$), the contraction effect on the initial error is the dominant factor impacting the upper bound in the large NFE regime where δ is small. Consequently, the upper bound for SDE is significantly lower. This provides insight into why SDE outperforms ODE in the large NFE regime, where the influence of discretization errors diminishes and the contraction effect dominates. In light of the distinct advantages of SDE and ODE, it is natural to ask whether we can combine their strengths. Specifically, can we devise a sampling algorithm that maintains a comparable level of discretization error as ODE, while also benefiting from, or even amplifying, the contraction effects induced by the stochasticity of SDE? In the next section, we introduce a novel algorithm, termed *Restart*, designed to achieve these two goals simultaneously.

5.3 Harnessing Noise with Restart

In this section, we present the Restart sampling algorithm, which incorporates stochasticity during sampling while enabling fast generation. We introduce the algorithm in Sec 5.3.1,

followed by a theoretical analysis in Sec 5.3.2. Our analysis shows that Restart achieves a better Wasserstein upper bound compared to those of SDE and ODE in Theorem 3 due to greater contraction effects.

5.3.1 Restart Sampling

In the Restart algorithm, simulation performs a few repeated back-and-forth steps within a pre-defined time interval $[t_{\min}, t_{\max}] \subset [0, T]$, as depicted in Figure 5.1a. This interval is embedded into the simulation of the original backward ODE referred to as the *main backward process*, which runs from T to 0. In addition, we refer to the backward process within the Restart interval $[t_{\min}, t_{\max}]$ as the *Restart backward process*, to distinguish it from the main backward process.

Starting with samples at time $t_{\min}$, which are generated by following the main backward process, the Restart algorithm adds a large noise to transit the samples from $t_{\min}$ to $t_{\max}$ with the help of the forward process. The forward process does not require any evaluation of the neural network $s_\theta(x, t)$, as it is generally defined by an analytical perturbation kernel capable of transporting distributions from $t_{\min}$ to $t_{\max}$. For instance, in the case of diffusion models, the perturbation kernel is $\mathcal{N}(\mathbf{0}, (\sigma(t_{\max})^2 - \sigma(t_{\min})^2)\boldsymbol{I}_{d\times d})$. The added noise in this step induces a more significant contraction compared to the small, interleaved noise in SDE. The step acts as if partially restarting the main backward process by increasing the time. Following this step, Restart simulates the backward ODE from $t_{\max}$ back to $t_{\min}$ using the neural network predictions as in regular ODE. We repeat these forward-backward steps within $[t_{\min}, t_{\max}]$ interval K times in order to further derive the benefit from contraction. Specifically, the forward and backward processes in the i^{th} iteration ($i \in \{0, \dots, K-1\}$) proceed as follows:

$$\text{(Restart forward process)} \qquad \mathbf{x}^{i+1}_{t_{\max}} = \mathbf{x}^{i}_{t_{\min}} + \varepsilon_{t_{\min}\to t_{\max}} \tag{5.1}$$

$$\text{(Restart backward process)} \qquad \mathbf{x}^{i+1}_{t_{\min}} = \mathrm{ODE}_\theta(\mathbf{x}^{i+1}_{t_{\max}}, t_{\max} \to t_{\min}) \tag{5.2}$$

where the initial $\mathbf{x}^0_{t_{\min}}$ is obtained by simulating the ODE until $t_{\min}$: $\mathbf{x}^0_{t_{\min}} = \mathrm{ODE}_\theta(x_T, T \to t_{\min})$, and the noise $\varepsilon_{t_{\min}\to t_{\max}}$ is sampled from the corresponding perturbation kernel from

$t_{\min}$ to $t_{\max}$. The Restart algorithm not only adds substantial noise in the Restart forward process (Equation 5.1), but also separates the stochasticity from the ODE, leading to a greater contraction effect, which we will demonstrate theoretically in the next subsubsection. For example, we set $[t_{\min}, t_{\max}] = [0.05, 0.3]$ for the VP model [27] on CIFAR-10. Repetitive use of the forward noise effectively mitigates errors accumulated from the preceding simulation up until $t_{\max}$. Furthermore, the Restart algorithm does not suffer from large discretization errors as it is mainly built from following the ODE in the Restart backward process (Equation 5.2). The effect is that the Restart algorithm is able to reduce the total sampling errors even in the small NFE regime. Detailed pseudocode for the Restart sampling process can be found in Algorithm 9, Appendix B.3.1.

5.3.2 Theoretical Analysis

We provide a theoretical analysis of the Restart algorithm under the same setting as Theorem 3. In particular, we prove the following theorem, which shows that Restart achieves a much smaller contracted error in the Wasserstein upper bound than SDE (Theorem 3), thanks to the separation of the noise from the drift, as well as the large added noise in the Restart forward process (Equation 5.1). The repetition of the Restart cycle K times further leads to a enhanced reduction in the initial accumulated error. We denote the intermediate distribution in the i^{th} Restart iteration, following the discretized trajectories and the learned field s_θ, as $p^{\text{Restart}_\theta(i)}_{t\in[t_{\min},t_{\max}]}$.

Theorem 4 (Informal)**.** *Under the same setting of Theorem 3, assume* $K \leq \frac{C}{L_2(t_{max}-t_{min})}$ *for some universal constant* C*. Then*

$$\underbrace{W_1(p^{Restart_\theta(K)}_{t_{min}}, p_{t_{min}})}_{\textit{total error}} \leq \underbrace{B\cdot(1-\lambda)^K\, TV(p^{Restart_\theta(0)}_{t_{max}}, p_{t_{max}})}_{\textit{upper bound on contracted error}} + \underbrace{(K+1)\cdot O\left(\delta+\epsilon_{approx}\right)(t_{max}-t_{min})}_{\textit{upper bound on additional sampling error}}$$

where $\lambda < 1$ *is the same contraction factor as Theorem 3.* $O()$ *hides polynomial dependency on various Lipschitz constants, dimension.*

Proof sketch. To bound the total error, we introduce an auxiliary process $q^{\text{Restart}_\theta(i)}_{t\in[t_{\min},t_{\max}]}$, which initiates from true distribution $p_{t_{\max}}$ and performs the Restart iterations. This process differs from $p^{\text{Restart}_\theta(i)}_{t\in[t_{\min},t_{\max}]}$ only in its initial distribution at $t_{\max}$ ($p_{t_{\max}}$ versus $p^{\text{Restart}_\theta(0)}_{t_{\max}}$). We bound the total error by the following triangular inequality:

$$\underbrace{W_1(p^{\text{Restart}_\theta(K)}_{t_{\min}}, p_{t_{\min}})}_{\text{total error}} \le \underbrace{W_1(p^{\text{Restart}_\theta(K)}_{t_{\min}}, q^{\text{Restart}_\theta(K)}_{t_{\min}})}_{\text{contracted error}} + \underbrace{W_1(q^{\text{Restart}_\theta(K)}_{t_{\min}}, p_{t_{\min}})}_{\text{additional sampling error}}$$

To bound the contracted error, we construct a careful coupling process between two individual trajectories sampled from $p^{\text{Restart}_\theta(i)}_{t_{\min}}$ and $q^{\text{Restart}_\theta(i)}_{t_{\min}}, i = 0, \dots, K-1$. Before these two trajectories converge, the Gaussian noise added in each Restart iteration is chosen to maximize the probability of the two trajectories mapping to an identical point, thereby maximizing the mixing rate in TV. After converging, the two processes evolve under the same Gaussian noise, and will stay converged as their drifts are the same. Lastly, we convert the TV bound to W_1 bound by multiplying B. The bound on the additional sampling error echoes the ODE analysis in Theorem 3: since the noise-injection and ODE-simulation stages are separate, we do not incur the higher discretization error of SDE. □

We defer the formal version and proof of Theorem 4 to Appendix A.2.1. The first term in RHS bounds the contraction on the initial error at time $t_{\max}$ and the second term reflects the additional sampling error of ODE accumulated across repeated Restart iterations. Comparing the Wasserstein upper bound of SDE and ODE in Theorem 3, we make the following three observations: *(1)* Each Restart iteration has a smaller contraction factor $1-\lambda$ compared to the one in SDE, since Restart separates the large additive noise (Equation 5.1) from the ODE (Equation 5.2). *(2)* Restart backward process (Equation 5.2) has the same order of discretization error $O(\delta)$ as the ODE, compared to $O(\sqrt{\delta})$ in SDE. Hence, the Restart allows for small NFE due to ODE-level discretization error. *(3)* The contracted error further diminishes exponentially with the number of repetitions K though the additional error increases linearly with K. It suggests that there is a sweet spot of K that strikes a balance between reducing the initial error and increasing additional sampling error. Ideally, one should pick a larger K when the initial error at time $t_{\max}$ greatly outweighs the incurred error in the repetitive backward process from $t_{\max}$ to $t_{\min}$. We provide empirical evidences in

Sec 5.4.2.

While Theorem 3 and Theorem 4 compare the upper bounds on errors of different methods, we provide empirical validation in Section 5.4.1 by directly calculating these errors, showing that the Restart algorithm indeed yields a smaller total error due to its superior contraction effects. The main goal of Theorem 3 and Theorem 4 is to study how the already accumulated error changes using different samplers, and to understand their ability to self-correct the error by stochasticity. In essence, these theorems differentiate samplers based on their performance post-error accumulation. For example, by tracking the change of accumulated error, Theorem 1 sheds light on the distinct "winning regions" of ODE and SDE: ODE samplers have smaller discretization error and hence excel at the small NFE regime. In contrast, SDE performs better in large NFE regime where the discretization error is negligible and its capacity to contract accumulated errors comes to the fore.

5.3.3 Practical Considerations

The Restart algorithm offers several degrees of freedom, including the time interval $[t_{\min}, t_{\max}]$ and the number of restart iterations K. Here we provide a general recipe of parameter selection for practitioners, taking into account factors such as the complexity of the generative modeling tasks and the capacity of the network. Additionally, we discuss a stratified, multi-level Restart approach that further aids in reducing simulation errors along the whole trajectories for more challenging tasks.

Where to Restart? Theorem 4 shows that the Restart algorithm effectively reduces the accumulated error at time $t_{\max}$ by a contraction factor in the Wasserstein upper bound. These theoretical findings inspire us to position the Restart interval $[t_{\min}, t_{\max}]$ towards the end of the main backward process, where the accumulated error is more substantial. In addition, our empirical observations suggest that a larger time interval $t_{\max} - t_{\min}$ is more beneficial for weaker/smaller architectures or more challenging datasets. Even though a larger time interval increases the additional sampling error, the benefits of the contraction significantly outweigh the downside, consistent with our theoretical predictions. We leave the development of principled approaches for optimal time interval selection for future works.

Multi-level Restart For challenging tasks that yield significant approximation errors,

the backward trajectories may diverge substantially from the ground truth even at early stage. To prevent the ODE simulation from quickly deviating from the true trajectory, we propose implementing multiple Restart intervals in the backward process, alongside the interval placed towards the end. Empirically, we observe that a 1-level Restart is sufficient for CIFAR-10, while for more challenging datasets such as ImageNet [156], a multi-level Restart results in enhanced performance [156].

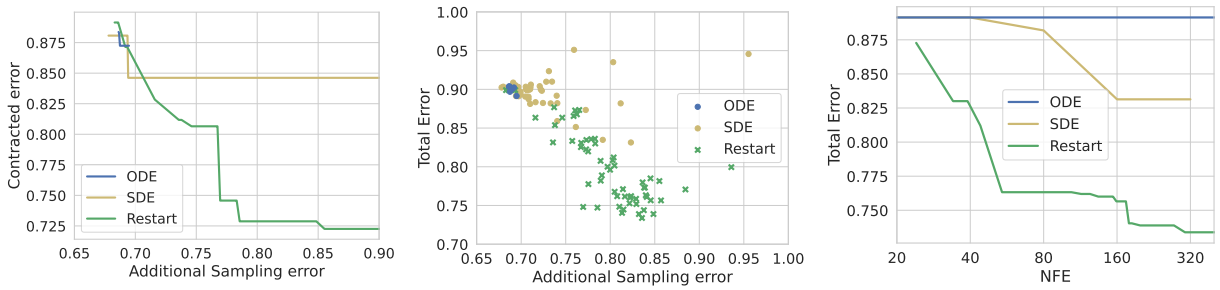
5.4 Experiments

In Sec 5.4.1, we first empirically verify the theoretical analysis relating to the Wasserstein upper bounds. We then evaluate the performance of different sampling algorithms on standard image generation benchmarks, including CIFAR-10 [157] and ImageNet 64×64 [156] in Sec 5.4.2. Lastly, we employ Restart on text-to-image generation, using Stable Diffusion model [153] pre-trained on LAION-5B [158] with resolution 512×512, in Sec 5.4.3.

5.4.1 Sampling Error versus Contracted Error

Our proposed Restart sampling algorithm demonstrates a higher contraction effect and smaller addition sampling error compared to SDE, according to Theorem 3 and Theorem 4. Although our theoretical analysis compares the upper bounds of the total, contracted and additional sampling errors, we further verify their relative values through a synthetic experiment.

Setup We construct a 20-dimensional dataset with 2000 points sampled from a Gaussian mixture, and train a four-layer MLP to approximate the score field $\nabla_{\mathbf{x}} \log p_t$. We implement the ODE, SDE, and Restart methods within a predefined time range of $[t_{\min}, t_{\max}] = [1.0, 1.5]$, where the process outside this range is conducted via the first-order ODE. To compute various error types, we define the distributions generated by three methods as outlined in the proof of Theorem 4 and directly gauge the errors at end of simulation $t = 0$ instead of $t = t_{\min}$: (1) the generated distribution as p_0^{Sampler}, where Sampler $\in \{\text{ODE}_\theta, \text{SDE}_\theta, \text{Restart}_\theta(K)\}$; (2) an auxiliary distribution q_0^{Sampler} initiating from true distribution $p_{t_{\max}}$ at time $t_{\max}$. The only difference between p_0^{Sampler} and q_0^{Sampler} is their initial distribution at $t_{\max}$ ($p_{t_{\max}}^{\text{ODE}_\theta}$ versus $p_{t_{\max}}$); and (3) the true data distribution p_0. In line with Theorem 4, we use Wasserstein-1



(a) (b) (c)

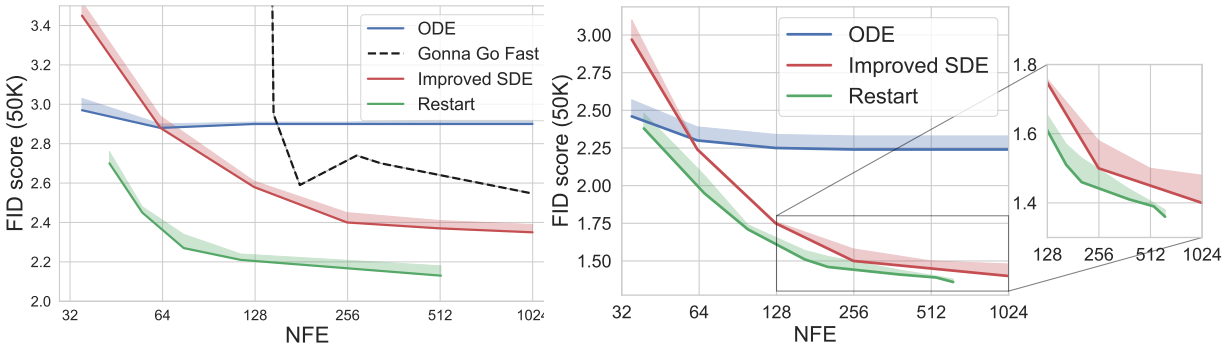
Figure 5.2: Additional sampling error versus **(a)** contracted error, where the Pareto frontier is plotted and **(b)** total error, where the scatter plot is provided. **(c)** Pareto frontier of NFE versus total error.

distance $W_1(p_0^{\text{Sampler}}, q_0^{\text{Sampler}})$ / $W_1(q_0^{\text{Sampler}}, p_0)$ to measure the contracted error / additional sampling error, respectively. Ultimately, the total error corresponds to $W_1(p_0^{\text{Sampler}}, p_0)$. Detailed information about dataset, metric and model can be found in the Appendix B.3.2.

Results In our experiment, we adjust the parameters for all three processes and calculate the total, contracted, and additional sampling errors across all parameter settings. Figure 5.2a depicts the Pareto frontier of additional sampling error versus contracted error. We can see that Restart consistently achieves lower contracted error for a given level of additional sampling error, compared to both the ODE and SDE methods, as predicted by theory. In Figure 5.2b, we observe that the Restart method obtains a smaller total error within the additional sampling error range of $[0.8, 0.85]$. During this range, Restart also displays a strictly reduced contracted error, as illustrated in Figure 5.2a. This aligns with our theoretical analysis, suggesting that the Restart method offers a smaller total error due to its enhanced contraction effects. From Figure 5.2c, Restart also strikes a better balance between efficiency and quality, as it achieves a lower total error at a given NFE.

5.4.2 Experiments on Standard Benchmarks

To evaluate the sample quality and inference speed, we report the FID score [41] (lower is better) on 50K samplers and the number of function evaluations (NFE). We borrow the pretrained VP/EDM/PFGM++ models on CIFAR-10 or ImageNet 64×64 from [27], [31]. We also use the EDM discretization scheme [27] (see Appendix B.3.1 for details) during



(a) CIFAR-10, VP (b) ImageNet 64×64, EDM

Figure 5.3: FID versus NFE on **(a)** unconditional generation on CIFAR-10 with VP; **(b)** class-conditional generation on ImageNet with EDM.

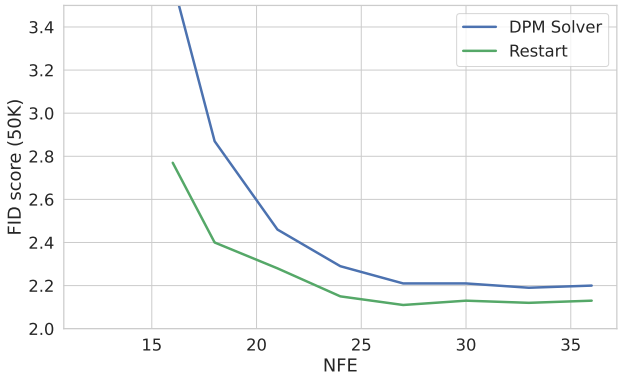


Figure 5.4: CIFAR-10, VP, in the low NFE regime. Restart consistently outperforms the DPM-solver with an NFE ranging from 16 to 36.

sampling.

For the proposed Restart sampler, the hyperparameters include the number of steps in the main/Restart backward processes, the number of Restart iteration K, as well as the time interval $[t_{\min}, t_{\max}]$. We pick the $t_{\min}$ and $t_{\max}$ from the list of time steps in EDM discretization scheme with a number of steps 18. For example, for CIFAR-10 (VP) with NFE=75, we choose $t_{\min}$=0.06, $t_{\max}$=0.30, K=10, where 0.30/0.06 is the $12^{\text{th}}/14^{\text{th}}$ time step in the EDM scheme. We also adopt EDM scheme for the Restart backward process in $[t_{\min}, t_{\max}]$. In addition, we apply the multi-level Restart strategy (Sec 5.3.3) to mitigate the error at early time steps for the more challenging ImageNet 64×64. We provide the detailed Restart configurations in Appendix B.3.2.

For SDE, we compare with the previously best-performing stochastic samplers proposed

by [27] (**Improved SDE**). We use their optimal hyperparameters for each dataset. We also report the FID scores of the adaptive SDE [28] (**Gonna Go Fast**) on CIFAR-10 (VP). Since the vanilla reverse-diffusion SDE [37] has a significantly higher FID score, we omit its results from the main charts and defer them to Appendix B.5.3. For ODE samplers, we compare with the Heun's 2^{nd} order method [154] (**Heun**), which arguably provides an excellent trade-off between discretization errors and NFE [27]. To ensure a fair comparison, we use Heun's method as the sampler in the main/Restart backward processes in Restart.

We report the FID score versus NFE in Figure 5.3a and Table 5.1 on CIFAR-10, and Figure 5.3b on ImageNet 64×64 with EDM. Our main findings are: **(1)** Restart outperforms other SDE or ODE samplers in balancing quality and speed, across datasets and models. As demonstrated in the figures, Restart achieves a 10-fold / 2-fold acceleration compared to the previous best SDE results on CIFAR-10 (VP) / ImageNet 64×64 (EDM) at the same FID score. In comparison to ODE sampler (Heun), Restart obtains a better FID score, with the gap increasing significantly with NFE. **(2)** For stronger models such as EDM and PFGM++, Restart further improve over the ODE baseline on CIFAR-10. In contrast, the Improved SDE negatively impacts the performance of EDM, as also observed in [27]. It suggests that Restart incorporates stochasticity more effectively. **(3)** Restart establishes new state-of-the-art FID scores for UNet architectures without additional training. In particular, Restart achieves FID scores of 1.36 on class-cond. ImageNet 64×64 with EDM, and 1.88 on uncond. CIFAR-10 with PFGM++.

To further validate that Restart can be applied in low NFE regime, we show that one can employ faster ODE solvers such as the **DPM-solver-3** [26] to further accelerate Restart. Figure 5.4 shows that the Restart consistently outperforms the DPM-solver with an NFE ranging from 16 to 36. This demonstrates Restart's capability to excel over ODE samplers, even in the small NFE regime. It also suggests that Restart can consistently improve other ODE samplers, not limited to the DDIM, Heun. Surprisingly, when paired with the DPM-solver, **Restart achieves an FID score of** 2.11 **on VP setting when NFE is 30, which is significantly lower than any previous numbers (even lower than the SDE sampler with an NFE greater than 1000 in [37]), and make VP model on par with the performance with more advanced models (such as EDM).**

We include detailed Restart configuration in Table B.6 in Appendix B.3.2.

Theorem 10 shows that each Restart iteration reduces the contracted errors while increasing the additional sampling errors in the backward process. In Figure 5.5, we explore the choice of the number of Restart iterations K on CIFAR-10. We find that FID score initially improves and later worsens with increasing iterations K, with a smaller turning point for stronger EDM model. This supports the theoretical analysis that sampling errors will eventually outweigh the contraction benefits as K increases, and EDM only permits fewer Restart iterations due to smaller accumulated errors. It also suggests that, as a rule of thumb, we should apply greater Restart strength (*e.g.*, larger K) for weaker or smaller architectures and vice versa.

Table 5.1: Uncond. CIFAR-10 with EDM and PFGM++

	NFE	FID
EDM-VP [27]		
ODE (Heun)	63	1.97
	35	1.97
Improved SDE	63	2.27
	35	2.45
Restart	43	**1.90**
PFGM++ [31]		
ODE (Heun)	63	1.91
	35	1.91
Restart	43	**1.88**

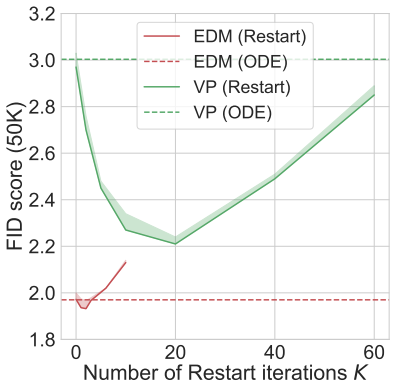
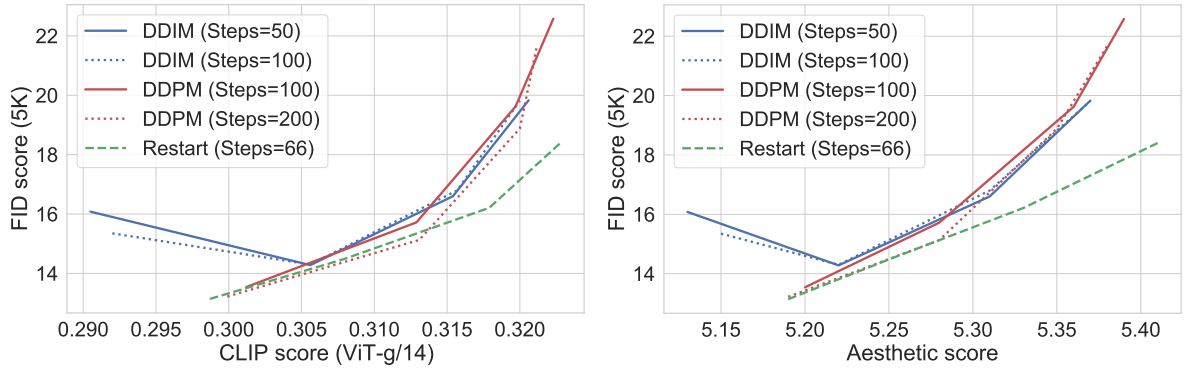


Figure 5.5: FID score with a varying number of Restart iterations K.

5.4.3 Experiments on Large-scale Text-to-Image Model

We further apply Restart to the text-to-image Stable Diffusion v1.5 [1] pre-trained on LAION-5B [158] at a resolution of 512 × 512. We employ the commonly used classifier-free guidance [159], [160] for sampling, wherein each sampling step entails two function evaluations – the conditional and unconditional predictions. Following [160], [161], we use the COCO [162] validation set for evaluation. We assess text-image alignment using the CLIP score [163] with the open-sourced ViT-g/14 [164], and measure diversity via the FID score. We also evaluate visual quality through the Aesthetic score, as rated by the LAION-Aesthetics Predictor V2 [165]. Following [166], we compute all evaluation metrics using 5K captions randomly sampled from the validation set and plot the trade-off curves between CLIP/Aesthetic scores

[1] https://huggingface.co/runwayml/stable-diffusion-v1-5



(a) FID versus CLIP score

(b) FID versus Aesthetic score

Figure 5.6: FID score versus **(a)** CLIP ViT-g/14 score and **(b)** Aesthetic score for text-to-image generation at 512×512 resolution, using Stable Diffusion v1.5 with a varying classifier-free guidance weight $w = 2, 3, 5, 8$.

and FID score, with the classifier-free guidance weight w in $\{2, 3, 5, 8\}$.

We compare with commonly used ODE sampler DDIM [25] and the stochastic sampler DDPM [167]. For Restart, we adopt the DDIM solver with 30 steps in the main backward process, and Heun in the Restart backward process, as we empirically find that Heun performs better than DDIM in the Restart. In addition, we select different sets of hyperparameters for each guidance weight. For instance, when $w = 8$, we use $[t_{\text{min}}, t_{\text{max}}]$=$[0.1, 2]$, K=2 and 10 steps in Restart backward process. We defer the detailed Restart configuration to Appendix B.3.2, and the results of Heun to Appendix B.3.3.

As illustrated in Figure 5.6a and Figure 5.6b, Restart achieves better FID scores in most cases, given the same CLIP/Aesthetic scores, using only 132 function evaluations (*i.e.*, 66 sampling steps). Remarkably, Restart achieves substantially lower FID scores than other samplers when CLIP/Aesthetic scores are high (*i.e.*, with larger w values). Conversely, Restart generally obtains a better text-image alignment/visual quality given the same FID. We also observe that DDPM generally obtains comparable performance with Restart in FID score when CLIP/Aesthetic scores are low, with Restart being more time-efficient. These findings suggest that Restart balances diversity (FID score) against text-image alignment (CLIP score) or visual quality (Aesthetic score) more effectively than previous samplers.

In Figure 5.7, we visualize the images generated by Restart, DDIM and DDPM with $w = 8$. Compared to DDIM, the Restart generates images with superior details (*e.g.*, the

(a) Restart (Steps=66) (b) DDIM (Steps=100) (c) DDPM (Steps=100)

Figure 5.7: Visualization of generated images with classifier-free guidance weight $w = 8$, using four text prompts ("A photo of an astronaut riding a horse on mars.", "A raccoon playing table tennis", "Intricate origami of a fox in a snowy forest" and "A transparent sculpture of a duck made out of glass") and the **same** random seeds.

rendition of duck legs by DDIM is less accurate) and visual quality. Compared to DDPM, Restart yields more photo-realistic images (*e.g.*, the astronaut). We provide extended of text-to-image generated samples in Appendix B.3.4.

5.5 Conclusion

In this chapter, we introduce the Restart sampling for generative processes involving differential equations, such as diffusion models and PFGMs. By interweaving a forward process that adds a significant amount of noise with a corresponding backward ODE, Restart harnesses and even enhances the individual advantages of both ODE and SDE. Theoretically, Restart provides greater contraction effects of stochasticity while maintaining ODE-level discretization error. Empirically, Restart achieves a superior balance between quality and time, and improves the text-image alignment/visual quality and diversity trade-off in the text-to-image Stable Diffusion models.

A current limitation of the Restart algorithm is the absence of a principled way for hyperparameters selection, including the number of iterations K and the time interval $[t_{\min}, t_{\max}]$. At present, we adjust these parameters based on the heuristic that weaker/smaller models, or more challenging tasks, necessitate a stronger Restart strength. In the future

direction, we anticipate developing a more principled approach to automating the selection of optimal hyperparameters for Restart based on the error analysis of models, in order to fully unleash the potential of the Restart framework.

Chapter 6

Non-I.I.D. Diverse Sampling with Diffusion Models

In practical deployment, most diffusion models generate four samples per user query (*e.g.,* Midjourney, Stable Diffusion). Therefore, the diversity of these mini-batch samples becomes an important issue alongside generation speed discussed in the previous chapter — users expect a diverse set of samples from which to choose. In this chapter, we tackle the question of how to improve diversity and sample efficiency by moving beyond the common assumption of independent samples. For this, we propose particle guidance, an extension of diffusion-based generative sampling where a joint-particle time-evolving potential enforces diversity. We analyze theoretically the joint distribution that particle guidance generates, its implications on the choice of potential, and the connections with methods in other disciplines. Empirically, we test the framework both in the setting of conditional image generation, where we are able to increase diversity without affecting quality, and molecular conformer generation, where we reduce the previous state-of-the-art median error.

This chapter was previously published in [33]. Gabriele Corso contributed significantly to the materials in this chapter. My contributions are conceiving the idea with Gabriele, conducting the image experiments, and helping with paper writing.

6.1 Introduction

Deep generative modeling has become pervasive in many computational tasks across computer vision, natural language processing, physical sciences, and beyond. In many applications, these models are used to take a number of representative samples of some distribution of interest like Van Gogh's style paintings or the 3D conformers of a small molecule. Although independent samples drawn from a distribution will perfectly represent it in the limit of infinite samples, this may not be the optimal strategy for a finite number. Therefore, while deep learning methods have so far largely focused on the task of taking independent identically distributed (I.I.D.) samples from some distribution, this paper examines how one can use deep generative models to take a finite number of samples that can better represent the distribution of interest.

In other fields where *finite-samples* approximations are critical, researchers have developed various techniques to tackle this challenge. In molecular simulations, several enhanced sampling methods, like metadynamics and replica exchange, have been proposed to sample diverse sets of low-energy structures and estimate free energies. In statistics, Stein Variational Gradient Descent (SVGD) is an iterative technique to match a distribution with a finite set of particles. However, these methods are not able to efficiently sample complex distributions like images.

Towards the goal of better *finite-samples* generative models, that combine the power of recent advances with sample efficiency, we propose a general framework for sampling sets of particles using diffusion models. This framework, which we call *particle guidance* (PG), is based on the use of a time-evolving potential to guide the inference process. We present two different strategies to instantiate this new framework: the first, *fixed potential particle guidance*, provides ready-to-use potentials that require no further training and have little inference overhead; the second, *learned potential particle guidance*, requires a training process but offers better control and theoretical guarantees.

The theoretical analysis of the framework leads us to two key results. On one hand, we obtain an expression for the joint marginal distribution of the sampled process when using any arbitrary guidance potential. On the other, we derive a simple objective one can use to

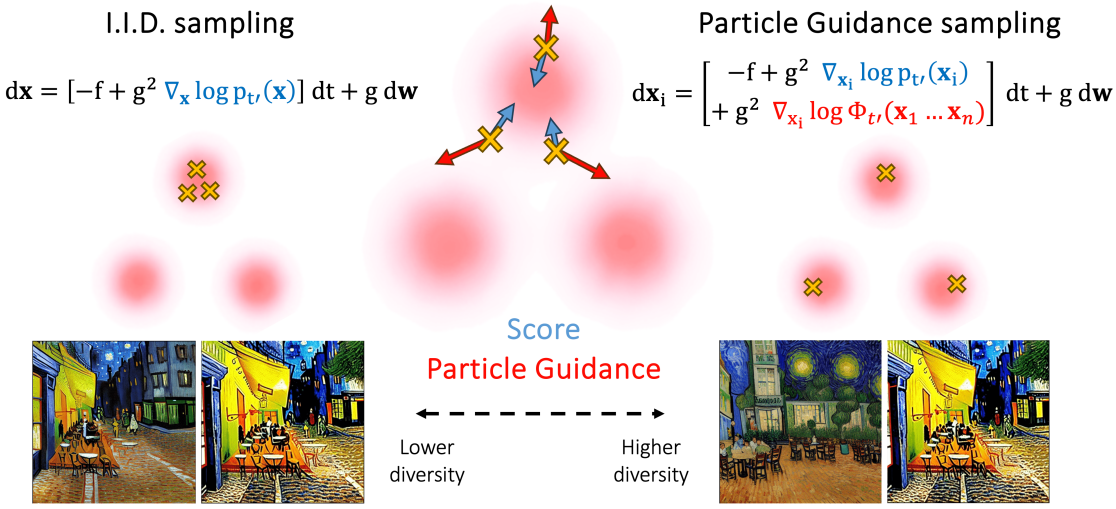


Figure 6.1: Comparison of I.I.D. and particle guidance sampling. The center figure represents each step, with the distribution in pink and the samples as yellow crosses, where particle guidance uses not only the score (in blue) but also the guidance from joint-potential (red), leading it to discover different modes (right-hand samples vs those on the left). At the bottom, Van Gogh cafe images samples generated with Stable Diffusion with and without particle guidance. A more detailed discussion on the suboptimality of I.I.D. sampling is presented in Appendix B.4.1.

train a model to learn a time-evolving potential that exactly samples from a joint distribution of interest. We show this provides optimal joint distribution given some diversity constraint and it can be adapted to the addition of further constraints such as the preservation of marginal distributions. Further, we also demonstrate the relations of particle guidance to techniques for non-I.I.D. sampling developed in other fields and natural processes and discuss its advantages.

Empirically, we demonstrate the effectiveness of the method in both synthetic experiments and two of the most successful applications of diffusion models: text-to-image generation and molecular conformer generation. In the former, we show that particle guidance can improve the diversity of the samples generated with Stable Diffusion [153] while maintaining a quality comparable to that of I.I.D. sampling. For molecular conformer generation, applied to the state-of-the-art method Torsional Diffusion [168], particle guidance is able to simultaneously improve precision and coverage, reducing their median error by respectively 19% and 8%. In all settings, we also study the critical effect that different potentials can have on the diversity and sample quality.

6.2 Promoting Sample Diversity with Particle Guidance

Our goal is to define a sampling process that promotes the diversity of a finite number of samples while retaining the advantages and flexibility that characterize diffusion models. Let $p(\mathbf{x})$ be some probability distribution of interest and $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ be the score that we have learned to reverse the diffusion process $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$. Similarly to how classifier guidance is applied, we modify the reverse diffusion process by adding the gradient of a potential. However, we are now sampling together a whole set of particles $\mathbf{x}_1$, ..., $\mathbf{x}_n$, and the potential $\log \Phi_t$ is not only a function of the current point but a permutation invariant function of the whole set:

$$\mathrm{d}\mathbf{x}_i = \Big[-\mathbf{f}(\mathbf{x}_i, t') + g^2(t') \Big(\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i) + \nabla_{\mathbf{x}_i} \log \Phi_{t'}(\mathbf{x}_1, ..., \mathbf{x}_n) \Big) \Big] \mathrm{d}t + g(t')\mathrm{d}\mathbf{w}. \quad (6.1)$$

where the points are initially sampled I.I.D. from a prior distribution p_T. We call this idea *particle guidance* (PG). This framework allows one to impose different properties, such as diversity, on the set of particles being sampled without the need to retrain a new score model operating directly on the space of sets.

We will present and study two different instantiations of this framework:

1. **Fixed Potential PG** where the time-evolving joint potential is handcrafted, leading to very efficient sampling of diverse sets without the need for any additional training. We present this instantiation in Section 6.4 and show its effectiveness on critical real-world applications of diffusion models in Section 6.4.2.

2. **Learned Potential PG** where we learn the time-evolving joint potential to provably optimal joint distributions. Further, this enables direct control of important properties such as the preservation of marginal distributions. We present this instantiation in Section 6.5.

6.3 Connections with Existing Methods

As discussed in the introduction, other fields have developed methods to improve the tradeoff between sampling cost and coverage of the distribution of interest. In this section, we will briefly introduce four methods (coupled replicas, metadynamics, SVGD and electrostatics) and draw connections with *particle guidance*.

6.3.1 Coupled Replicas and Metadynamics

In many domains linked to biochemistry and material science, researchers study the properties of the physical systems by collecting several samples from their Boltzmann distributions using molecular dynamics or other enhanced sampling methods. Motivated by the significant cost that sampling each individual structure requires, researchers have developed a range of techniques to improve sample efficiency and speed by, for example, reducing the correlation of subsequent samples from slow-mixing Markov chains. The most popular of these techniques are parallel sampling with coupled replicas and sequential sampling with metadynamics.

As the name suggests, replica methods involve directly taking n samples of a system with the different sampling processes, replicas, occurring in parallel. In particular, coupled replica methods [169], [170] create a dependency between the replicas by adding, like *particle guidance*, an extra potential Φ to the energy function to enforce diversity or match experimental observables. This results in energy-based sampling procedures that target:

$$\tilde{p}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n) \prod_{i=1}^{n} p(\mathbf{x}_i).$$

Metadynamics [171], [172] was also developed to more efficiently sample the Boltzmann distribution of a given system. Unlike replica methods and our approach, metadynamics is a sequential sampling technique where new samples are taken based on previously taken ones to ensure diversity, typically across certain collective variables of interest $s(\mathbf{x})$. In its original

formulation, the Hamiltonian at the k^{th} sample is augmented with a potential as:

$$\tilde{H}_k = H - \omega \sum_{j<k} \exp\left(- \frac{\|s(\mathbf{x}) - s(\mathbf{x}_j^0)\|^2}{2\sigma^2} \right)$$

where H is the original Hamiltonian, $\mathbf{x}_j^0$ are the previously sampled elements and ω and σ parameters set a priori. Once we take the gradient and perform Langevin dynamics to sample, we obtain dynamics that, with the exception of the fixed Hamiltonian, resemble those of *particle guidance* in Eq. 6.4 where

$$\nabla_{\mathbf{x}_i} \log \Phi_t(\mathbf{x}_1, \cdots, \mathbf{x}_n) \leftarrow \nabla_{\mathbf{x}_i} \omega \sum_{j<i} \exp\left(- \frac{\|s(\mathbf{x}_i) - s(\mathbf{x}_j^0)\|^2}{2\sigma^2} \right).$$

Although they differ in their parallel or sequential approach, both coupled replicas and metadynamics can be broadly classified as energy-based generative models. As seen here, energy-based models offer a simple way of controlling the joint distribution one converges to by simply adding a potential to the energy function. On the other hand, however, the methods typically employ an MCMC sampling procedure, which lacks the critical *finite-time sampling* property of diffusion models and significantly struggles to cover complex probability distributions such as those of larger molecules and biomolecular complexes. Additionally, the MCMC typically necessitates a substantial number of steps, generally proportional to a polynomial of the data dimension [173]. With particle guidance, we instead aim to achieve both properties (controllable diversity and finite time sampling) at the same time. We can simulate the associated SDE/ODE with a total number of steps that is independent of the data dimension.

6.3.2 Stein Variational Gradient Descent

Stein Variational Gradient Descent (SVGD) [174] is a well-established method in the variational inference community to iteratively transport a set of particles to match a target

distribution. Given a set of initial particles $\{\mathbf{x}_1^0 \dots \mathbf{x}_n^0\}$, it updates them at every iteration as:

$$\mathbf{x}_i^{\ell-1} \leftarrow \mathbf{x}_i^{\ell} + \epsilon_\ell \psi(\mathbf{x}_i^{\ell}) \quad \text{where} \quad \psi(\mathbf{x}) = \frac{1}{n-1} \sum_{j=1}^{n} [k(\mathbf{x}_j^{\ell}, \mathbf{x}) \nabla_{\mathbf{x}_j^{\ell}} \log p(\mathbf{x}_j^{\ell}) + \nabla_{\mathbf{x}_j^{\ell}} k(\mathbf{x}_j^{\ell}, \mathbf{x})] \quad (6.2)$$

where k is some (similarity) kernel and ϵ_ℓ the step size. Although SVGD was developed with the intent of sampling a set of particles that approximate some distribution p without the direct goal of obtaining diverse samples, SVGD and our method have a close relation.

This relation between our method and SVGD can be best illustrated under specific choices for drift and potential under which the probability flow ODE discretization of *particle guidance* can be approximated as (derivation in Appendix A.3.5):

$$\mathbf{x}_i^{t+\Delta t} \approx \mathbf{x}_i^{t} + \epsilon_t(\mathbf{x}_i) \psi_t(\mathbf{x}_i^{t}) \quad \text{where} \quad \psi(\mathbf{x}) = \frac{1}{n-1} \sum_{j=1}^{n} [k_t(\mathbf{x}_j^{t}, \mathbf{x}) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}_j^{t}} k_t(\mathbf{x}_j^{t}, \mathbf{x})] \quad (6.3)$$

Comparing this with Eq. 6.2, we can see a clear relation in the form of the two methods, with some key distinctions. Apart from the different constants, the two methods use different terms for the total score component. Interestingly both methods use smoothed-out scores, however, on the one hand, particle guidance uses the *diffused* score at the specific particle $\mathbf{x}_i$, $\nabla_{\mathbf{x}_i} \log p_t(\mathbf{x}_i)$, while on the other, SVGD smoothes it out by taking a weighted average of the score of nearby particles weighted by the similarity kernel $(\sum_j k(\mathbf{x}_i, \mathbf{x}_j) \nabla_{\mathbf{x}_j} \log p(\mathbf{x}_j)) / (\sum_j k(\mathbf{x}_i, \mathbf{x}_j))$.

The reliance of SVGD on other particles for the "smoothing of the score", however, causes two related problems, firstly, it does not have the *finite-time sampling* guarantee that the time evolution of diffusion models provides and, secondly, it suffers from the collapse to few local modes near the initialization and cannot discover isolated modes in data distribution [175]. This challenge has been theoretically [176] and empirically [177] studied with several works proposing practical solutions. In particular, relevant works use an annealing schedule to enhance exploration [178] or use score matching to obtain a noise-conditioned kernel for SVGD [179]. Additionally, we empirically observe that the score smoothing in SVGD results in blurry samples in image generation.

6.3.3 Electrostatics

Recent works [31], [50] have shown promise in devising novel generative models inspired by the evolution of point charges in high-dimensional electric fields defined by the data distribution. It becomes natural therefore to ask whether particle guidance could be seen as describing the evolution of point charges when these are put in the same electric field such that they are not only attracted by the data distribution but also repel one another. One can show that this evolution can indeed be seen as the combination of Poisson Flow Generative Models with particle guidance, where the similarity kernel is the extension of Green's function in N+1-dimensional space, *i.e.*, $k(x, y) \propto 1/||x - y||^{N-1}$. We defer more details to Appendix A.3.5.

6.4 Fixed Potential Particle Guidance

In this section, we will present and study a simple, yet effective, instantiation of particle guidance based on the definition of the time-evolving potential as a combination of predefined kernels. As we will see in the experiments in Section 6.4.2, this leads to significant sample efficiency improvements with no additional training required and little inference overhead.

To promote diversity and sample efficiency, in our experiments, we choose the potential $\log \Phi_t$ to be the negative of the sum of a pairwise similarity kernel k between each pair of particles $\log \Phi_t(\mathbf{x}_1, ...\mathbf{x}_n) = -\frac{\alpha_t}{2} \sum_{i,j} k_t(\mathbf{x}_i, \mathbf{x}_j)$ obtaining:

$$\mathrm{d}\mathbf{x}_i = \left[-\mathbf{f}(\mathbf{x}_i, t') + g^2(t') \left(\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i) - \alpha_{t'} \nabla_{\mathbf{x}_i} \sum_{j=1}^{n} k_{t'}(\mathbf{x}_i, \mathbf{x}_j) \right) \right] \mathrm{d}t + g(t') \mathrm{d}\mathbf{w} \tag{6.4}$$

Intuitively, the kernel term will push our different samples to be dissimilar from one another while at the same time the score term will try to match our distribution. Critically, this does not come at a significant additional runtime as, in most domains, the cost of running the pairwise similarity kernels is very small compared to the execution of the large score network architecture. Moreover, it allows the use of domain-specific similarity kernels and does not require training any additional classifier or score model. We can also view the

particle guidance Equation 6.4 as a sum of reverse-time SDE and a guidance term. Thus, to attain a more expedited generation speed, the reverse-time SDE can also be substituted with the probability flow ODE [180].

6.4.1 Theoretical Analysis

To understand the effect that *particle guidance* has beyond simple intuition, we study the joint distribution of sets of particles generated by the proposed reverse diffusion. However, unlike methods related to energy-based models (see coupled replicas, metadynamics, SVGD in Sec. 6.3) analyzing the effect of the addition of a time-evolving potential $\log \Phi_t$ in the reverse diffusion is non-trivial.

While the score component in *particle guidance* is the score of the sequence of probability distributions $\tilde{p}_t(\mathbf{x}_1, \dots, \mathbf{x}_n) = \Phi_t(\mathbf{x}_1, \dots, \mathbf{x}_n) \prod_{i=1}^n p_t(\mathbf{x}_i)$, we are not necessarily sampling exactly $\tilde{p}_0$ because, for an arbitrary time-evolving potential Φ_t, this sequence of marginals does not correspond to a diffusion process. One strategy used by other works in similar situations [181] relies on taking, after every step or at the end, a number of Langevin steps to reequilibrate and move the distribution back towards $\tilde{p}_t$. This, however, increases significantly the runtime cost (every Langevin step requires score evaluation) and is technically correct only in the limit of infinite steps leaving uncertainty in the real likelihood of our samples. Instead, in Theorem 5, we use the Feynman-Kac theorem to derive a formula for the exact reweighting that particle guidance has on a distribution (derivation in Appendix A.3.1):

Theorem 5. *Under integrability assumptions, sampling $\mathbf{x}_1^T, ..., \mathbf{x}_n^T$ from p_T and following the particle guidance reverse diffusion process, we obtain samples from the following joint probability distribution at time $t = 0$:*

$$\hat{p}_0(\mathbf{x}_1, \dots, \mathbf{x}_n) = \mathbb{E}[Z \exp[-\int_0^T g(t)^2 \{\langle \nabla \log \Phi_t(\mathbf{X}_t), \nabla \log \hat{p}_t(\mathbf{X}_t) \rangle + \Delta \log \Phi_t(\mathbf{X}_t)\} \mathrm{d}t]],$$

with Z (explicit in the appendix) such that

$$\prod_{i=1}^N p_0(\mathbf{x}_i) = \mathbb{E}[Z],$$

$(\mathbf{X}_t)_{t\in[0,T]}$ *is a stochastic process driven by the equation*

$$\mathrm{d}\mathbf{X}_t = \{\mathbf{f}(\mathbf{X}_t, t) - g(t)^2 \nabla \log p_t(\mathbf{X}_t)\}\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \qquad \mathbf{X}_0 = \{\mathbf{x}_i\}_{i=1}^N.$$

Hence the density $\hat{p}_0$ can be understood as a reweighting of the random variable Z that represents I.I.D. sampling.

Riemannian Manifolds. Note that our theoretical insights can also be extended to the manifold framework. This is a direct consequence of the fact that the Feynman-Kac theorem can be extended to the manifold setting, see for instance [182].

Preserving Invariances The objects that we learn to sample from with generative models often present invariances such as the permutation of the atoms in a molecule or the roto-translation of a conformer. To simplify the learning process and ensure these are respected, building such invariances in the model architecture is common practice. In the case of diffusion models, to obtain a distribution that is invariant to the action of some group G, such as that of rotations or permutations, it suffices to have an invariant prior and build a score model that is G-equivariant [183], [184]. Similarly, in our case, we are interested in distributions that are invariant to the action of G on any of the set elements (see Section 6.4.2), we show that a sufficient condition for this invariance to be maintained is that the time-evolving potential Φ_t is itself invariant to G-transformations of any of its inputs (see Proposition 3 in Appendix A.3.4).

6.4.2 Experiments

Fixed potential particle guidance can be implemented on top of any existing trained diffusion model with the only requirement of specifying the potential/kernel to be used in the domain. We present three sets of empirical results in three very diverse domains. First, in Appendix B.4.2, we work with a synthetic experiment formed by a two-dimensional Gaussian mixture model, where we can visually highlight some properties of the method. In this section instead, we consider text-to-image and molecular conformer generation, two important tasks where

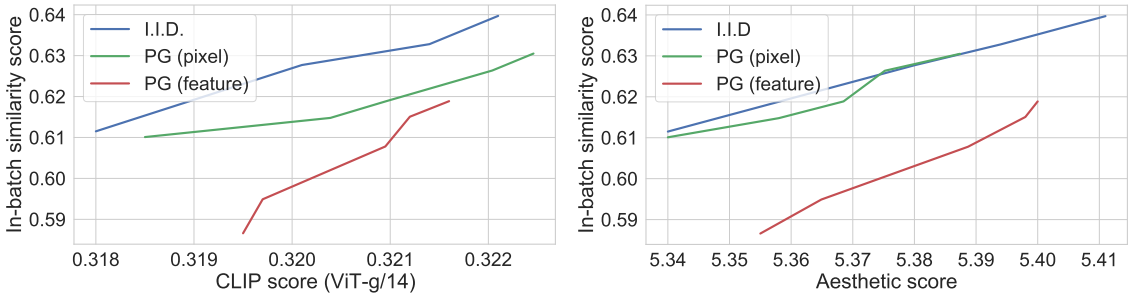
diffusion models have established new state-of-the-art performances, and show how, in each of these tasks, particle guidance can provide improvements in sample efficiency pushing the diversity-quality Pareto frontier.

Text-to-image Generation

In practice, the most prevalent text-to-image diffusion models, such as Stable Diffusion [153] or Midjourney, generally constrain the output budget to four images per given prompt. Ideally, this set of four images should yield a diverse batch of samples for user selection. However, the currently predominant method of classifier-free guidance [159] tends to push the mini-batch samples towards a typical mode to enhance fidelity, at the expense of diversity.

To mitigate this, we apply the proposed particle guidance to text-to-image generation. We use Stable Diffusion v1.5, pre-trained on LAION-5B [158] with a resolution of 512×512, as our testbed. We apply an Euler solver with 30 steps to solve for the ODE version of particle guidance. Following [185], we use the validation set in COCO 2014 [162] for evaluation, and the CLIP [163]/Aesthetic score [165] (higher is better) to assess the text-image alignment/visual quality, respectively. To evaluate the diversity within each batch of generated images corresponding to a given prompt, we introduce the *in-batch similarity score*. This metric represents the average pairwise cosine similarity of features within an image batch, utilizing the pre-trained DINO [186] as the feature extractor. Contrasting the FID score, the in-batch similarity score specifically measures the diversity of a batch of images generated for a given prompt. We use a classifier-free guidance scale from 6 to 10 to visualize the trade-off curve between the diversity and CLIP/Aesthetic score, in line with prior works [160], [185]. For particle guidance, we implement the RBF kernel on the down-sampled pixel space (the latent space of the VAE encoder-) in Stable Diffusion, as well as the feature space of DINO. Please refer to Appendix B.4.2 for more experimental details.

As shown in Figure 6.2a and Figure 6.2b, particle guidance (PG) consistently obtains a better (lower) in-batch similarity score in most cases, given the same CLIP/Aesthetic score, with a classifier-free guidance scale ranging from 6 to 10. Conversely, we observe that while the in-batch similarity score of I.I.D. sampling improves with the reduced classifier-free guidance scale, particle guidance continues to surpass I.I.D. sampling in terms of CLIP/Aesthetic score



(a) In-batch similarity versus CLIP score (b) In-batch similarity versus Aesthetic score

Figure 6.2: In-batch similiarity score versus **(a)** CLIP ViT-g/14 score and **(b)** Aesthetic score for text-to-image generation at 512×512 resolution, using Stable Diffusion v1.5 with a varying guidance scale from 6 to 10.

(a) I.I.D. (b) PG (c) Training data (d) I.I.D. (e) PG

Figure 6.3: Text prompt: (a,b) “A baby eating a cake with a tie around his neck with balloons in the background" (COCO); (c,d,e) “VAN GOGH CAFE TERASSE copy.jpg", with original training data in (c).

given the same in-batch similarity. When the potential is the similarity kernel applied in the feature space, particle guidance notably attains a lower in-batch similarity score compared to I.I.D. sampling or to the approach in the original downsampled pixel space. This suggests that utilizing a semantically meaningful feature space is more appropriate for determining distances between images.

In Figure 6.3, we further visualize generated batches of four images per prompt by I.I.D. sampling and particle guidance (feature) with the same random seeds, when fixing the classifier-free guidance scale to 9. We can see that particle guidance improves the visual diversity in the generated batch. Interestingly, particle guidance can also help to alleviate the memorization issue of Stable Diffusion [187]. For example, given the text prompt of a painting from LAION dataset, particle guidance (Figure 6.3d) avoids the multiple replications of the training data in the I.I.D. setting (the top-left and the bottom-right images in Figure 6.3c). We provide extended samples in Appendix B.4.3, and additionally show that SVGD (Eq. 6.2) fails to promote diversity, instead yielding a set of blurry images.

Molecular Conformer Generation

Molecular conformer generation is a key task in computational chemistry that consists of finding the set of different conformations that a molecule most likely takes in 3D space. Critically it is often important to find all or most of the low-energy conformers as each can determine a different behavior (e.g. by binding to a protein). This necessity is reflected in the metrics used by the community that look both at coverage (also called recall) and precision over the set predictions.

Over the past few years, molecular conformer generation has been extensively studied by the machine learning community, with well-established benchmarks [188] and several generative models designed specifically for this task [168], [184], [189]. However, all these methods are based on training a generative model to generate single samples and then running this model several times (more than 200 on average in the standard GEOM-DRUGS dataset) to generate a large number of I.I.D. samples.

As discussed before, however, this strategy is suboptimal to generate representative sets of samples and cover the distribution. Therefore, we take the state-of-the-art conformer

Table 6.1: Quality of generated conformer ensembles for the GEOM-DRUGS test set in terms of Coverage (%) and Average Minimum RMSD (Å). We follow the experimental setup from [189], for experimental details and introduction of the baselines please refer to Appendix B.4.2.

	Recall				Precision			
	Coverage ↑		AMR ↓		Coverage ↑		AMR ↓	
Method	Mean	Med	Mean	Med	Mean	Med	Mean	Med
RDKit ETKDG	38.4	28.6	1.058	1.002	40.9	30.8	0.995	0.895
OMEGA	53.4	54.6	0.841	0.762	40.5	33.3	0.946	0.854
GeoMol	44.6	41.4	0.875	0.834	43.0	36.4	0.928	0.841
GeoDiff	42.1	37.8	0.835	0.809	24.9	14.5	1.136	1.090
Torsional Diffusion	72.7	80.0	0.582	0.565	55.2	56.9	0.778	0.729
TD w/ particle guidance	**77.0**	**82.6**	**0.543**	**0.520**	**68.9**	**78.1**	**0.656**	**0.594**

generation model, *torsional diffusion*, and, without retraining the model itself, we show that we can obtain significant improvements in both coverage and precision via particle guidance.

Torsional diffusion [168] defines the diffusion process over the manifold defined by changes in torsion angles from some initial conformer because of the relative rigidity of the remaining degrees of freedom. Given this observation, we also define the guidance kernel on this manifold as an RBF kernel over the dihedral angle differences.

Another important consideration when dealing with molecular conformers is given by the permutation symmetries that characterize several molecules: conformers that appear different might be very similar under permutations of the order of the atoms that do not change the bond structure. To maximize the sample efficiency and avoid generating similar conformers, we make the kernel invariant to these transformations. For this, we employ the simple strategy to take the minimum value of the original kernel under the different perturbations (formalized in Appendix B.4.2).

Table 6.1 shows that by applying particle guidance to SDE-based reverse process of torsional diffusion (see Appendix B.4.2 for details) we are able to balance coverage and precision being able to obtain, without retraining the model, significantly improved results on both metrics with 8% and 19% simultaneous reductions respectively in recall and precision median AMR.

6.5 Learned Potential Particle Guidance

While the fixed potential particle guidance seen so far is very effective in improving the diversity of samples with little overhead, it is hard to argue about the optimality of the resulting joint distribution. This is because of the complexity of the expression obtained in Theorem 5 and its dependence on the data distribution itself. Furthermore, in some domains, particularly in scientific applications, researchers need to control the distribution that they are sampling. This is necessary, for example, to apply correct importance weights or compute free energy differences. While Theorem 5 allows us to theoretically analyze properties of the distribution, the joint and marginal distributions remain largely intractable.

In this section, we analyze how we can sample from desired joint probability distribution by learning a tailored time-evolving potential for particle guidance. Using the maximum entropy theorem [190], we can show that the distribution satisfying a bound on the expected value of a (diversity) metric Φ_0 while minimizing the KL divergence with the independent distribution is:

$$\hat{p}_0(\mathbf{x}_1, ..., \mathbf{x}_n) \propto \Phi_0(\mathbf{x}_1, ..., \mathbf{x}_n)^{\beta(\alpha)} \prod_{i=1}^{n} p(\mathbf{x}_i) \tag{6.5}$$

where β is a function of α, the value of the bound on $E_{\hat{p}}[\log \Phi_0]$.

6.5.1 Training Procedure

We now have to learn a time-evolving potential Φ_t that when used as part of the particle guidance framework generates $\hat{p}_0$ (we assume Φ_0 is chosen such that $\beta(\alpha) = 1$). To achieve this, we mandate that the generation process of particle guidance in Eq. 6.1 adheres to the sequence of marginals $\hat{p}_t(\mathbf{x}_1^t, ..., \mathbf{x}_n^t) = \Phi_t(\mathbf{x}_1^t, ..., \mathbf{x}_n^t) \prod_{i=1}^{n} p_t(\mathbf{x}_i^0)$ and learn Φ_t^θ to satisfy this evolution. Under mild assumptions, using Doob h-theory (derivation in Appendix A.3.2), we show that we can learn the Φ_t^θ by the following objective:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\mathbf{x}_1^0, ..., \mathbf{x}_n^0 \sim p_0} \, \mathbb{E}_{\mathbf{x}_i^t \sim p_{t|0}(\cdot|\mathbf{x}_i^0)} [\|\Phi_0(\mathbf{x}_1^0, ..., \mathbf{x}_n^0) - \Phi_t^\theta(\mathbf{x}_1^t, ..., \mathbf{x}_n^t)\|^2] \tag{6.6}$$

where $p_{t|0}$ is the Gaussian perturbation kernel in diffusion models. Importantly, here the

initial $\mathbf{x}_i^0$ are sampled independently from the data distribution so this training scheme can be easily executed in parallel to learning the score of p_t.

6.5.2 Preserving Marginal Distributions

While the technique discussed in the previous section is optimal in the maximum entropy perspective, it does not (for arbitrary Φ_0) preserve the marginal distributions of individual particles, i.e. marginalizing $\mathbf{x}_i$ over $\hat{p}$ does not recover p. Although not critical in many settings and not respected, for a finite number of particles, neither by the related methods in Section 6.3 nor by the fixed potential PG, this is an important property in some applications.

Using again the maximum entropy theorem, we can show that the distribution satisfying a bound on the expected value of a (diversity) metric Φ_0' and preserving the marginal distribution while minimizing the KL divergence with the independent distribution can be written as:

$$\hat{p}_0(\mathbf{x}_1, ..., \mathbf{x}_n) \propto \Phi_0'(\mathbf{x}_1, ..., \mathbf{x}_n)^{\beta(\alpha)} \prod_{i=1}^{n} p(\mathbf{x}_i)\gamma_\theta(\mathbf{x}_i) \tag{6.7}$$

for some scalar function over individual particles γ_θ. In Appendix A.3.3, we derive a new training scheme to learn the parameters of γ_θ. This relies on setting the normalization constant to an arbitrary positive value and learning values of θ that respect the marginals. Once γ_θ is learned, its parameters can be frozen and the training procedure of Eq. 6.6 can be started.

6.6 Conclusion

In this chapter, we have analyzed how one can improve the sample efficiency of generative models by moving beyond I.I.D. sampling and enforcing diversity, a critical challenge in many real applications that has been largely unexplored. Our proposed framework, particle guidance, steers the sampling process of diffusion models toward more diverse sets of samples via the definition of a time-evolving joint potential. We have studied the theoretical properties of the framework such as the joint distribution it converges to for an arbitrary potential and how to learn potential functions that sample some given joint distribution achieving optimality

and, if needed, preserving marginal distributions. Finally, we evaluated its performance in two important applications of diffusion models text-to-image generation and molecular conformer generation, and showed how in both cases it is able to push the Pareto frontier of sample diversity vs quality.

We hope that particle guidance can become a valuable tool for practitioners to ensure diversity and fair representation in existing tools even beyond the general definition of diversity directly tackling known biases of generative models. Further, we hope that our methodological and theoretical contributions can spark interest in the research community for better joint-particle sampling methods.

Part III

Novel Generative Models from Physical Processes

In Part III, we explore new generative models inspired by physical processes beyond the Brownian motion utilized in diffusion models. Chapters 7 and 8 introduce a novel family of generative models based on electrostatic theory, which competes with diffusion models in terms of sample quality and robustness. Additionally, we present a systematic approach for translating physical processes into valid generative models in Chapter 9.

Chapter 7

Generative Models from Electrostatics

In previous chapters, we focus on techniques developed for a particular type of physics-inspired generative model: diffusion models. The stability and scalability of diffusion models stem from their underlying physical process, known as the diffusion process in thermodynamics. This physical process describes a natural and smooth degradation from data to a simple prior distribution. However, numerous other physical processes have similar properties. These are largely unexplored and hold great potential.

In this chapter, we propose a new "Poisson flow" generative model (PFGM) inspired by *electrostatics*, mapping a uniform distribution on a high-dimensional hemisphere to any data distribution. We interpret the data points as electrical charges on the $z = 0$ hyperplane in a space augmented with an additional dimension z, generating a high-dimensional electric field (the gradient of the solution to the Poisson equation). We prove that if these charges flow upward along electric field lines, their initial distribution in the $z = 0$ plane transforms into a distribution on the hemisphere of radius r that becomes *uniform* in the $r \to \infty$ limit. To learn the bijective transformation, we estimate the normalized field in the augmented space. For sampling, we devise a backward ODE that is anchored by the physically meaningful additional dimension: the samples hit the (unaugmented) data manifold when the z reaches zero. Experimentally, PFGM achieves current state-of-the-art performance among the normalizing flow models on CIFAR-10. It also performs on par with the state-of-the-art SDE approaches while offering significant acceleration on image generation tasks. Additionally, PFGM appears more tolerant of estimation errors on weaker network architecture and robust

to the step size in the Euler method.

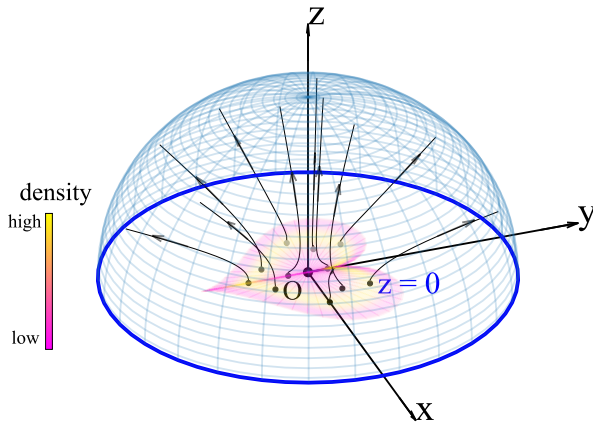
This chapter was previously published as [30].

7.1 Introduction

Although recent advances on diffusion [167] and scored-based models [34] achieve comparable sample quality to GAN's without adversarial training, these models have a slow stochastic sampling process. [34] proposes backward ODE samplers (normalizing flow) that speed up the sampling process but these methods have not yet performed on par with the SDE counterparts.

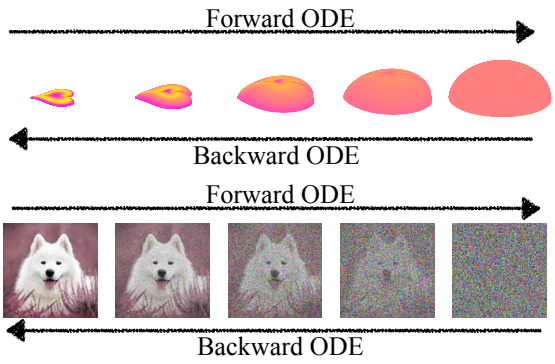
In this chapter, we present a new "Poisson flow" generative model (**PFGM**), exploiting a remarkable physics fact that generalizes to N dimensions. As illustrated in Figure 7.1a, motion in a viscous fluid transforms any planar charge distribution into a uniform angular distribution. Specifically, we interpret N-dimensional data points $\mathbf{x}$ (images, say) as positive electric charges in the $z = 0$ plane of an $N + 1$-dimensional space (see Figure 7.1a) filled with a viscous liquid (say honey). A positive charge with $z > 0$ will be repelled by the other charges and move in the direction of their repulsive force, eventually crossing an imaginary hemisphere of radius r. We show that, remarkably, if the original charge distribution is let loose just above $z = 0$, this law of motion will cause a *uniform* distribution for their hemisphere crossings in the $r \to \infty$ limit.

Our Poisson flow generative process reverses the forward process: we generate a uniform distribution of negative charges on the hemisphere, then track their motion back to the $z = 0$ plane, where they will be distributed as the data distribution. A Poisson flow can be viewed as a type of continuous normalizing flows [34], [191], [192] in the sense that it continuously maps between an arbitrary distribution and an easily sampled one: in the previous works an N-dimensional Gaussian and in PFGM a uniform distribution on an N-dimensional hemisphere. In practice, we implement the Poisson flow by solving a pair of forward/backward ordinary differential equations (ODEs) induced by the electric field (Figure 7.1b) given by the N-dimensional version of Coulomb's law (the gradient of the solution to the Poisson's equation with the data as sources). We will interchangeably refer to this gradient as the *Poisson field*,



(a)

case $N = 3$.



(b)

Figure 7.1: **(a)** 3D Poisson field trajectories for a heart-shaped distribution **(b)** The evolvements of a distribution (**top**) or an (augmented) sample (**bottom**) by the forward/backward ODEs pertained to the Poisson field.

The proposed generative model PFGM has a stable training objective and empirically outperforms previously state-of-the-art continuous flow methods [34], [54]. As a different iterative method, PFGM offers two advantages compared to score-based methods [34], [49]. First, the ODE process of PFGM achieves faster sampling speeds than the SDE samplers in [34]. while retaining comparable performance. Second, our backward ODE exhibits better generation performance than the reverse-time ODEs of VE/VP/sub-VP SDEs [34], as well as greater stability on a weaker architecture NSCNv2 [49]. The rationale for robustness is that the time variables in these ODE baselines are strongly correlated with the sample norms during training time, resulting in a less error-tolerant inference. In contrast, the tie between the anchored variable and the sample norm in PFGM is much weaker.

Experimentally, we show that PFGM achieves current state-of-the-art performance on CIFAR-10 dataset in the normalizing flow family, with FID/Inception scores of 2.48/9.65 (w/ DDPM++ [34]) and 2.35/9.68 (w/ DDPM++ deep [34]). It performs competitively with current state-of-the-art SDE samplers [34] and provides 10× to 20× speed up across datasets. Notably, the backward ODE in PFGM is the *only* ODE-based sampler that can produce decent samples on its own on NCSNv2 [49], while other ODE baselines fail without corrections. In addition, PFGM demonstrates the robustness to the step size in the Euler method, with a varying number of function evaluations (NFE) ranging from 10 to 100. We further showcase the

utility of the invertible forward/backward ODEs of the Poisson field on likelihood evaluation and image manipulations, and its scalability to higher resolution images on LSUN bedroom 256×256 dataset.

7.2 Background

Poisson equation Let $\mathbf{x} \in \mathbb{R}^N$ and $\rho(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ be a *source* function. We assume that the source function has a compact support, $\rho \in \mathcal{C}^0$ and $N \geq 3$. The Poisson equation is

$$\nabla^2 \varphi(\mathbf{x}) = -\rho(\mathbf{x}), \tag{7.1}$$

where $\varphi(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ is called the *potential function*, and $\nabla^2 \equiv \sum_{i=1}^N \frac{\partial^2}{\partial x_i^2}$ is the Laplacian operator. It is usually helpful to define the gradient field $\mathbf{E}(\mathbf{x}) = -\nabla\varphi(\mathbf{x})$ and rewrite the Poisson equation as $\nabla \cdot \mathbf{E} = \rho$, known in physics as Gauss's law [193]. The Poisson equation is widely used in physics, giving rise to Newton's gravitational theory [194] and the electrostatic theory [193], when $\rho(\mathbf{x})$ is interpreted as mass density or electric charge density, respectively. $\mathbf{E}$ is the N-dimensional analog of the electric field. The Poisson equation Eq. (7.1) (with zero boundary condition at infinity) admits a unique simple integral solution [1]:

$$\varphi(\mathbf{x}) = \int G(\mathbf{x}, \mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \quad G(\mathbf{x}, \mathbf{y}) = \frac{1}{(N-2)S_{N-1}(1)} \frac{1}{||\mathbf{x} - \mathbf{y}||^{N-2}}, \tag{7.2}$$

where $S_{N-1}(1)$ is a geometric constant representing the surface area of the unit $(N-1)$-sphere [2], and $G(\mathbf{x}, \mathbf{y})$ is the extension of Green's function in N-dimensional space (details in Appendix A.4.4). The negative gradient field of $\varphi(\mathbf{x})$, referred as *Poisson field* of the source ρ, is

$$\mathbf{E}(\mathbf{x}) = -\nabla\varphi(\mathbf{x}) = -\int \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \quad \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) = -\frac{1}{S_{N-1}(1)} \frac{\mathbf{x} - \mathbf{y}}{||\mathbf{x} - \mathbf{y}||^N}. \tag{7.3}$$

Qualitatively, the Poisson field $\mathbf{E}(\mathbf{x})$ points away from sources, or equivalently $-\mathbf{E}(\mathbf{x})$ points

[1] Eq. (7.2) is valid for $N \geq 3$. When $N = 2$, the Green's function is $G(\mathbf{x}, \mathbf{y}) = -\log(||\mathbf{x} - \mathbf{y}||)/2\pi$. We assume $N \geq 3$ since N is typically large in the relevant applications.

[2] The N-sphere with radius r is defined as $\{\mathbf{x} \in \mathbb{R}^{N+1}, ||\mathbf{x}|| = r\}$

towards sources, as illustrated in Figure 7.1. It is straightforward to check that when $\rho(\mathbf{x}) \to \delta(\mathbf{x}-\mathbf{y})$, we get $\varphi(\mathbf{x}) \to G(\mathbf{x},\mathbf{y})$ and $\mathbf{E}(\mathbf{x}) \to -\nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y})$. This implies that $G(\mathbf{x},\mathbf{y})$ and $-\nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y})$ can be interpreted as the potential function and the gradient field generated by a unit point source, *e.g.*, a point charge, located at $\mathbf{y}$. When $\rho(\mathbf{x})$ takes general forms but has bounded support, simple asymptotics exist for $||\mathbf{x}|| \gg ||\mathbf{y}||$. To the lowest order, $\mathbf{E}(\mathbf{x}) = \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y})|_{\mathbf{y}=\mathbf{0}} \sim \mathbf{x}/||\mathbf{x}||^N$ behaves as if it were generated by a unit point source at $\mathbf{y} = 0$. In physics, the power law decay is considered to be long-range (compared to exponential decay) [193].

Particle dynamics in a Poisson field The Poisson field immediately defines a flow model, where the probability distribution evolves according to the gradient flow $\partial p_t(\mathbf{x})/\partial t = -\nabla \cdot (p_t(\mathbf{x})\mathbf{E}(\mathbf{x}))$. The gradient flow is a special case of the Fokker-Planck equation [195], where the diffusion coefficient is zero. Intuitively we can think of $p_t(\mathbf{x})$ as represented by a population of particles. The corresponding (non-diffusion) case of the Itô process is the forward ODE $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{E}(\mathbf{x})$. We can interpret the trajectories of the ODE as particles moving according to the Poisson field $\mathbf{E}(x)$, with initial states drawn from p_0. The physical picture of the forward ODE is a charged particle under the influence of electric fields in the overdamped limit (details in Appendix A.4.5).

The dynamics is also *rescalable* in the sense that the particle trajectory remains the same for $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \pm f(\mathbf{x})\mathbf{E}(\mathbf{x})$ for $f(\mathbf{x}) > 0, f(\mathbf{x}) \in \mathcal{C}^1$, because the time rescaling $dt \to f(\mathbf{x}(t))dt$ recovers $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \pm\mathbf{E}(\mathbf{x})$. Note that the dynamics is stiff due to the power law factor in the denominator in Equation 7.3, posing computational challenges. Luckily the rescalablility allows us to rescale $\mathbf{E}(\mathbf{x})$ properly to get new ODEs (formally defined later in Section 7.3.3) that are more amenable for sampling.

Generative Modeling via ODE Generative modeling can be done by transforming a base distribution to a data distribution via mappings defined by ODEs. The ODE-based samplers allow for adaptive sampling, exact likelihood evaluation and modeling of continuous-time dynamics [34], [191]. Previous works broadly fall into two lines. [191], [196] introduce a continuous-time normalizing flow model that can be trained with maximum likelihood by the instantaneous change-of-variables formula [191]. For sampling, they directly integrate the learned invertiable mapping over time. Another work [34] unifies the scored-based model [49],

[197] and diffusion model [167] into a general diffusion process, and uses the reverse-time ODE of the diffusion process for sampling. They show that the reverse-time ODE produces high quality samples with improved architecture.

7.3 Poisson Flow Generative Models: Learning and Inference

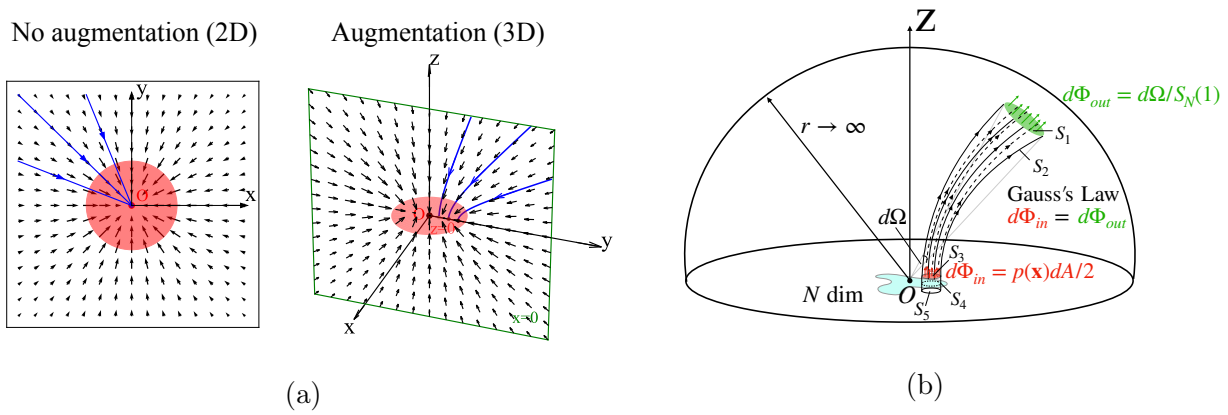
In this section, we start with the properties of the Poisson flow in the augmented space and show how to draw samples from the data distribution by following the backward ODE of the Poisson flow (Section 7.3.1). We then discuss how to actually learn a normalized Poisson field from data samples through simulations of the forward ODE (Section 7.3.2) and present an equivalent backward ODE that allows for exponentially decay on z (Section 7.3.3).

7.3.1 Augmenting the Data with Additional Dimension

We wish to generate samples $\mathbf{x} \in \mathbb{R}^N$ from a distribution $p(\mathbf{x})$ supported on a bounded region. We may set the source $\rho(\mathbf{x}) = p(\mathbf{x}) \in \mathcal{C}^0$ [3] and compute the resulting gradient field $\mathbf{E}(\mathbf{x})$ from Eq. (7.3). Since $-\mathbf{E}(\mathbf{x})$ points towards sources, the backward ODE $d\mathbf{x}/dt = -\mathbf{E}(\mathbf{x})$ will take samples close to the sources. One may naively hope that the backward ODE is a generative model that recovers $p(\mathbf{x})$. Unfortunately, the backward ODE has the problem of mode collapse. We illustrate this phenomenon with a 2D uniform disk. The reverse Poisson field $-\mathbf{E}(\mathbf{x})$ on the 2D (x, y)-plane points towards the center of the disk O (Figure 7.2a left), so all particle trajectories (blue lines) will eventually hit O. If we instead add an additional dimension z (Figure 7.2a right), particles can hit different points on the disk and faithfully recover the data distribution.

Consequently, instead of solving the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -p(\mathbf{x})$ in the original data space, we solve the Poisson equation in an augmented space $\tilde{\mathbf{x}} = (\mathbf{x}, z) \in \mathbb{R}^{N+1}$ with an additional variable $z \in \mathbb{R}$. We augment the training data $\tilde{\mathbf{x}}$ in the new space by setting

[3] A probability distribution $p(\mathbf{x})$ is a special case of "charge density" $\rho(x)$ because $p(\mathbf{x})$ need to be non-negative and integrates to unity. Here we focus on applications to probability distribution of data, which is the objective to be modeled in generative modeling.



(a) (b)

Figure 7.2: **(a)** Poisson field (black arrows) and particle trajectories (blue lines) of a 2D uniform disk (red). **Left** (no augmentation, 2D): all particles collapse to the disk center. **Right** (augmentation, 3D): particles hit different points on the disk. **(b)** Proof idea of Theorem 6. By Gauss's Law, the outflow flux $d\Phi_{out}$ equals the inflow flux $d\Phi_{in}$. The factor of two in $p(\mathbf{x})\mathrm{d}A/2$ is due to the symmetry of Poisson fields in $z < 0$ and $z > 0$.

$z = 0$ such that $\tilde{\mathbf{x}} = (\mathbf{x}, 0)$. As a consequence, the data distribution in the augmented space is $\tilde{p}(\tilde{\mathbf{x}}) = p(\mathbf{x})\delta(z)$, where δ is the Dirac delta function. By Equation 7.3, the Poisson field by solving the new Poisson equation $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$ has an analytical form:

$$\forall \tilde{\mathbf{x}} \in \mathbb{R}^{N+1}, \mathbf{E}(\tilde{\mathbf{x}}) = -\nabla\varphi(\tilde{\mathbf{x}}) = \frac{1}{S_N(1)} \int \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{y}}}{||\tilde{\mathbf{x}} - \tilde{\mathbf{y}}||^{N+1}} \tilde{p}(\tilde{\mathbf{y}}) d\tilde{\mathbf{y}} \tag{7.4}$$

The associated forward/backward ODEs of the Poisson field are $d\tilde{\mathbf{x}}/dt = \mathbf{E}(\tilde{\mathbf{x}}), d\tilde{\mathbf{x}}/dt = -\mathbf{E}(\tilde{\mathbf{x}})$. Intuitively, theses ODEs uniquely define trajectories of particles between the $z = 0$ hyperplane and an enclosing hemisphere (cf. Figure 7.1a). In the following theorem, we show that the backward ODE defines a transformation between the uniform distribution on an infinite hemisphere and the data distribution $\tilde{p}(\tilde{\mathbf{x}})$ in the $z = 0$ plane. We present the formal proof to Appendix A.4, illustrated by Figure 7.2b. The proof is based on the idea that when the radius of hemisphere $r \to \infty$, the data distribution $\tilde{p}(\tilde{\mathbf{x}})$ can be effectively viewed as a delta distribution at origin. Consequently, the Poisson field points in the radial direction at $r \to \infty$, perpendicular to $S_N^+(r)$ (Green arrows in Figure 7.2b).

Theorem 6. *Suppose particles are sampled from a uniform distribution on the upper ($z > 0$) half of the sphere of radius r and evolved by the backward ODE $\frac{\mathrm{d}\tilde{\mathbf{x}}}{\mathrm{d}t} = -\mathbf{E}(\tilde{\mathbf{x}})$ until they reach the $z = 0$ hyperplane, where the Poisson field $\mathbf{E}(\tilde{\mathbf{x}})$ is generated by the source $\tilde{p}(\tilde{\mathbf{x}})$. In the*

$r \to \infty$ limit, under some mild conditions detailed in Appendix A.4, this process generates a particle distribution $\tilde{p}(\tilde{\mathbf{x}})$, i.e., a distribution $p(\mathbf{x})$ in the $z = 0$ hyperplane.

Proof sketch. Suppose the flux of the backward ODE connects a solid angle $\mathrm{d}\Omega$ (on $S_N^+(r)$) with an area $\mathrm{d}A$ (on $\mathrm{supp}(\tilde{p}(\tilde{\mathbf{x}}))$. According to Gauss's law, the outflow flux $\mathrm{d}\Phi_{out} = \mathrm{d}\Omega / S_N(1)$ on the hemisphere (Green arrows in Figure 7.2b) equals the inflow flux $\mathrm{d}\Phi_{in} = p(\mathbf{x})\mathrm{d}A/2$ on $\mathrm{supp}(\tilde{p}(\tilde{\mathbf{x}}))$ (Red arrows in Figure 7.2b). $\mathrm{d}\Phi_{in} = \mathrm{d}\Phi_{out}$ gives $\mathrm{d}\Omega/\mathrm{d}A = p(\mathbf{x})S_N(1)/2 \propto p(\mathbf{x})$. Together, by change-of-variable, we conclude that the final distribution in the $z = 0$ hyperplane is $p(\mathbf{x})$. □

The theorem states that starting from an infinite hemisphere, one can recover the data distribution $\tilde{p}$ by following the inverse Poisson field $-\mathbf{E}(\tilde{\mathbf{x}})$. We defer the formal proof and technical assumptions of the theorem to Appendix A.4. The property allows generative modeling by following the Poisson flow of $\nabla^2 \varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$.

7.3.2 Learning the Normalized Poisson Field

Given a set of training data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ i.i.d sampled from the data distribution $p(\mathbf{x})$, we define the empirical version of the Poisson field (Equation 7.4) as follows:

$$\hat{\mathbf{E}}(\tilde{\mathbf{x}}) = c(\tilde{\mathbf{x}}) \sum_{i=1}^{n} \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}}$$

where the gradient field is calculated on n *augmented* datapoints $\{\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 0)\}_{i=1}^n$, and $c(\tilde{\mathbf{x}}) = 1/\sum_{i=1}^n \frac{1}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}}$ is the multiplier for numerical stability. We further normalize the field to resolve the variations in the magnitude of the norm $\| \hat{\mathbf{E}}(\tilde{\mathbf{x}}) \|_2$, and fit the neural network to the more amenable negative normalized field $\mathbf{v}(\tilde{\mathbf{x}}) = -\sqrt{N}\hat{\mathbf{E}}(\tilde{\mathbf{x}})/\| \hat{\mathbf{E}}(\tilde{\mathbf{x}}) \|_2$. The Poisson field is rescalable (cf. Section 7.2) and thus trajectories of its forward/backward ODEs are invariant under normalization. We denote the empirical field calculated on batch data $\mathcal{B}$ by $\hat{\mathbf{E}}_{\mathcal{B}}$ and the negative normalized field as $\mathbf{v}_{\mathcal{B}}(\tilde{\mathbf{x}}) = -\sqrt{N}\hat{\mathbf{E}}_{\mathcal{B}}(\tilde{\mathbf{x}})/\| \hat{\mathbf{E}}_{\mathcal{B}}(\tilde{\mathbf{x}}) \|_2$.

Similar to the scored-based models, we sample points inside the hemisphere by perturbing the augmented training data. Given a training point $\mathbf{x} \in \mathcal{D}$, we add noise to its augmented

version $\{\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 0)\}_{i=1}^n$ to construct the perturbed point $(\mathbf{y}, z)$:

$$\mathbf{y} = \mathbf{x} + \parallel \epsilon_{\mathbf{x}} \parallel (1+\tau)^m \mathbf{u}, \quad z = |\epsilon_z|(1+\tau)^m \tag{7.5}$$

where $\epsilon = (\epsilon_{\mathbf{x}}, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I_{N+1\times N+1})$, $\mathbf{u} \sim \mathcal{U}(S_{N-1}(1))$ and $m \sim \mathcal{U}[0, M]$. The upper limit M, standard deviation σ and τ are hyper-parameters. With fixed ϵ and $\mathbf{u}$, the added noise increases exponentially with m. The rationale behind the design is that points farther away from the data support play a less important role in generative modeling, sharing a similar spirit with the choice of noisy scales in score-based models [34], [49].

In practice, we sample the points by perturbing a mini-batch data $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{B}|}$ in each iteration. We uniformly sample the power m in $[0, M]$ for each datapoint. We select a large M (typically around 300) to ensure the perturbed points can reach a large enough hemisphere. We use a larger batch $\mathcal{B}_L$ for the estimation of normalized field since the empirical normalized field is biased, which empirically gives better results. Denoting the set of perturbed points as $\{\tilde{\mathbf{y}}_i\}_{i=1}^{|\mathcal{B}|}$, we train the neural network f_θ on these points to estimate the negative normalized field by minimizing the following loss:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \parallel f_\theta(\tilde{\mathbf{y}}_i) - \mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \parallel_2^2$$

We summarize the training process in Algorithm 2. In practice, we add a small constant γ to the denominator of the normalized field to overcome the numerical issue when $\exists i, ||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i|| \approx 0$.

Algorithm 2 Learning the normalized Poisson Field

Input: Training iteration T, Initial model f_θ, dataset $\mathcal{D}$, constant γ, learning rate η.
for $t = 1 \dots T$ **do**
  Sample a large batch $\mathcal{B}_L$ from $\mathcal{D}$ and subsample a batch of datapoints $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{B}|}$ from $\mathcal{B}_L$
  Simulate the ODE: $\{\tilde{\mathbf{y}}_i = \text{perturb}(\mathbf{x}_i)\}_{i=1}^{|\mathcal{B}|}$
  Calculate the normalized field by $\mathcal{B}_L$: $\mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) = -\sqrt{N}\hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i)/(\parallel \hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \parallel_2 + \gamma), \forall i$
  Calculate the loss: $\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \parallel f_\theta(\tilde{\mathbf{y}}_i) - \mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \parallel_2^2$
  Update the model parameter: $\theta = \theta - \eta \nabla \mathcal{L}(\theta)$
end for
return f_θ

Algorithm 3 perturb($\mathbf{x}$)

Sample the power $m \sim \mathcal{U}[0, M]$
Sample the initial noise $(\epsilon_{\mathbf{x}}, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I_{(N+1)\times(N+1)})$
Uniformly sample the vector from the unit ball $\mathbf{u} \sim \mathcal{U}(S_N(1))$
Construct training point $\mathbf{y} = \mathbf{x} + \parallel \epsilon_{\mathbf{x}} \parallel (1+\tau)^m \mathbf{u}$, $z = |\epsilon_z|(1+\tau)^m$
return $\tilde{\mathbf{y}} = (\mathbf{y}, z)$

7.3.3 Inference Anchored by the Additional Dimension

After estimating the normalized field $\mathbf{v}$, we can sample from the data distribution by the backward ODE $\mathrm{d}\tilde{\mathbf{x}} = \mathbf{v}(\tilde{\mathbf{x}})\mathrm{d}t$. Nevertheless, the boundary condition of the above ODE is unclear: the starting and terminal time t of the ODE are both unknown. To remedy the issue, we propose an equivalent backward ODE in which $\mathbf{x}$ evolves with the augmented variable z:

$$\mathrm{d}(\mathbf{x}, z) = (\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}z}\mathrm{d}z, \mathrm{d}z) = (\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}, 1)\mathrm{d}z$$

where $\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}, \mathbf{v}(\tilde{\mathbf{x}})_z$ are the corresponding components of $\mathbf{x}, z$ in vector $\mathbf{v}(\tilde{\mathbf{x}})$. In the new ODE, we replace the time variable t with the physically meaningful variable z, permitting explicit starting and terminal conditions: when $z = 0$, we arrive at the data distribution and we can freely choose a large $z_{\max}$ as the starting point in the backward ODE. The backward ODE is compatible with general-purpose ODE solvers, *e.g.*, RK45 method [198] and forward Euler method. The popular black-box ODE solvers, such as the one in Scipy library [199], typically use a common starting time for the same batch of samples. Since the distribution on the $z = z_{\max}$ hyperplane is no longer uniform, we derive the prior distribution by radially projecting uniform distribution on the hemisphere with radius $r = z_{\max}$ to the $z = z_{\max}$ hyperplane:

$$p_{\text{prior}}(\mathbf{x}) = \frac{2z_{\max}^{N+1}}{S_N(z_{\max})(\parallel \mathbf{x} \parallel_2^2 + z_{\max}^2)^{\frac{N+1}{2}}} = \frac{2z_{\max}}{S_N(1)(\parallel \mathbf{x} \parallel_2^2 + z_{\max}^2)^{\frac{N+1}{2}}}$$

where $S_N(r)$ is the surface area of N-sphere with radius r. The reason behind the radial projection is that the Poisson field points in the radial direction at $r \to \infty$. The new backward ODE also defines a bijective transformation between $p_{\text{prior}}(\mathbf{x})$ on the infinite

hyperplane ($z_{\max} \to \infty$) and the data distribution $\tilde{p}(\tilde{\mathbf{x}})$, analogous to Theorem 6. In order to sample from $p_{\text{prior}}(\mathbf{x})$, it is suffice to sample the norm (radius) from the distribution: $p_{\text{radius}}(\| \mathbf{x} \|_2) \propto \| \mathbf{x} \|_2^{N-1} / (\| \mathbf{x} \|_2^2 + z_{\max}^2)^{\frac{N+1}{2}}$ and then uniformly sample its angle. We provide detailed derivations and practical sampling procedure in Appendix A.4.2. We further achieve exponential decay on the z dimension by introducing a new variable t':

$$[\text{Backward ODE}] \quad \mathrm{d}(\mathbf{x}, z) = (\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}} \mathbf{v}(\tilde{\mathbf{x}})_z^{-1} z, z) \mathrm{d}t' \tag{7.6}$$

The z component in the backward ODE, *i.e.*, $\mathrm{d}z = z\mathrm{d}t'$, can be solved by $z = e^{t'}$. Since z reaches zero as $t' \to -\infty$, we instead choose a tiny positive number $z_{\min}$ as the terminal condition. The corresponding starting/terminal time of the variable t' are $\log z_{\max} / \log z_{\min}$ respectively. Empirically, this simple change of variable leads to $2\times$ faster sampling with almost no harm to the sample quality. In addition, we substitue the predicted $\mathbf{v}(\tilde{\mathbf{x}})_z$ with a more accurate one when z is small (Appendix B.5.2). We defer more details of the simulation of backward ODE to Appendix B.5.2.

7.4 Experiments

In this section, we demonstrate the effectiveness of the backward ODE associated with PFGM on image generation tasks. In Section 7.4, we show that PFGM achieves currently best in class performance in the normalizing flow family. In comparison to the existing state-of-the-art SDE or MCMC approaches, PFGM exhibits $10\times$ or $20\times$ acceleration while maintaining competitive or higher generation quality. Meanwhile, unlike existing ODE baselines that heavily rely on corrector to generate decent samples on weaker architectures, PFGM exhibits greater stability against error (Section 7.4). Finally, we show that PFGM is robust to the step size in the Euler method (Section 7.4), and its associated ODE allows for likelihood evaluation and image manipulation by editing the latent space (Section 7.4).

Table 7.1: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE).

	Invertible?	Inception ↑	FID ↓	NFE ↓
PixelCNN [15]	✗	4.60	65.9	1024
IGEBM [21]	✗	6.02	40.6	60
ViTGAN [200]	✗	9.30	6.66	1
StyleGAN2-ADA [48]	✗	9.83	2.92	1
StyleGAN2-ADA (cond.) [48]	✗	10.14	2.42	1
NCSN [197]	✗	8.87	25.32	1001
NCSNv2 [49]	✗	8.40	10.87	1161
DDPM [167]	✗	9.46	3.17	1000
NCSN++ VE-SDE [34]	✗	9.83	2.38	2000
NCSN++ deep VE-SDE [34]	✗	9.89	2.20	2000
Glow [16]	✓	3.92	48.9	1
DDIM, T=50 [54]	✓	-	4.67	50
DDIM, T=100 [54]	✓	-	4.16	100
NCSN++ VE-ODE [34]	✓	9.34	5.29	194
NCSN++ deep VE-ODE [34]	✓	9.17	7.66	194
DDPM++ backbone				
VP-SDE [34]	✗	9.58	2.55	1000
sub-VP-SDE [34]	✗	9.56	2.61	1000
VP-ODE [34]	✓	9.46	2.97	134
sub-VP-ODE [34]	✓	9.30	3.16	146
PFGM (ours)	✓	**9.65**	**2.48**	**104**
DDPM++ deep backbone				
VP-SDE [34]	✗	9.68	2.41	1000
sub-VP-SDE [34]	✗	9.57	2.41	1000
VP-ODE [34]	✓	9.47	2.86	134
sub-VP-ODE [34]	✓	9.40	3.05	146
PFGM (ours)	✓	**9.68**	**2.35**	**110**

Image Generation

Setup For image generation tasks, we consider the CIFAR-10 [201], CelebA 64×64 [51] and LSUN bedroom 256×256 [202]. Following [49], we first center-crop the CelebA images and then resize them to 64×64. We choose $M = 291$ (CIFAR-10 and CelebA)/356 (LSUN bedroom), $\sigma = 0.01$ and $\tau = 0.03$ for the perturbation Algorithm 3, and $z_{\min} = 1e-3$, $z_{\max} = 40$ (CIFAR-10)/60 (CelebA 64^2)/100 (LSUN bedroom) for the backward ODE. We further clip the norms of initial samples into $(0, 3000)$ for CIFAR-10, $(0, 6000)$ for CelebA 64^2 and $(0, 30000)$ for LSUN bedroom. We adopt the DDPM++ and DDPM++ deep architectures [34] as our backbones. We add the scalar z (resp. predicted direction on z) as input (resp. output) to accommodate the additional dimension. We take the same set of hyper-parameters, such as batch size, learning rate and training iterations from [34]. We provide more training details in Appendix B.5.2, and discuss how to set these hyper-parameters for general datasets in B.5.2 and B.5.2.

Baselines We compare PFGM to modern autoregressive model [15], GAN [48], [200], normalizing flow [16] and EBM [21]. We also compare with variants of score-based models such as DDIM [54] and current state-of-the-art SDE/ODE methods [34]. We denote the methods that use forward-time SDEs in [34] such as Variance Exploding (**VE**) SDE/Variance Preserving (**VP**) SDE/ sub-Variance Preserving (**sub-VP**), and the corresponding backward SDE/ODE, as **A-B**, where A $\in$ {VE, VP, sub-VP} and B $\in$ {SDE, ODE}. We follow the model selection protocol in [34], which selects the checkpoint with the smallest FID score over the course of training every 50k iterations.

Numerical Solvers The backward ODE (Equation 7.6) is compatible with any general purpose ODE solver. In our experiments, the default solver of ODEs is the black box solver in the Scipy library [199] with the RK45 [53] method (**RK45**), unless otherwise specified. For VE/VP/subVP-SDEs, we use the predictor-corrector (**PC**) sampler introduced in [34]. For VP/sub-VP-SDEs, we apply the predictor-only sampler, because its performance is on par with the PC sampler while requiring half computation.

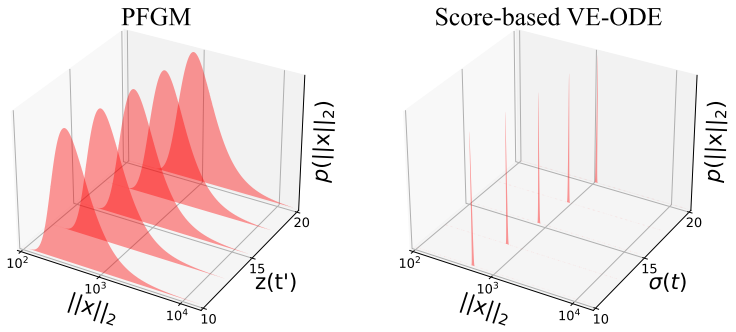


Figure 7.3: Sample norm distributions with varying time variables (σ for VE-ODE and z for PFGM)

Results For quantitative evaluation on CIFAR-10, we report the Inception [55] (higher is better) and FID [41] scores (lower is better) in Table 7.1. We also include our preliminary experimental results on a weaker architecture NCSNv2 [49] in Appendix B.5.3. We measure the inference speed by the average NFE (number of function evaluation). We also explicitly indicate which methods belong to the invertible flow family.

Our main findings are: **(1) PFGM achieves the best Inception scores and FID scores among the normalizing flow models.** Specifically, PFGM obtains an Inception score of 9.68 and a FID score of 2.48 using the DDPM++ deep architecture. To our knowledge, these are the highest FID and Inception scores by flow models on CIFAR-10. **(2) PFGM achieves a** $10\times \sim 20\times$ **faster inference speed than the SDE methods using similar architectures while retaining comparable sample quality.** As shown in Table 7.1, PFGM requires NFEs of 110, whereas the SDE methods typically use $1000 \sim 2000$ inference steps. PFGM outperforms all the baselines on DDPM++ in all metrics. In addition, PFGM generally samples faster than other ODE baselines with the same RK45 solver. **(3) The backward ODE in PFGM is compatible with architectures with varying capacities.** PFGM consistently outperforms other ODE baselines on DDPM++ (Table 7.1) or NCSNv2 (Appendix B.5.3) backbones. **(4) PFGM shows scalability to higher resolution datasets.** In Appendix B.5.3, we show that PFGM are capable of scale-up to LSUN bedroom 256×256. In particular, PFGM has comparable performance with VE-SDE with $15\times$ fewer NFE.

In Figure 7.4, we visualize the uncurated samples from PFGM on CIFAR-10, CelebA

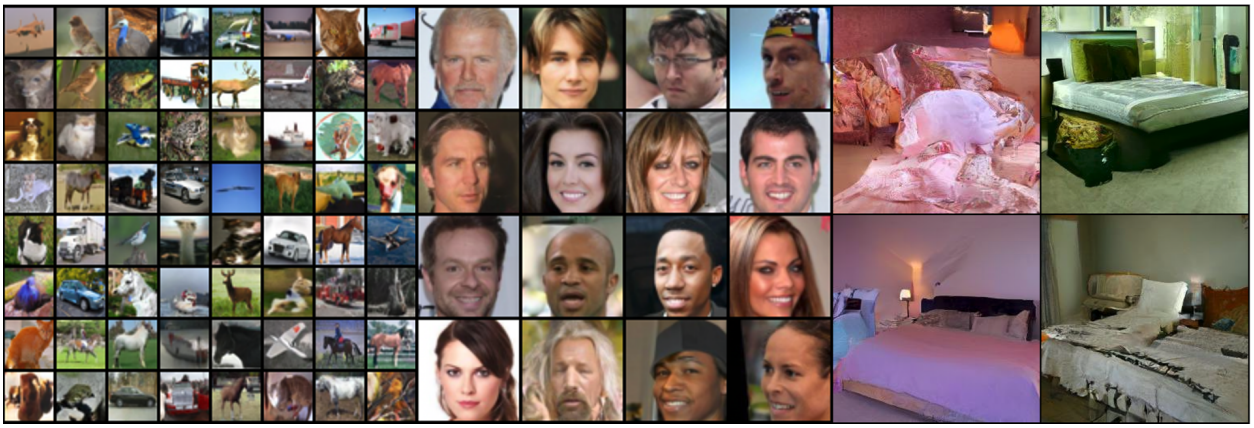


Figure 7.4: Uncurated samples on datasets of increasing resolution. From left to right: CIFAR-10 32×32, CelebA 64×64 and LSUN bedroom 256×256.

64×64 and LSUN bedroom 256×256. We provide more samples in Appendix B.5.4.

Failure of VE/VP-ODEs on NCSNv2 Architecture

Table 7.2: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE). All the methods below the *NCSNv2 backbone* separator use the NCSNv2 [49] network architecture as the backbone.

	Inception ↑	FID ↓	NFE ↓
PixelCNN [15]	4.60	65.93	1024
IGEBM [21]	6.02	40.58	60
WGAN-GP [203]	$7.86 \pm .07$	36.4	1
SNGAN [204]	$8.22 \pm .05$	21.7	1
NCSN [197]	$\mathbf{8.87 \pm .12}$	25.32	1001
NCSNv2 backbone			
Langevin dynamics [49]	$8.40 \pm .07$	**10.87**	1161
VE-SDE [34]	$8.23 \pm .02$	10.94	1000
VP-SDE [34]	$6.85 \pm .01$	44.05	1000
VE-ODE (Euler w/ corrector)	$8.05 \pm .03$	11.33	1000
VP-ODE (Euler w/ corrector)	$7.33 \pm .07$	37.74	1000
PFGM (Euler)	$8.00 \pm .09$	11.78	200
PFGM (RK45)	$8.30 \pm .05$	11.22	**118**

In our preliminary experiments on NCSNv2 architectures, we empirically observe that the VE/VP-ODEs have FID scores greater than 90 on CIFAR-10. In particular, VE/VP-ODEs can only generate decent samples when applying the Langevin dynamics corrector, and

Table 7.3: FID/NFE on CelebA 64×64

	FID ↓	NFE ↓
NCSN [197]	26.89	1001
NCSNv2 backbone		
Langevin dynamics [49]	10.23	2501
VE-SDE [34]	8.15	1000
VP-SDE [34]	34.52	1000
VE-ODE (Euler w/ corrector)	8.30	200
VP-ODE (Euler w/ corrector)	41.81	200
PFGM (Euler)	7.85	**100**
PFGM (RK45)	7.93	110
DDPM++ backbone		
PFGM (RK45)	**3.68**	110

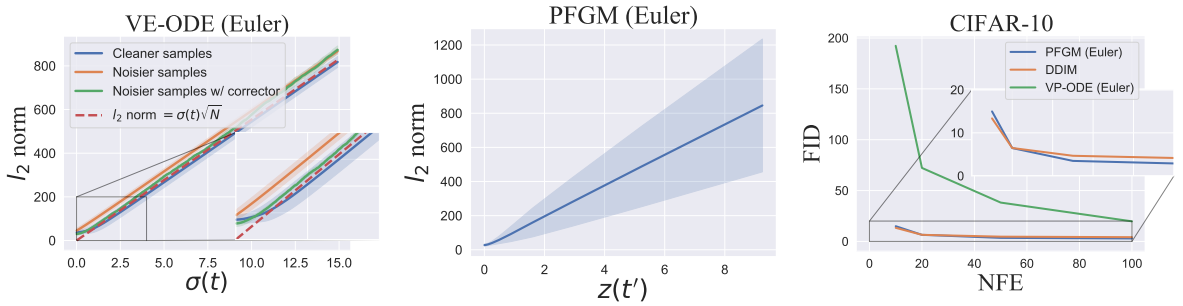
even then, their performances are still inferior to PFGM (Table B.21, Table B.22). The poor performance on NCSNv2 stands in striking contrast to their high sample quality on NCSN++/DDPM++ in [34]. **It indicates that the VE/VP-ODEs are more susceptible to estimation errors than PFGM.** We hypothesize that the strong norm-σ correlation seen during the training of score-based models causes the problem.

For score-based models, the l_2 norms of perturbed training samples and the standard deviations $\sigma(t)$ of Gaussian noises have a strong correlation, *e.g.*, l_2 norm $\approx \sigma(t)\sqrt{N}$ for large $\sigma(t)$ in VE [34]. In contrast, as shown in Figure 7.3, PFGM allocates high mass across a wide spectrum of the training sample norms. During sampling, VE/VP-ODEs could break down when the trajectories of backward ODEs deviate from the norm-$\sigma(t)$ relation to which most training samples pertain. The weaker NCSNv2 backbone incurs larger errors and thus leads to their failure. The PFGM is more resistant to estimate errors because of the greater range of training sample norms.

To further verify the hypothesis above, we split a batch of VE-ODE samples into cleaner and noisier samples according to visual quality (Figure 7.5a). In Figure 7.6a, we investigate the relation for cleaner and noisier samples during the forward Euler simulation of VE-ODE when $\sigma(t) < 15$. We can see that the trajectory of cleaner samples stays close to the norm-$\sigma(t)$

(a) VE-ODE (b) VE-ODE w/ corrector

Figure 7.5: **(a)** Samples from VE-ODE (Euler w/o corrector). We highlight the noisier images with red boxes. The rest are cleaner images. **(b)** Samples from VE-ODE (Euler w/ corrector). We mark the noisier samples after correction with green boxes.



(a) Norm-$\sigma(t)$ in VE-ODE (b) Norm-$z(t')$ in PFGM (c) FID vs. NFE on CIFAR-10

Figure 7.6: **(a)** Norm-$\sigma(t)$ relation during the backward sampling of VE-ODE (Euler). **(b)** Norm-$z(t')$ relation during the backward sampling of PFGM (Euler). The shaded areas mean the standard deviation of norms. **(c)** Number of steps versus FID score.

relation (the red dash line), whereas that of the noisier samples diverges from the relation. The Langevin dynamics corrector changes the trajectory of noisier samples to align with the relation. Figure 7.6b further shows that the anchored variable $z(t')$ and the norms in the backward ODE of PFGM are not strongly correlated, giving rise to the robustness against the imprecise estimation on NCSNv2. We defer more details to Appendix B.5.1.

Effects of Step Size in the Forward Euler Method

In order to accelerate the inference speed of ODEs, we can increase the step size (decrease the NFEs) in numerical solvers such as the forward Euler method. It also enables the trade-off

between sample quality and computational efficiency in real-world deployment. We study the effects of increasing step size on PFGM, VP-ODE and DDIM [54] using the forward Euler method, with a varying NFE ranging from 10 to 100.

In Figure 7.6c, we report the sample quality measured by FID scores on CIFAR-10. As expected, all the methods have higher FID scores when decreasing the NFE. We observe that the sample quality of PFGM degrades gracefully as we decrease the NFE. Our method shows significantly better robustness to step sizes than the VP-ODE, especially when only taking a few Euler steps. In addition, PFGM obtains better FID scores than DDIM on most NFEs except for 10 where PFGM is marginally worse. This suggests that the PFGM is a promising method for accommodating instantaneous resource availability, as high-quality samples can be generated in limited steps.

Utilities of ODE: likelihood evaluation and latent representation

Similar to the family of discrete normalizing flows [16], [17], [205] and continuous probability flow [34], the forward ODE in PFGM defines an invertible mapping between the data space and latent space with a known prior. Formally, we define the invertible forward $\mathcal{M}$ mapping by integrating the corresponding forward ODE $d(\mathbf{x}, z) = (\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}z, z)dt'$ of Equation 7.6:

$$\mathbf{x}(\log z_{\max}) = \mathcal{M}(\mathbf{x}(\log z_{\min})) \equiv \mathbf{x}(\log z_{\min}) + \int_{\log z_{\min}}^{\log z_{\max}} \mathbf{v}(\mathbf{x}(t'))_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}}(t'))_z^{-1}e^{t'}\mathrm{d}t'$$

where $\log z_{\min}/\log z_{\max}$ are the starting/terminal time in the forward ODE. The forward mapping transfers the data distribution to the prior distribution p_{prior} on the $z = z_{\max}$ hyperplane (cf. Section 7.3.3): $p_{\text{prior}}(\mathbf{x}(\log z_{\max})) = \mathcal{M}(p(\mathbf{x}(\log z_{\min})))$. The invertibility enables likelihood evaluation and creates a meaningful latent space on the $z = z_{\max}$ hyperplane. In addition, we can adapt to the computational constraints by adjusting the step size or the precision in numerical ODE solvers.

Likelihood evaluation We evaluate the data likelihood by the instantaneous change-of-variable formula [34], [191]. In Table 7.4, we report the bits/dim on the uniformly dequantized CIFAR-10 test set and compare with existing baselines that use the same setup. We observe that PFGM achieves better likelihoods than discrete normalizing flow models, even without

maximum likelihood training. Among the continuous flow models, sub-VP-ODE shows the lowest bits/dim, although its sample quality is worse than VP-ODE and PFGM (Table 7.1). The exploration of the seeming trade-off between likelihood and sample quality is left for future works.

Latent representation Since the samples are uniquely identifiable by their latents via the invertible mapping $\mathcal{M}$, PFGM further supports image manipulation using its latent representation on the $z = z_{\max}$ hyperplane. We include the results of image interpolation and the temperature scaling [16], [17], [34] to Appendix B.5.3 and Appendix B.5.3. For interpolation, it shows that we can travel along the latent space to obtain perceptually consistent interpolations between CelebA images.

Table 7.4: Bits/dim on CIFAR-10

	bits/dim ↓
RealNVP [17]	3.49
Glow [16]	3.35
Residual Flow [196]	3.28
Flow++ [205]	3.29
DDPM (L) [167]	$\leq$ 3.70*
DDPM++ backbone	
VP-ODE [34]	3.20
sub-VP-ODE [34]	**3.02**
PFGM (ours)	3.19

7.5 Conclusion

We present a new deep generative model by solving the Poisson equation whose source term is the data distribution. We estimate the normalized gradient field of the solution in an augmented space with an additional dimension. For sampling, we devise a backward ODE that exponential decays on the physically meaningful additional dimension. Empirically, our approach has currently best performance over other normalizing flow baselines, and achieving 10× to 20× acceleration over the stochastic methods. Our backward ODE shows greater stability against errors than popular ODE-based methods, and enables efficient adaptive sampling. We further demonstrate the utilities of the forward ODE on likelihood evaluation and image interpolation. Future directions include improving the normalization of Poisson fields. More principled approaches can be used to get around the divergent near-field behavior. For example, we may exploit renormalization, a useful tool in physics, to make the Poisson field well-behaved in near fields.

Chapter 8

An Extended View of Electrostatics in Higher-dimensional Space

In Chapter 7, we discuss a new type of generative models — PFGM — arising from electrostatics theory. In this chapter, we will introduce its extended version *PFGM++* that unifies diffusion models and Poisson Flow Generative Models (PFGM). These models realize generative trajectories for N dimensional data by embedding paths in $N{+}D$ dimensional space while still controlling the progression with a simple scalar norm of the D additional variables. The new models reduce to PFGM when $D{=}1$ and to diffusion models when $D{\to}\infty$. The flexibility of choosing D allows us to trade off robustness against rigidity as increasing D results in more concentrated coupling between the data and the additional variable norms. We dispense with the biased large batch field targets used in PFGM and instead provide an unbiased perturbation-based objective similar to diffusion models. To explore different choices of D, we provide a direct alignment method for transferring well-tuned hyperparameters from diffusion models ($D{\to}\infty$) to any finite D values. Our experiments show that models with finite D can be superior to previous state-of-the-art diffusion models on CIFAR-10/FFHQ $64{\times}64$ datasets/LSUN Churches $256{\times}256$, with median Ds. Furthermore, we demonstrate that models with smaller D exhibit improved robustness against modeling errors.

This chapter was previously published as [31].

8.1 Introduction

Physics continues to inspire new deep generative models such as *diffusion models* [10], [27], [34], [167] based on thermodynamics [206] or *Poisson flow generative models* (PFGM) [30] derived from electrostatics in previous chapter. The associated generative processes involve iteratively denoising samples by following physically meaningful trajectories. Diffusion models learn a noise-level dependent score function so as to reverse the effects of forward diffusion, progressively reducing the noise level σ along the generation trajectory. PFGMs in turn augment N-dimensional data points with an extra dimension and evolve samples drawn from a uniform distribution over a large $N{+}1$-dimensional hemisphere back to the $z{=}0$ hyperplane where the clean data (as charges) reside by tracing learned electric field lines.

In this chapter, we introduce a broader family of physics-inspired generative models that we call ***PFGM++***. These models extend the electrostatic view into higher dimensions through multi-dimensional $\mathbf{z} \in \mathbb{R}^D$ augmentations. When interpreting N-dimensional data points $\mathbf{x}$ as positive charges, the electric field lines define a surjection from a uniform distribution on an infinite $N{+}D$-dimensional hemisphere to the data distribution located on the $\mathbf{z}{=}\mathbf{0}$ hyperplane. We can therefore draw generative samples by following the electric field lines, evolving points from the hemisphere back to the $\mathbf{z}{=}\mathbf{0}$ hyperplane. We leverage the symmetry of z to reduce the vector to a scalar $\|\mathbf{z}\|_2 = r$, simplifying the sampling process. The use of symmetry turns the aforementioned surjection into a bijection between an easy-to-sample prior on a large $r = r_{\max}$ hyper-cylinder to the data distribution. The symmetry reduction also permits D to take any positive values, including reals. We derive a new perturbation-based training objective akin to denoising score matching [38] that avoids the need to use large batches to construct electric field line targets in PFGM. The perturbation-based objective is more efficient, unbiased, and compatible with paired sample training of conditional generation models.

The models in the new family differ based on their augmentation dimension D which is now a hyper-parameter. By setting $D{=}1$ we obtain PFGM while $D{\to}\infty$ leads to diffusion models. We establish $D{\to}\infty$ equivalence with popular diffusion models [27], [34] both in terms of their training objectives as well as their inferential processes. We demonstrate that

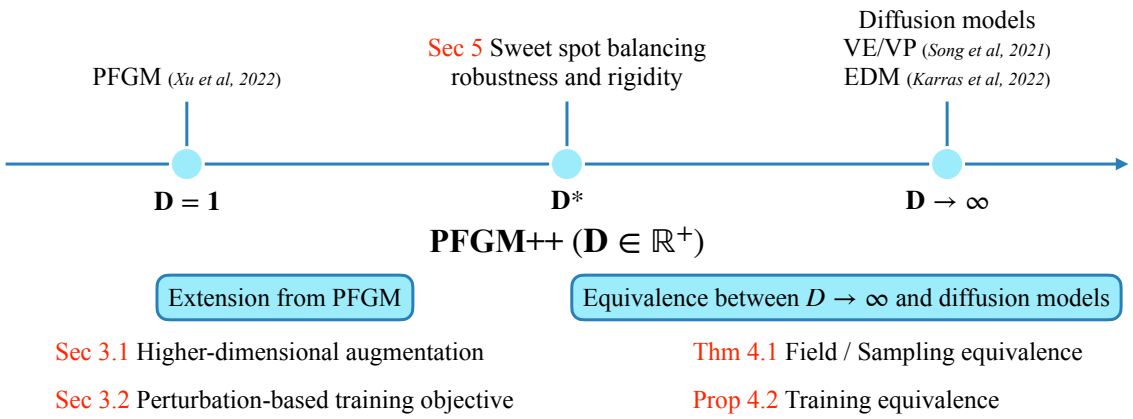


Figure 8.1: Overview of paper contributions and structure. PFGM++ unify PFGM and diffusion models, as well as the potential to combine their strengths (robustness and rigidity).

the hyper-parameter D controls the balance between robustness and rigidity: using a small D widens the distribution of noisy training sample norms in comparison to the norm of the augmented variables, leading to a more robust generative process. However, small D also leads to a heavy-tailed problem of training samples, making the training process more challenging (neural networks cannot rigidly predict the fields correctly). Neither D=1 nor $D\to\infty$ offers an ideal balance between being insensitive to missteps (robustness) and allowing effective learning (rigidity). Instead, we adjust D in response to different architectures and tasks. To facilitate quickly finding the best D we provide an alignment method to directly transfer other hyperparameters across different choices of D.

Experimentally, we show that some models with finite D outperform the previous state-of-the-art diffusion models ($D\to\infty$), *i.e.*, EDM [27], on image generation tasks. In particular, intermediate D=2048/128/131072 achieve the best performance among other choices of D ranging from 64 to ∞, with min FID scores of 1.91/2.43/6.52 on CIFAR-10/ FFHQ 64×64/LSUN Churches 256×256 datasets in unconditional generation, using 35/79/99 NFE. In class-conditional generation, D=2048 achieves new state-of-the-art FID of 1.74 on CIFAR-10. We further verify that in general, decreasing D leads to improved robustness against a variety of sources of errors, *i.e.*, controlled noise injection, large sampling step sizes and post-training quantization.

Our contributions are summarized as follows: **(1)** We propose PFGM++ as a new family of generative models based on expanding augmented dimensions and show that symmetries

involved enable us to define generative paths simply based on the scalar norm of the augmented variables (Section 8.2.1); **(2)** We propose a perturbation-based objective to dispense with any biased large batch derived electric field targets, allowing unbiased training (Section 8.2.2); **(3)** We prove that the score field and the training objective of diffusion models arise in the limit $D \to \infty$ (Section 8.3); **(4)** We demonstrate the trade-off between robustness and rigidity by varying D (Section 8.4). *We also detail the hyperparameter transfer procedures from EDM/DDPM ($D \to \infty$) to finite Ds in Appendix B.6.1*; **(5)** We empirically show that models with finite D achieve superior performance to diffusion models while exhibiting improved robustness (Section 8.5).

8.2 PFGM++: Augmenting the Data with Arbitrary Dimension D

In this section, we present our new family of generative models PFGM++, generalizing PFGM [30] in terms of the augmented space dimensionality. We show that the electric fields in $N{+}D$-dimensional space with $D \in \mathbb{Z}^+$ still constitute a valid generative model (Sec 8.2.1). Furthermore, we show that the additional D-dimensional augmented variable can be condensed into their scalar norm due to the inherent symmetry of the electric field. To improve the training process, we propose an efficient perturbation-based objective for training PFGM++ (Sec 8.2.2) without relying on the large batch approximation in the original PFGM.

8.2.1 Electric Field in Higher-Dimensional Space

While PFGM [30] consider the electric field in a $N{+}1$-dimensional augmented space, we augment the data $\mathbf{x}$ with D-dimensional variables $\mathbf{z} = (z_1, \ldots, z_D)$, *i.e.*, $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{z})$ and $D \in \mathbb{Z}^+$. Similar to the $N{+}1$-dimensional electric field (Equation 7.4), the electric field at the augmented data $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{N+D}$ is:

$$\mathbf{E}(\tilde{\mathbf{x}}) = \frac{1}{S_{N+D-1}(1)} \int \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{y}}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) \mathrm{d}\mathbf{y} \tag{8.1}$$

Analogous to the theoretical results presented in PFGM, with the electric field as the drift term, the ODE $d\tilde{\mathbf{x}}{=}\mathbf{E}(\tilde{\mathbf{x}})\mathrm{d}t$ defines a surjection from a uniform distribution on an infinite $N{+}D$-*dim* hemisphere (the measure we used on hemisphere is defined as the "surface area" of the hypersphere, *i.e.*, $\bar{r}^{N+D-1}d\Omega$, where $d\Omega$ is the solid angle on the $N{+}D{-}1$-dimensional sphere with radius $\bar{r}$) and the data distribution on the N-*dim* $\mathbf{z}{=}\mathbf{0}$ hyperplane. However, the mapping has $SO(D)$ symmetry on the surface of D-dim cylinder $\sum_{i=1}^{D} z_i^2 = r^2$ for any positive r. We provide an illustrative example at the bottom of Figure 8.2 ($D{=}2, N{=}1$), where the electric flux emitted from a line segment (red) has rotational symmetry through the ring area (blue) on the $z_1^2 + z_2^2 = r^2$ cylinder. Hence, instead of modeling the individual behavior of each z_i, it suffices to track the norm of augmented variables — $r(\tilde{\mathbf{x}}) = \|\mathbf{z}\|_2$ — due to symmetry. Specifically, note that $\mathrm{d}z_i = \mathbf{E}(\tilde{\mathbf{x}})_{z_i}\mathrm{d}t$, and the time derivative of r is

$$\begin{aligned}\frac{\mathrm{d}r}{\mathrm{d}t} &= \sum_{i=1}^{D} \frac{z_i}{r}\frac{\mathrm{d}z_i}{\mathrm{d}t} = \int \frac{\sum_{i=1}^{D} z_i^2}{S_{N+D-1}(1) r \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y})\mathrm{d}\mathbf{y} \\ &= \frac{1}{S_{N+D-1}(1)} \int \frac{r}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y})\mathrm{d}\mathbf{y}\end{aligned}$$

Henceforth we replace the notation for augmented data with $\tilde{\mathbf{x}} = (\mathbf{x}, r)$ for simplicity. After the symmetry reduction, the field to be modeled has a similar form as Equation 8.1 except that the last D sub-components $\{\mathbf{E}(\tilde{\mathbf{x}})_{z_i}\}_{i=1}^{D}$ are condensed into a scalar $E(\tilde{\mathbf{x}})_r = \frac{1}{S_{N+D-1}(1)} \int \frac{r}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y})\mathrm{d}\mathbf{y}$. Therefore, we can use the physically meaningful r as the anchor variable in the ODE $\mathrm{d}\mathbf{x}/\mathrm{d}r$ by change-of-variable:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}r} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}r} = \frac{\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}}{E(\tilde{\mathbf{x}})_r} \tag{8.2}$$

Indeed, the ODE $\mathrm{d}\mathbf{x}/\mathrm{d}r$ turns the aforementioned surjection into a bijection between an easy-to-sample prior distribution on the $r{=}r_{\max}$ hyper-cylinder [1] and the data distribution on $r{=}0$ (*i.e.*, $\mathbf{z}{=}\mathbf{0}$) hyperplane. The following theorem states the observation formally:

Theorem 7. *Assume the data distribution $p \in \mathcal{C}^1$ and p has compact support. As $r_{max}{\to}\infty$, for $D \in \mathbb{R}^+$, the ODE $\mathrm{d}\mathbf{x}/\mathrm{d}r = \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r$ defines a bijection between $\lim_{r_{max}\to\infty} p_{r_{max}}(\mathbf{x}) \propto$*

[1] The hyper-cylinder here is consistent with the hemisphere in PFGM [30], because hyper-cylinders degrade to hyper-planes for $D = 1$, which are in turn isomorphic to hemispheres.

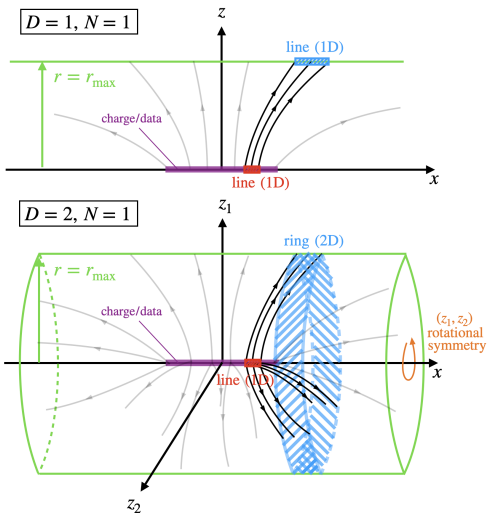


Figure 8.2: The augmented dimension D affects electric field lines (gray), which connect charge/data on a line (purple) to latent space (green). When $D = 1$ (top) or $D = 2$ (bottom), electric field lines map the same red line segment to a blue line segment or onto a blue ring, respectively. The mapping defined by electric lines has $SO(2)$ symmetry on the surface of $z_1^2 + z_2^2 = r^2$ cylinder.

$\lim_{r_{max}\to\infty} r_{max}^D/(\|\mathbf{x}\|_2^2 + r_{max}^2)^{\frac{N+D}{2}}$ *when* $r = r_{max}$ *and the data distribution* p *when* $r = 0$.

Proof sketch. The r-dependent intermediate distribution of the ODE (Equation 8.2) is $p_r(\mathbf{x}) \propto \int r^D/\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} p(\mathbf{y}) \mathrm{d}\mathbf{y}$, which satisfies initial/terminal conditions, *i.e.*, $p_{r=0}{=}p$, $\lim_{r_{\max}\to\infty} p_{r_{\max}} \propto \lim_{r_{\max}\to\infty} r_{\max}^D/(\|\mathbf{x}\|_2^2 + r_{\max}^2)^{\frac{N+D}{2}}$, as well as the continuity equation of the ODE, *i.e.*, $\partial_r p_r + \nabla_{\mathbf{x}} \cdot (p_r \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r) = 0$. □

We defer the formal proof to Appendix A.5.1. Note that in the theorem we further extend the domain of D from positive integers to positive real numbers. In practice, the starting condition of the ODE is some sufficiently large $r_{\max}$ such that $p_{r_{\max}}(\mathbf{x}) \approx r_{\max}^D/(\|\mathbf{x}\|_2^2 + r_{\max}^2)^{\frac{N+D}{2}}$. The terminal condition is $r{=}\,0$, which represents the generated samples reaching the data support. The proposed PFGM++ framework thus permits choosing arbitrary D, including $D = 1$ which recovers the original PFGM formulation. Interestingly, we will also show that when $D{\to}\infty$, PFGM++ recover the diffusion models (Sec 8.3). In addition, as discussed in Sec 8.4, the choice of D is important, since it controls two properties of the associated electric field, *i.e.*, robustness and rigidity, which affect the sampling performance.

8.2.2 Efficient Training with Perturbation Kernel

Although the training process in PFGM can be directly applied to PFGM++, we propose a more efficient training objective to dispense with the large batch in PFGM. The objective from PFGM paper [30] requires sampling a large batch of data $\{\mathbf{y}_i\}_{i=1}^n \sim p^n(\mathbf{y})$ in each training step to approximate the integral in the electric field (Equation 8.1):

$$\mathbb{E}_{\{\mathbf{y}_i\}_{i=1}^n \sim p^n(\mathbf{y})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \tilde{p}_{\text{train}}(\tilde{\mathbf{x}}|\tilde{\mathbf{y}}_1=(\mathbf{y}_1,\mathbf{0}))} \left[\left\| f_\theta(\tilde{\mathbf{x}}) - \frac{\sum_{i=0}^{n-1} \frac{\tilde{\mathbf{x}}-\tilde{\mathbf{y}}_i}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}_i\|^{N+D}}}{\left\| \sum_{i=0}^{n-1} \frac{\tilde{\mathbf{x}}-\tilde{\mathbf{y}}_i}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}_i\|^{N+D}} \right\|_2 + \gamma} \right\|_2^2 \right]$$

where $\tilde{p}_{\text{train}}$ is heuristically designed to cover the regions that the backward ODE traverses and γ in the denominator is a tiny positive number to prevent numerical issues. This objective has several obvious drawbacks: *(1)* The large batch incurs additional overheads; *(2)* Its minimizer is a biased estimator of the electric field (Equation 8.1); *(3)* The large batch is incompatible with typical paired sample training of conditional generation, where each condition is paired with only one sample, such as text-to-image [153], [160] and text-to-3D generation [207], [208].

To remedy these issues, we propose a perturbation-based objective without the need for the large batch, while achieving an unbiased minimizer and enabling paired sample training of conditional generation. Inspired by denoising score-matching [38], we design the perturbation kernel to guarantee that the minimizer in the following square loss objective matches the ground-truth electric field in Equation 8.1:

$$\mathbb{E}_{r \sim p(r)} \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{p_r(\mathbf{x}|\mathbf{y})} \left[\|f_\theta(\tilde{\mathbf{x}}) - (\tilde{\mathbf{x}} - \tilde{\mathbf{y}})\|_2^2 \right] \tag{8.3}$$

where $r \in (0, \infty)$, $p(r)$ is the training distribution over r, $p_r(\mathbf{x}|\mathbf{y})$ is the perturbation kernel and $\tilde{\mathbf{y}}=(\mathbf{y}, 0)/\tilde{\mathbf{x}}=(\mathbf{x}, r)$ are the clean/perturbed augmented data. The minimizer of Equation 8.3 is $f_\theta^*(\tilde{\mathbf{x}}) \propto \int p_r(\mathbf{x}|\mathbf{y})(\tilde{\mathbf{x}} - \tilde{\mathbf{y}})p(\mathbf{y})\mathrm{d}\mathbf{y}$, which matches the direction of electric field $\mathbf{E}(\tilde{\mathbf{x}}) \propto \int (\tilde{\mathbf{x}} - \tilde{\mathbf{y}})/\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} p(\mathbf{y})\mathrm{d}\mathbf{y}$ when setting the perturbation kernel to $p_r(\mathbf{x}) \propto 1/(\|\mathbf{x}\|_2^2 + r^2)^{\frac{N+D}{2}}$. Denoting the r-dependent intermediate marginal distribution as

$p_r(\mathbf{x}) = \int p_r(\mathbf{x}|\mathbf{y})p(\mathbf{y})d\mathbf{y}$, the following proposition states that the choice of $p_r(\cdot|\mathbf{y})$ guarantee that the minimizer of the square loss to match the direction of the electric field:

Proposition 1. *With perturbation kernel* $p_r(\mathbf{x}|\mathbf{y}) \propto 1/(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}$, *for* $\forall \mathbf{x} \in \mathbb{R}^N, r > 0$, *the minimizer* $f_\theta^*(\tilde{\mathbf{x}})$ *in the PFGM++ objective (Equation 8.3) matches the direction of electric field* $\mathbf{E}(\tilde{\mathbf{x}})$ *in Equation 8.1. Specifically,* $f_\theta^*(\tilde{\mathbf{x}}) \propto (S_{N+D-1}(1)/p_r(\mathbf{x}))\mathbf{E}(\tilde{\mathbf{x}})$.

We defer the proof to Appendix A.5.2. The proposition indicates that the minimizer $f_\theta^*(\tilde{\mathbf{x}})$ can match the direction of $\mathbf{E}(\tilde{\mathbf{x}})$ with sufficient data and model capacity. The current training target in Equation 8.3 is the directional vector between the clean data $\tilde{\mathbf{y}}$ and perturbed data $\tilde{\mathbf{x}}$ akin to denoising score-matching for diffusion models [27], [34]. In addition, the new objective allows for conditional generations under a one-sample-per-condition setup. Since the perturbation kernel is isotropic, we can decompose $p_r(\cdot|\mathbf{y})$ in hyperspherical coordinates to $\mathcal{U}_\psi(\psi)p_r(R)$, where $\mathcal{U}_\psi$ is the uniform distribution over the angle component and the distribution of the perturbed radius $R = \|\mathbf{x}-\mathbf{y}\|_2$ is

$$p_r(R) \propto \frac{R^{N-1}}{(R^2+r^2)^{\frac{N+D}{2}}}$$

We defer the practical sampling procedure of the perturbation kernel to Appendix B.6.2. The mean of the r-dependent radius distribution $p_r(R)$ is around $r\sqrt{N/D}$. Hence we explicitly normalize the target in Equation 8.3 by $r/\sqrt{D}$, to keep the norm of the target around the constant $\sqrt{N}$, similar to diffusion models [34]. In addition, we drop the last dimension of the target because it is a constant — $(\tilde{\mathbf{x}}-\tilde{\mathbf{y}})_r/(r/\sqrt{D}) = \sqrt{D}$. Together, the new objective is

$$\mathbb{E}_{r\sim p(r)}\mathbb{E}_{p(\tilde{\mathbf{y}})}\mathbb{E}_{p_r(\tilde{\mathbf{x}}|\tilde{\mathbf{y}})}\left[\left\| f_\theta(\tilde{\mathbf{x}}) - \frac{\mathbf{x}-\mathbf{y}}{r/\sqrt{D}}\right\|_2^2\right] \tag{8.4}$$

which is essentially a rescaled version of Equation 8.3. After training the neural network through objective Equation 8.4, we can use the ODE (Equation 8.2) anchored by r to generate samples, *i.e.*, $\mathrm{d}\mathbf{x}/\mathrm{d}r = \mathbf{E}(\tilde{\mathbf{x}})_\mathbf{x}/E(\tilde{\mathbf{x}})_r = f_\theta(\tilde{\mathbf{x}})/\sqrt{D}$, starting from the prior distribution $p_{r_{\max}}$. We would like to highlight that PFGM++ maintain the same memory requirements as PFGM ($D=1$) or diffusion models ($D=\infty$) during both training and sampling. This is achieved by condensing the high-dimensional augmented variable $\mathbf{z}$ into the scalar r.

8.3 Diffusion Models as $D \to \infty$ Special Cases

Diffusion models generate samples by simulating ODE/SDE involving the score function $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$ at different intermediate distributions p_σ [27], [34], where σ is the standard deviation of the Gaussian kernel. In this section, we show that both sampling and training schemes in diffusion models are equivalent to those in $D \to \infty$ case under the PFGM++ framework. To begin with, we show that the electric field (Equation 8.1) in PFGM++ has the same direction as the score function when D tends to infinity, and their sampling processes are also identical.

Theorem 8. *Assume the data distribution $p \in \mathcal{C}^1$. Consider taking the limit $D \to \infty$ while holding $\sigma = r/\sqrt{D}$ fixed. Then, for all $\mathbf{x}$,*

$$\lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \left\| -\frac{\sqrt{D}}{E(\tilde{\mathbf{x}})_r}\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}} - \sigma\nabla_{\mathbf{x}} \log p_{\sigma=r/\sqrt{D}}(\mathbf{x}) \right\|_2 = 0$$

where $\mathbf{E}(\tilde{\mathbf{x}} = (\mathbf{x}, r))_{\mathbf{x}}$ is given in Equation 8.1. Further, given the same initial point, the trajectory of the PFGM++ ODE ($\mathrm{d}\mathbf{x}/\mathrm{d}r = \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r$) matches the diffusion ODE [27] ($\mathrm{d}\mathbf{x}/\mathrm{d}t = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p_{\sigma(t)}(\mathbf{x})$) in the same limit.

Proof sketch. By re-expressing the $\mathbf{x}$ component $\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}$ in the electric field and the score $\nabla_{\mathbf{x}} \log p_\sigma$ in diffusion models, the proof boils down to show that $\lim_{D\to\infty, r=\sigma\sqrt{D}} p_r(\mathbf{x}|\mathbf{y}) \propto \exp(-\|\mathbf{x}-\mathbf{y}\|_2^2/2\sigma^2)$ for $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{N+D}$:

$$\begin{aligned} & \lim_{D\to\infty, r=\sigma\sqrt{D}} \frac{1}{\left(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2\right)^{\frac{N+D}{2}}} \\ \propto & \lim_{D\to\infty, r=\sigma\sqrt{D}} e^{-\frac{(N+D)}{2}\ln(1+\frac{\|\mathbf{x}-\mathbf{y}\|^2}{r^2})} \\ = & \lim_{D\to\infty, r=\sigma\sqrt{D}} e^{-\frac{(N+D)\|\mathbf{x}-\mathbf{y}\|_2^2}{2r^2}} = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}} \end{aligned} \tag{8.5}$$

The equivalence of trajectories can be proven by change-of-variable $\mathrm{d}\sigma = \mathrm{d}r/\sqrt{D}$. Their prior distributions are also the same since $\lim_{D\to\infty} p_{r_{\max}=\sigma_{\max}\sqrt{D}}(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \boldsymbol{I})$. □

We defer the formal proof to Appendix A.5.3. Since $\|\mathbf{x}-\mathbf{y}\|_2^2/r^2 \approx N/D$ when $\mathbf{x} \sim$

$p_r(\mathbf{x}), \mathbf{y} \sim p(\mathbf{y})$, Equation 8.5 approximately holds under the condition $D \gg N$. Remarkably, the theorem states that PFGM++ recover the field and sampling of previous popular diffusion models, such as VE/VP [49] and EDM [27], by choosing the appropriate schedule and scale function in [27].

In addition to the field and sampling equivalence, we demonstrate that the proposed PFGM++ objective (Equation 8.4) with perturbation kernel $p_r(\mathbf{x}|\mathbf{y}) \propto 1/(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}$ recovers the weighted sum of the denoising score matching objective [38] for training continuous diffusion model [27], [34] when $D \to \infty$. All previous objectives for training diffusion models can be subsumed in the following form [27], under different parameterizations of the neural networks f_θ:

$$\mathbb{E}_{\sigma \sim p(\sigma)} \lambda(\sigma) \mathbb{E}_{p(\mathbf{y})} \mathbb{E}_{p_\sigma(\mathbf{x}|\mathbf{y})} \left[\left\| f_\theta(\mathbf{x}, \sigma) - \frac{\mathbf{x}-\mathbf{y}}{\sigma} \right\|_2^2 \right] \tag{8.6}$$

where $p_\sigma(\mathbf{x}|\mathbf{y}) \propto \exp(-\|\mathbf{x}-\mathbf{y}\|_2^2 / 2\sigma^2)$. The objective of the diffusion models resembles the one of PFGM++ (Equation 8.4). Indeed, we show that when $D \to \infty$, the minimizer of the proposed PFGM++ objective at $\tilde{\mathbf{x}} = (\mathbf{x}, r)$ is $f_\theta^*(\mathbf{x}, r = \sigma\sqrt{D}) = -\sigma \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$, the same as the minimizer of diffusion objective at the noise level $\sigma = r/\sqrt{D}$.

Proposition 2. *When $r = \sigma\sqrt{D}, D \to \infty$, the minimizer in the PFGM++ objective (Equation 8.4) is **equaivalent to** the minimizer in the weighted sum of denoising score matching objective (Equation 8.6)*

We defer the proof to Appendix A.5.4. The proposition states that the training objective of diffusion models is essentially the same as PFGM++'s when $D \to \infty$. Combined with Theorem 8, PFGM++ thus recover both the training and sampling processes of diffusion models when $D \to \infty$.

Transfer hyperparameters to finite Ds The training hyperparameters of diffusion models ($D \to \infty$) have been highly optimized through a series of works [27], [34], [167]. It motivates us to transfer hyperparameters, such as $r_{\max}$ and $p(r)$, of $D \to \infty$ to finite Ds. Here we present an alignment method that enables a "zero-shot" transfer of hyperparameters across different Ds. Our alignment method is inspired by the concept of phases in [23],

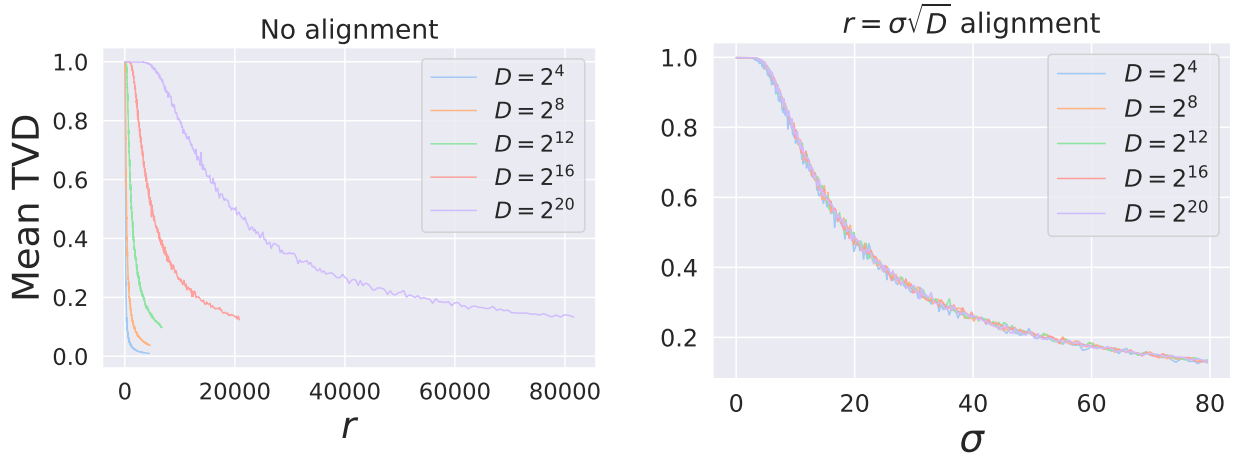
which demonstrates that the score field in the forward process of diffusion models can be decomposed into three successive phases. As we move from the near field (Phase 1) to the far field (Phase 3), the perturbed data become influenced by more modes in the data distribution. The authors show that the posterior $p_{0|\sigma}$ serves as a phase indicator, as it gradually evolves from a delta distribution to a uniform distribution when transitioning from Phase 1 to Phase 3.

We aim to align the phases for two distinct $D_1, D_2 > 0$. In Appendix B.6.1, we demonstrate that when $r \propto \sqrt{D}$, the phase of the intermediate distribution p_r is approximately invariant to all $D > 0$ (including $D{\to}\infty$). In other words, when $r_{D_1}/r_{D_2} = \sqrt{D_1/D_2}$, the phases of $p_{r_{D_1}}$ and $p_{r_{D_2}}$, under D_1 and D_2 respectively, are roughly aligned. Theorem 8 further shows that the relation $r{=}\sigma\sqrt{D}$ makes PFGM++ equivalent to diffusion models when $D{\to}\infty$. Together, the $r{=}\sigma\sqrt{D}$ formula aligns the phases of p_σ in diffusion models and $p_{r=\sigma\sqrt{D}}$ in PFGM++ for $\forall D{>}0$. Such alignment enables directly transferring the finely tuned hyperparameters $\sigma_{\max}, p(\sigma)$ in previous state-of-the-art diffusion models [27] with $r_{\max}{=}\sigma_{\max}\sqrt{D}, p(r){=}p(\sigma{=}r/\sqrt{D})/\sqrt{D}$. *We put the practical hyperparameter transfer procedures in Appendix B.6.1.*

We empirically verify the alignment formula on the CIFAR-10 [157]. [23] shows that the posterior $p_{0|r}(\mathbf{y}|\mathbf{x}) \propto p_r(\mathbf{x}|\mathbf{y})p(\mathbf{y})$ gradually grows towards a uniform distribution from the near to the far field. As a result, the mean total variational distance (TVD) between a uniform distribution and the posterior serves as an indicator of the phase of p_r: $\mathbb{E}_{p_r(\mathbf{x})}\text{TVD}\left(U(\cdot) \parallel p_{0|r}(\cdot|\mathbf{x})\right)$. Figure 8.3 reports the mean TVD before and after the $r{=}\sigma\sqrt{D}$ alignment. We observe that the mean TVDs of a wide range of Ds take similar values after the alignment, suggesting that the phases of $p_{r=\sigma\sqrt{D}}$ are roughly aligned for different Ds.

8.4 Balancing Robustness and Rigidity

In this section, we first delve into the behaviors of PFGM++ with different Ds (Sec 8.4.1) based on the alignment formula. Then we demonstrate how to leverage D to balance the robustness and rigidity of models (Sec 8.4.2). We defer all experimental details in this section to Appendix B.6.2.

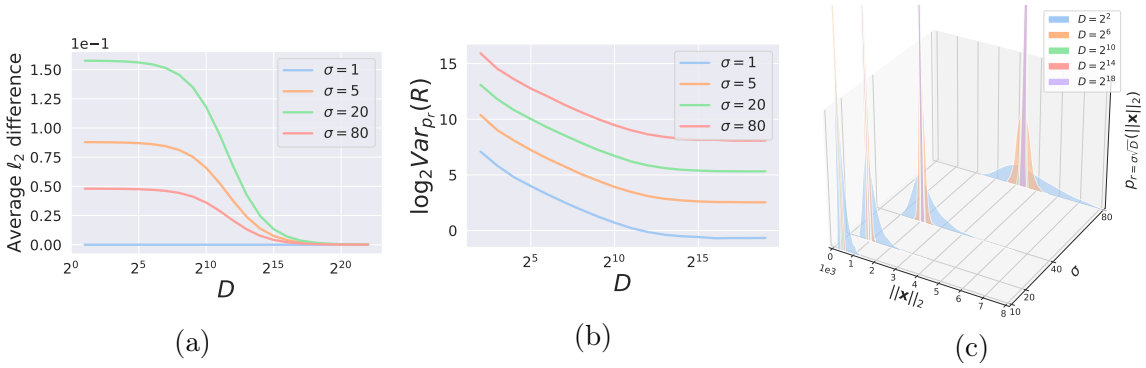


(a) No alignment

(b) $r = \sigma\sqrt{D}$ alignment

Figure 8.3: Mean TVD between the posterior $p_{0|r}(\cdot|\mathbf{x})$ ($\mathbf{x}$ is perturbed sample) and the uniform prior, w/o **(a)** and w/ **(b)** the phase alignment ($r = \sigma\sqrt{D}$).

8.4.1 Behavior of Perturbation Kernel When Varying D



(a)

(b)

(c)

Figure 8.4: **(a)** Average ℓ_2 difference between scaled electric field and score function, versus D. **(b)** Log-variance of radius distribution versus D. **(c)** Density of radius distributions $p_{r=\sigma\sqrt{D}}(R)$ with varying σ and D.

According to Theorem 8, when $D \to \infty$, the field in PFGM++ has the same direction as the score function, *i.e.*, $\sqrt{D}\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r = \sigma\nabla_{\mathbf{x}} \log p_{\sigma=r/\sqrt{D}}(\mathbf{x})$. In addition to the theoretical analysis, we provide further empirical study to characterize the convergence towards diffusion models as D increases. Figure 8.4a reports the average ℓ_2 difference between the two quantities, *i.e.*, $\mathbb{E}_{p_\sigma(\mathbf{x})}[\| - \sqrt{D}\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r - \sigma\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\|_2]$ with $r = \sigma\sqrt{D}$. We observe that the difference monotonically decreases as a function of D, and converges to 0 as predicted by theory. For $\sigma = 1$, the distance remains 0 since the empirical posterior $p_{0|r}$ (a categorical

distribution) concentrates around a single example for all D. This is because the distance between the perturbed data $\mathbf{x}$ and a specific data point is much smaller than the distance between $\mathbf{x}$ and any other data points in the training set. The posterior will gradually allocate all the mass on a certain datapoint for all D when decreasing σ.

Next, we examine the behavior of the perturbation kernel after the phase alignment. Recall that the isotropic perturbation kernel $p_r(\mathbf{x}|\mathbf{y}) \propto 1/(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}$ can be decomposed into a uniform angle component and a radius distribution $p_r(R) \propto R^{N-1}/(R^2+r^2)^{\frac{N+D}{2}}$. Figure 8.4b shows the variance of the radius distribution significantly decreases as D increases. The results imply that with relatively large r, the norm of the training sample in $p_r(\mathbf{x})$ becomes increasingly concentrated around a specific value as D increases, reaching its highest level of concentration as $D \to \infty$ (diffusion models). Figure 8.4c further shows the density of training sample norms in $p_{r=\sigma\sqrt{D}}(\mathbf{x})$ on CIFAR-10. We can see that the range of the high-mass region gradually shrinks when D increases.

8.4.2 Balancing the Trade-off by Controlling D

As noted in [30], diffusion models ($D \to \infty$) are more susceptible to estimation errors compared to PFGM ($D=1$) due to the strong correlation between σ and the training sample norm, as demonstrated in Figure 8.4c. When D and r are large, the marginal distribution $p_r(\mathbf{x})$ is approximately supported on the sphere with radius $r\sqrt{N/D}$. The backward ODE can lead to unexpected results if the sampling trajectories deviate from this norm-r relation present in training samples. This phenomenon was empirically confirmed by [30] for PFGM/diffusion models ($D=1$ and $D \to \infty$ cases) using a weaker architecture NCSNv2 [49], where PFGM was shown to be significantly more robust than diffusion models.

Smaller D, however, implies a heavy-tailed input distribution. Figure 8.4c illustrates that the examples used as the input to the neural network have a broader range of norms when D is small. In particular, when $D < 2^5$, the variance of perturbation radius can be larger than 2^{10} (Figure 8.4b). This broader input range can be challenging for any finite-capacity neural network. Although [30] introduced heuristics to bypass this issue in the $D=1$ case, *e.g.*, restricting the sampling/training regions, these heuristics also prevent the sampling process from faithfully recovering the data distribution.

Thus, we can view D as a parameter to optimize so as to balance the robustness of generation against rigidity that helps learning. Increased robustness allows practitioners to use smaller neural networks, e.g., by applying post-training quantization [209], [210]. In other words, smaller D allows for more aggressive quantization/larger sampling step sizes/smaller architectures. These can be crucial in real-world applications where computational resources and storage are limited. On the other hand, such gains need to be balanced against easier training afforded by larger values of D. The ability to optimize the balance by varying D can be therefore advantageous. We expect that there exists a sweet spot of D in the middle striking the balance, as the model robustness and rigidity go in opposite directions.

8.5 Experiments

We consider the widely used benchmarks CIFAR-10 32×32 [157], FFHQ 64×64 [211] and LSUN Churches 256 × 256 [212] for image generation. For training, we utilize the improved NCSN++/DDPM++ architectures, preconditioning techniques and hyperparameters from the state-of-the-art diffusion model EDM [27]. Specifically, we use the alignment method developed in Sec 8.3 to transfer their tuned critical hyperparameters $\sigma_{\max}, \sigma_{\min}, p(\sigma)$ in the $D\to\infty$ case to finite D cases. According to the experimental results in [211], the log-normal training distribution $p(\sigma)$ has the most substantial impact on the final performances. For ODE solver during sampling, we use Heun's 2^{nd} method [154] as in EDM.

Table 8.1: CIFAR-10 sample quality (FID) and number of function evaluations (NFE).

	Min FID ↓	Top-3 Avg FID ↓	NFE ↓
DDPM [167]	3.17	-	1000
DDIM [54]	4.67	-	50
VE-ODE [34]	5.29	-	194
VP-ODE [34]	2.86	-	134
PFGM [30]	2.48	-	104
PFGM++ (unconditional)			
$D = 64$	1.96	1.98	35
$D = 128$	1.92	1.94	35
$D = 2048$	**1.91**	**1.93**	35
$D = 3072000$	1.99	2.02	35
$D \to \infty$ [27]	1.98	2.00	35
PFGM++ (class-conditional)			
$D = 2048$	**1.74**	-	35
$D \to \infty$ [27]	1.79	-	35

Table 8.2: FFHQ 64×64 sample quality (FID) with 79 NFE in unconditional setting

	Min FID ↓	Top-3 Avg FID ↓
$D = 128$	**2.43**	2.48
$D = 2048$	2.46	**2.47**
$D = 3072000$	2.49	2.52
$D \to \infty$ [27]	2.53	2.54

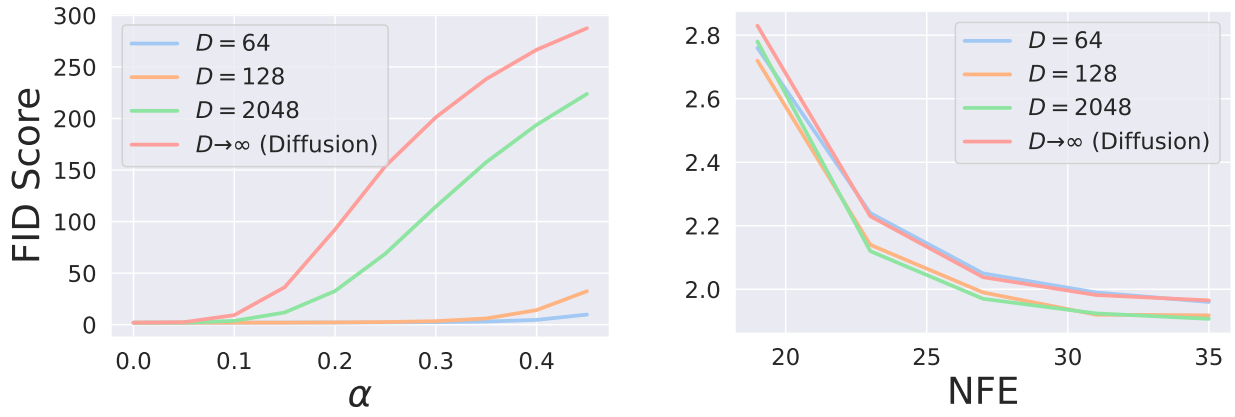
Table 8.3: LSUN Churches 256×256 sample quality (FID) with 99 NFE in unconditional setting

	Min FID ↓	Top-3 Avg FID ↓
$D = 131072$	**6.52**	6.58
$D \to \infty$ [27]	6.63	6.66

We compare models trained with $D \to \infty$ (EDM) and $D \in \{64, 128, 2048, 3072000\}$. In our experiments, we exclude the case of $D=1$ (PFGM) because the perturbation kernel is extremely heavy-tailed (Figure 8.4b), making it difficult to integrate with our perturbation-based objective without the restrictive region heuristics proposed in [30]. We also exclude the small $D = 64$ for the higher-resolution dataset FFHQ. Since the data dimension of LSUN Churches is relatively high (N=196608), we only try D=131072 to validate our ideas while saving computations. We include several popular generative models for reference and defer more training and sampling details to Appendix B.5.2.

Results: In Table 8.1, 8.2 and B.19, we report the sample quality measured by the FID score [41] (lower is better), and inference speed measured by the number of function evaluations. As in EDM, we report the minimum FID score over checkpoints. Since we empirically observe a large variation of FID scores on FFHQ across checkpoints (Appendix B.6.2), we also use the average FID score over the Top-3 checkpoints as another metric. Our main findings are **(1) Median Ds outperform previous best diffusion models [27] under PFGM++ framework.** We observe that the D=2048/128/131072 cases achieve the best performance among our choices on CIFAR-10/FFHQ/LSUN Churches, with min FID score of 1.91/2.43/6.52 in unconditional setting, using the perturbation-based objective. In addition, median Ds obtain better Top-3 average FID scores than EDM across datasets in unconditional setting and achieve a current state-of-the-art FID score of 1.74 in CIFAR-10 class-conditional

setting. **(2) There is a sweet spot between $(1, \infty)$.** Neither small D nor infinite D obtains the best performance, which confirms that there is a sweet spot in the middle, well-balancing rigidity and robustness. **(3) Model with $D \gg N$ recovers diffusion models.** We find that model with sufficiently large D roughly matches the performance of diffusion models, as predicted by the theory. Further results in Appendix B.6.3 show that D=3072000 and diffusion models obtain the same FID score when using a more stable training target [23] to mitigate the variations between different runs and checkpoints.



(a) (b)

Figure 8.5: FID score versus **(left)** α and **(right)** NFE on CIFAR-10.

Model Robustness versus D In Section 8.4, we show that the model robustness degrades with an increasing D by analyzing the behavior of perturbation kernels. To further validate the phenomenon, we conduct three sets of experiments with different sources of errors on CIFAR-10. We defer more details to Appendix B.6.2. Firstly, we perform controlled experiments to compare the robustness of models quantitatively. To simulate the errors, we inject noise into the intermediate point $\mathbf{x}_r$ in each of the 35 ODE steps: $\mathbf{x}_r = \mathbf{x}_r + \alpha \boldsymbol{\epsilon}_r$ where $\boldsymbol{\epsilon}_r \sim \mathcal{N}(\mathbf{0}, r/\sqrt{D}\boldsymbol{I})$, and α is a positive number controlling the amount of noise. Figure 8.5a demonstrates that as α increases, FID score exhibits a much slower degradation for smaller D. In particular, when D=64, 128, the sample quality degrades gracefully. We further visualize the generated samples in Appendix B.6.3. It shows that when α=0.2, models with D=64, 128 can still produce clean images while the sampling process of diffusion models ($D \to \infty$) breaks down.

In addition to the controlled scenario, we conduct two more realistic experiments: **(1)** We introduce more estimation error of neural networks by applying post-training quantization [213], which can directly compress neural networks without fine-tuning. Table 8.4 reports the FID score with varying quantization bit-widths for the convolution weight values. We can see that finite Ds have better robustness than the infinite case, and a lower D exhibits a larger performance gain when applying lower bit-widths quantization. **(2)** We increase the discretization error during sampling by using smaller NFEs, *i.e.*, larger sample steps. As shown in Figure 8.5b, gaps between D=128 and diffusion models gradually widen, indicating greater robustness against the discretization error. The rigidity issue of smaller D also affects the robustness to discretization error, as D=64 is consistently inferior to D=128.

Table 8.4: FID score versus quantization bit-widths on CIFAR-10.

Quantization bits:	9	8	7	6	5
$D = 64$	1.96	**1.96**	**2.12**	**2.94**	**28.50**
$D = 128$	1.93	1.97	2.15	3.68	34.26
$D = 2048$	**1.91**	1.97	**2.12**	5.67	47.02
$D \to \infty$	1.97	2.04	2.16	5.91	50.09

8.6 Conclusion

We present a new family of physics-inspired generative models called PFGM++, by extending the dimensionality of augmented variable in PFGM from 1 to $D \in \mathbb{R}^+$. Remarkably, PFGM++ includes diffusion models as special cases when $D \to \infty$. To address issues related to large batch training, we propose a perturbation-based objective. In addition, we show that D effectively controls the robustness and rigidity in the PFGM++ family. The multi-dimensional augmentation is crucial for empirical improvement, as it allows us to search for better models tailored to specific tasks and architectures, and enables the perturbation-based training objective (avoid the heavy-tailed problem when $D = 1$ as in PFGM [30]). On the other hand, the perturbation-based objective reduces training overheads and makes PFGM++ applicable to typical conditional generation settings. Empirical results show that models with finite values of D can perform better than previous state-of-the-art diffusion models, while also

exhibiting improved robustness.

There are many potential avenues for future research in the PFGM++ framework. For example, it may be possible to identify the "sweet spot" value of D for different architectures and tasks by analyzing the behavior of errors. Since PFGM++ enables adjusting robustness, another direction is to apply aggressive network compression techniques, *i.e.*, pruning and low-bit training, to smaller D. Furthermore, there may be opportunities to develop stochastic samplers for PFGM++, with the reverse SDE in diffusion models as a special case. Lastly, PFGM++ may yield more significant performance improvements over diffusion models (the $D \to \infty$ case) in fields with less optimized network architectures. Our theoretical and experimental results demonstrate that PFGM++ exhibit superior robustness compared to diffusion models when using a smaller D. This increased robustness can translate to more substantial improvements on weaker architectures.we expect PFGM++ to have more significant performance gains than diffusion models in domains other than image generation, where network architectures have already been extensively optimized. We will leave the application of PFGM++ to other fields for future work.

Chapter 9

Duality between Physical Processes and Generative Models

Having witnessed the success of diffusion models and PFGM, we ask if there is a systematic way to convert physical processes into generative models. The answer is yes! The well-established and mature nature of physics compared to generative modeling suggests a vast potential for the transfer of knowledge from the former to the latter. In this chapter, we explore the intriguing duality between physics and generative models. We present a unifying framework and algorithm that transforms physical processes into smooth density flow generative models. Additionally, we introduce a classification criterion based on the dispersion relations of the underlying physical partial differential equations (PDEs). This theoretical approach can be applied to various physical PDEs, leading to the discovery of new families of generative models. Our work lays the groundwork for the principled design of machine learning generative models derived from the theories of physics.

This chapter is based on the paper [13]. Ziming Liu and Di Luo contributed significantly to the materials in this chapter. My contributions are developing the idea with them and helping with paper writing.

9.1 Introduction

The connection between physics and generative models can be quite deep. Our Universe is arguably a generative model [214], [215]: Starting from the wave function of our early Universe, which was a simple multivariate Gaussian corresponding to spatially uniform fields with small quantum fluctuations, our Universe evolves to "generate" ever richer and more complex phenomenon. However, the dynamics that drive the evolution are described by (simple and elegant) partial differential equations (PDEs). The same applies to generative models that leverage continuous physical processes: although the whole transformation from latent to data distribution can be quite complicated, the movement at each step is simple, ready to be learned by deep neural networks.

In fact, the developments of generative models have been heavily influenced by physics [10], [12], [20], [24], [31], [38], [50], [153], [160], [197], [207], [216]–[220]. Recently, we have witnessed the success of physics-inspired deep generative models, such as diffusion models (DM) [10], [12], [24], [27] based on thermodynamics and Poisson flow generative models (PFGM) [31], [50] derived from electrostatics. The idea of diffusion models is to reverse the process of ink diffusing in water, while Poisson flows view data points as charged particles and let them move in electric fields. Given these two successful examples, it is thus natural to ask: can physics offer more to generative models (see FIG. 9.1)? Concretely, given a physical process,

$$\hat{L}\phi(\mathbf{x},t) \equiv F(\underbrace{\phi, \phi_t, \phi_{tt}, \nabla\phi, \nabla^2\phi, ...}_{\phi'}) = f(\mathbf{x},t), \tag{9.1}$$

can we convert it to a generative model?

In this chapter, we develop a unifying theory that exploits the duality between physics and generative models to answer this question. Our framework provides a concrete algorithm for translating a broad spectrum of physical processes into generative models in three steps. We demonstrate explicitly how the famous diffusion model can arise from diffusion equation in our formulation. Furthermore, we introduce a crucial criterion for classifying the potential of various physical processes and PDEs for conversion into generative models, using the dispersion relation as a guiding metric. Applying our methodology across different physical

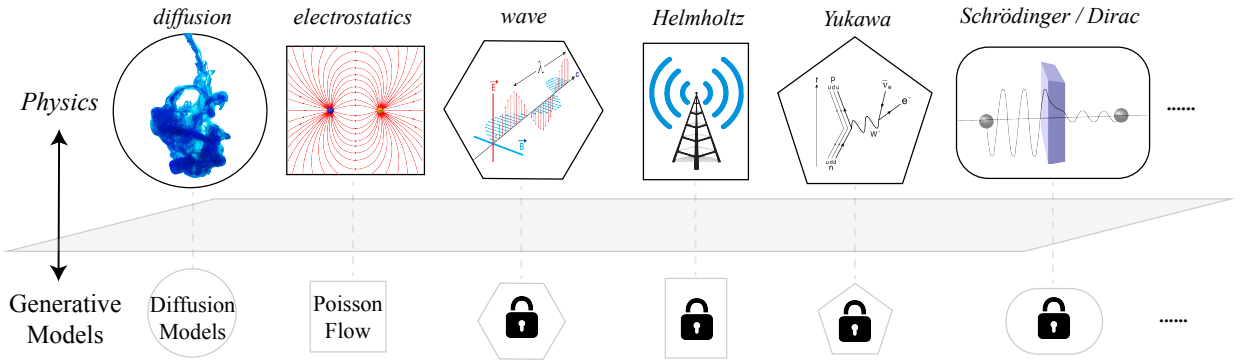


Figure 9.1: Duality between physics and generative models. So far only diffusion models and Poisson flows are discovered by researchers. Can we unlock more?

PDEs, we have unveiled a new family of generative models, which we call *smooth density flows* elaborated below. Our study establishes a foundational understanding and offers a principled approach to designing generative models rooted in the principles of physics.

9.2 Converting Physical Processes to Generative Models

This section reveals a connection between continuous physical processes and generative models. The key is to match their associated PDEs: each physical process is described by a PDE, while each generative model is associated with a density flow (which is also a PDE). To convert a physical process to a generative model, we need three conversion steps (see FIG. 9.2): **Step I** converts a physical process to a PDE, which is always feasible in the sense that PDEs are just mathematical abstractions of physical processes; **Step II** rewrites the PDE to a density flow (assuming the density flow condition). **Step III** converts the density flow to a generative model (assuming the smooth condition).

Step I. Physical processes are described by PDEs.—Continuous physical processes are described by partial differential equations, see Eq. (9.1), where $\phi(\mathbf{x}, t)$ is a scalar function defined on $\mathbb{R}^N \times \mathbb{R}^+$, $\hat{L}$ is a differential operator acting on $\phi(\mathbf{x}, t)$, $f(\mathbf{x}, t)$ is the source term, and subscripts stand for partial derivatives, e.g., $\phi_t \equiv \partial\phi/\partial t, \phi_{tt} \equiv \partial^2\phi/\partial t^2$, and ϕ' stands for the collections of partial derivatives of ϕ. For example, the diffusion equation interprets $\phi(\mathbf{x}, t)$ as temperature, $\hat{L}(\phi) = F(\phi') = \phi_t - \nabla^2\phi$, and $f(\mathbf{x}, t)$ is interpreted as heat source. For simplicity, we will mostly discuss linear PDEs which are also symmetric

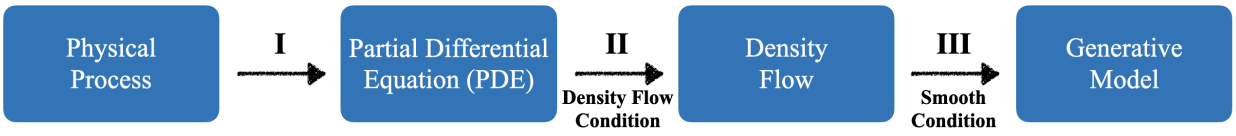


Figure 9.2: Framework that converts physical processes to generative models.

both in space and time [1], i.e., where F does not depend explicitly on $\mathbf{x}$ or t. The linearity and symmetries usually make $\phi(\mathbf{x}, t)$ analytically solvable and these solutions are available in many mathematical physics textbooks [221], [222].

For linear PDEs, the solution $\phi(\mathbf{x}, t)$ can be expressed as a convolution of the Green's function $G(\mathbf{x}, t; \mathbf{x}', t')$ with the source term $f(\mathbf{x}, t)$, i.e., $\phi(\mathbf{x}, t) = \int G(\mathbf{x}, t; \mathbf{x}', t') f(\mathbf{x}', t') \mathrm{d}^N \mathbf{x}' dt$ [223]. The Green's function $G(\mathbf{x}, t; \mathbf{x}', t')$ is defined as the solution of $\hat{L} G(\mathbf{x}, t; \mathbf{x}', t') = \delta(\mathbf{x} - \mathbf{x}')\delta(t - t')$ with $G(\mathbf{x}, t; \mathbf{x}', t') = 0$ when $t < t'$.

Step II. Converting PDEs to density flows.—When a density distribution $p(\mathbf{x}, t)$ has flow velocity $\mathbf{v}(\mathbf{x}, t)$ and birth rate $R(\mathbf{x}, t)$, it satisfies the following density flow equation (a.k.a Fokker-Planck equation):

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} + \nabla \cdot [p(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)] - R(\mathbf{x}, t) = 0, \tag{9.2}$$

which itself is a PDE. We want to convert the physical PDE to a density flow, i.e., rewrite Eq. (9.1) in the form of Eq. (9.2) (called the *density flow condition*). Although not all PDEs have this property, many physical PDEs satisfy this condition.

Step III. Converting density flows to generative models.—Given i.i.d. data samples from the probability distribution $p_{\text{data}}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^N$, the goal of generative models is to obtain new samples from $p_{\text{data}}(\mathbf{x})$. Recent generative models, including diffusion models and Poisson flow models, leverage this ordinary differential equation $\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}, t)$ for generation, with differences in the functional form of $\mathbf{v}(\mathbf{x}, t)$. This ODE can evolve the probability distribution $p(\mathbf{x}, t)$ as $\frac{\partial p(\mathbf{x}, t)}{\partial t} + \nabla \cdot [p(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)] = 0$, known as the *probability flow equation*, or the *continuity equation*. Here p and $\mathbf{v}$ are interpreted as a probability distribution and a velocity field, respectively. The probability distribution, starting as the data distribution $p(\mathbf{x}, 0) = p_{\text{data}}(\mathbf{x})$,

[1] This means that $\hat{L}$ remains unchanged under (1) translations in time $t \to t + \delta t$, (2) translations in space $\mathbf{x} \to \mathbf{x} + \delta \mathbf{x}$, and (3) rotations in space $\mathbf{x} \to \mathbf{R}\mathbf{x}$.

evolves to a (hopefully simple) final distribution $p(\mathbf{x}, T)$. To generate samples from $p_{\text{data}}(\mathbf{x})$, one can first draw samples from the final distribution $p_{\text{prior}}(\mathbf{x}) \equiv p(\mathbf{x}, T)$, and run the process $\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}, t)$ backward from $t = T$ to $t = 0$ [2].

Although we assumed conservation above, we can more generally allow a birth term $R(\mathbf{x}, t)$, giving Eq. (9.2). $R > 0$ ($R < 0$) means particles are born (die) in the forward process, and die (are born) in the backward process [3]. In this case, $p(\mathbf{x}, t)$ is a density distribution instead of a probability distribution, so we call Eq. (9.2) the *density flow equation*. This extension of density flows has found numerous applications in machine learning, such as unbalanced optimal transport methods for modeling single-cell dynamics and domain adaptation [224], [225], as well as in Bayesian inference for probabilistic modeling [226]. In practice, the density flow with birth/death dynamics can be simulated efficiently. For example, in diffusion Monte Carlo, the birth/death processes can be included as branching processes with population control on $R(\mathbf{x}, t)$ [227], [228].

Algorithm.—Above we have clarified all the steps in FIG. 9.2, so now we can assembly them into Algorithm 4. The algorithm takes in: (1) a partial differential equation $\hat{L}\phi = 0$ which describes certain physical process; (2) data distribution $p_{\text{data}}(\mathbf{x})$ represented by a set of training samples. The algorithm outputs generated samples. Below, we showcase how to use this algorithm to convert the diffusion equation into the diffusion models.

Remember that our goal is to convert the diffusion equation $\phi_t - \nabla^2\phi = 0$ to a density flow $\frac{\partial p}{\partial t} + \nabla \cdot (p\mathbf{v}) - R = 0$:

$$\begin{aligned} \phi_t - \nabla^2\phi =& \frac{\partial \phi}{\partial t} + \nabla \cdot (\phi(-\nabla \log \phi)) - 0 = 0 \\ \Leftrightarrow& \frac{\partial p}{\partial t} + \nabla \cdot (p\mathbf{v}) - R = 0 \end{aligned} \tag{9.3}$$

Matching the two sides gives:

$$p = \phi, \quad \mathbf{v} = -\nabla \log \phi, \quad R = 0. \tag{9.4}$$

[2] Note that time is usually defined in opposite directions in physics and machine-learning applications: our Universe generates complex structures as time moves forward, whereas generative models using e.g. diffusion make things simpler over time and generate complexity by evolving backward in time.

[3] In the time interval $[t, t + dt]$, a forward particle at $\mathbf{x}$ has probability $|R|dt$ to turn into two/zero particles when $R > 0/R < 0$.

Algorithm 4 Generative models from physical processes (GenPhys)

Input: partial differential equation $\hat{L}\phi(\mathbf{x},t)=0$, data distribution $p_{\text{data}}(\mathbf{x})$
(1) Rewrite $\hat{L}\phi(\mathbf{x},t)$ in the form $\frac{\partial p(\mathbf{x},t)}{\partial t}+\nabla\cdot[p(\mathbf{x},t)\mathbf{v}(\mathbf{x},t)]-R(\mathbf{x},t)$ such that $p=p(\phi,\phi_t,\nabla\phi,\cdots)$, $\mathbf{v}=\mathbf{v}(\phi,\phi_t,\nabla\phi,\cdots)$, $R=R(\phi,\phi_t,\nabla\phi,\cdots)$
(2) Solve $\hat{L}\phi(\mathbf{x},t)=p_{\text{data}}(\mathbf{x})\delta(t)$. If $\hat{L}$ is linear, we can express $\phi(\mathbf{x},t)$ in terms of the Green's function $G(\mathbf{x},t;\mathbf{x}')$: $\phi(\mathbf{x},t)=\int G(\mathbf{x},t;\mathbf{x}')p_{\text{data}}(\mathbf{x}')\mathrm{d}^N\mathbf{x}'$, where $\hat{L}G(\mathbf{x},t;\mathbf{x}')=\delta(\mathbf{x}-\mathbf{x}')\delta(t)$
(3) Using the relations in (1) and solutions in (2) to obtain $p(\mathbf{x},t)$, $\mathbf{v}(\mathbf{x},t)$, $R(\mathbf{x},t)$
(4) Train a neural network $\mathbf{s}_\theta(\mathbf{x},t)$ to fit $\mathbf{v}(\mathbf{x},t)$ such that $\mathbf{s}_\theta(\mathbf{x},t)\approx\mathbf{v}(\mathbf{x},t)$. Train another neural network $W_\alpha(\mathbf{x},t)$ to fit $R(\mathbf{x},t)$ such that $W_\alpha(\mathbf{x},t)\approx R(\mathbf{x},t)$
(5) Draw $\mathbf{x}(T)\sim p(\mathbf{x},T)$, simulate $\frac{d\mathbf{x}(t)}{dt}=\mathbf{s}_\theta(\mathbf{x},t)$ from $t=T$ to $t=0$ with the branching process $W_\alpha(\mathbf{x},t)$. Output $\mathbf{x}(0)$.
Return Generated samples $\mathbf{x}$

The solution ϕ of the diffusion equation is (with the initial condition $\phi(\mathbf{x},0)=p_{\text{data}}(\mathbf{x})$):

$$\begin{aligned}\phi(\mathbf{x},t)&=\int G(\mathbf{x},t;\mathbf{x}')p_{\text{data}}(\mathbf{x}')\mathrm{d}^N\mathbf{x}',\\ G(\mathbf{x},t;\mathbf{x}')&=\frac{1}{(2\pi t)^{N/2}}\exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2t}).\end{aligned}\tag{9.5}$$

Combining Eq. (9.4) and Eq. (9.5) gives

$$\begin{aligned}p(\mathbf{x},t)&=\frac{1}{(2\pi t)^{N/2}}\int p_{\text{data}}(\mathbf{x}')\exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2t})\mathrm{d}^N\mathbf{x}',\\ \mathbf{v}(\mathbf{x},t)&=\mathbb{E}_{p_t(\mathbf{x}'|\mathbf{x})}\left(\frac{\mathbf{x}-\mathbf{x}'}{t}\right),\\ R(\mathbf{x},t)&=0,\end{aligned}\tag{9.6}$$

where $p_t(\mathbf{x}'|\mathbf{x})\propto p_{\text{data}}(\mathbf{x}')G(\mathbf{x},t;\mathbf{x}')$. Note that $-\mathbf{v}(\mathbf{x},t)$ recovers the *score function* in [24], [197], *i.e.*, $\nabla_{\mathbf{x}}\log p(\mathbf{x},t)$.

9.3 Classification via Dispersion Relation

Although density flows and generative models are equivalent mathematically, generative models further have this practical consideration: $p_{\text{prior}}(\mathbf{x})\equiv p(\mathbf{x},T)$ should be asymptotically independent of $p_{\text{data}}(\mathbf{x})$ as $T\to\infty$. It motivates us to study a family of equations that satisfy

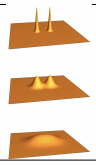
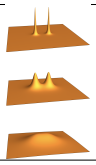
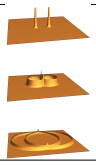
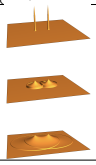
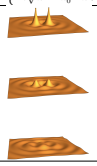
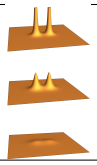
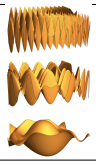
equation	diffusion equation	Poisson equation	ideal wave equation	dissipative wave equation	Helmholtz equation	screened Poisson equation (Yukawa)	Schrödinger equation (free)
PDE $L\phi = 0$	$\phi_t - \nabla^2\phi = 0$	$\phi_{tt} + \nabla^2\phi = 0$	$\phi_{tt} - \nabla^2\phi = 0$	$\phi_{tt} + 2\epsilon\phi_t - \nabla^2\phi = 0$	$\phi_{tt} + \nabla^2\phi + k_0^2\phi = 0$	$\phi_{tt} + \nabla^2\phi - m^2\phi = 0$	$i\phi_t + \nabla^2\phi = 0$
Rewritten	$\frac{\partial\phi}{\partial t} + \nabla\cdot(\phi(-\nabla\log\phi)) = 0$	$\frac{\partial(-\phi_t)}{\partial t} + \nabla\cdot((-\phi_t)(\frac{\nabla\phi}{\phi_t})) = 0$	$\frac{\partial(-\phi_t)}{\partial t} + \nabla\cdot((-\phi_t)(-\frac{\nabla\phi}{\phi_t})) = 0$	$\frac{\partial(-\phi_t - 2\epsilon\phi)}{\partial t} + \nabla\cdot((-\phi_t - 2\epsilon\phi)(\frac{\nabla\phi}{\phi_t + 2\epsilon\phi})) = 0$	$\frac{\partial(-\phi_t)}{\partial t} + \nabla\cdot((-\phi_t)(\frac{\nabla\phi}{\phi_t})) - k_0^2\phi = 0$	$\frac{\partial(-\phi_t)}{\partial t} + \nabla\cdot((-\phi_t)(\frac{\nabla\phi}{\phi_t})) + m^2\phi = 0$	$\frac{\partial\phi'}{\partial t} + \nabla\cdot(\lvert\phi\rvert^2(2\mathrm{Im}\nabla\log\phi)) = 0$
p	ϕ	$-\phi_t$	$-\phi_t$	$-(\phi_t + 2\epsilon\phi)$	$-\phi_t$	$-\phi_t$	$\lvert\phi\rvert^2$
$\mathbf{v}$	$-\nabla\log\phi$	$\frac{\nabla\phi}{\phi_t}$	$-\frac{\nabla\phi}{\phi_t}$	$\frac{\nabla\phi}{\phi_t + 2\epsilon\phi}$	$\frac{\nabla\phi}{\phi_t}$	$\frac{\nabla\phi}{\phi_t}$	$2\mathrm{Im}\nabla\log\phi$
R	0	0	0	0	$k_0^2\phi$	$-m^2\phi$	0
$G(r,t)$	$\frac{1}{(4\pi t)^{\frac{N}{2}}}\exp(-\frac{r^2}{4t})$	$\frac{1}{(\frac{1}{2}\pi r)^{\frac{N-1}{2}}}$	$\frac{1}{\sqrt{t^2-r^2}}\Theta(t-r)$ (2D)	$\frac{e^{-\epsilon t}\Theta(t-r)}{\sqrt{t^2-r^2}}$ (2D)	$(\frac{k_0}{\sqrt{t^2+r^2}})^{\frac{N-2}{2}}H^{(1)}_{\frac{N-2}{2}}(k_0\sqrt{t^2+r^2})$	$(\frac{m}{\sqrt{t^2+r^2}})^{\frac{N-2}{2}}K_{\frac{N-2}{2}}(m\sqrt{t^2+r^2})$	$\frac{1}{(4\pi it)^{\frac{N}{2}}}\exp(\frac{ir^2}{4t})$
$\hat{G}(k,t)$	$\exp(-k^2t)$	$\exp(-kt)$	$\exp(\pm ikt)$	$\exp(-\epsilon t + i\sqrt{k^2-\epsilon^2}t)$ $(k > \epsilon)$ $\exp(-(\epsilon + \sqrt{k^2-\epsilon^2})t)$ $(k \le \epsilon)$ $\hat{G}(k,t)$	$\exp(-i\sqrt{k_0^2-k^2}t)$ $(k \le k_0)$ $\exp(-\sqrt{k^2-k_0^2}t)$ $(k > k_0)$	$\exp(-\sqrt{k^2+m^2}t)$	$\exp(ik^2t)$
Dispersion relation $\omega(k)$	$\omega = -ik^2$	$\omega = \pm ik$	$\omega = \pm k$	$\omega = \begin{cases} i(-\epsilon \pm \sqrt{\epsilon^2-k^2}) & k \le \epsilon \\ i\epsilon \pm \sqrt{k^2-\epsilon^2} & k > \epsilon \end{cases}$	$\omega = \begin{cases} \pm\sqrt{k_0^2-k^2} & k \le k_0 \\ \pm i\sqrt{k^2-k_0^2} & k > k_0 \end{cases}$	$\omega = \pm i\sqrt{k^2+m^2}$	$\omega = k^2$
Illustration ϕ							
Density flow condition	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Smooth condition	Yes	Yes	No	Conditionally yes	Conditional Yes	Yes	No
smooth density flow?	Yes (Diffusion Models)	Yes (Poisson Flow)	No	Conditionally Yes (large ϵ)	Conditionally yes (small k)	Yes	No

Table 9.1: Summary of results for different physical equations, their properties, and whether they can be converted to a generative model. $\mathbf{x}'$ and $\mathbf{x}$ are the source point and the field point, and $r \equiv |\mathbf{x} - \mathbf{x}'|$.

the *smooth condition*, i.e., solutions become smoother over time. We call a physical process that satisfies both the density flow and smooth conditions as *smooth density flow*.

For the diffusion equation, we can show that the smooth condition is met. We define D to measure the independence of the final distribution on the initial condition:

$$\begin{aligned} D(\mathbf{x}_1', \mathbf{x}_2', T) &\equiv \int \sqrt{G(\mathbf{x}, T; \mathbf{x}_1')}\sqrt{G(\mathbf{x}, T; \mathbf{x}_2')}\mathrm{d}^N\mathbf{x} \\ &= \exp\left(-\frac{|\mathbf{x}_1 - \mathbf{x}_2|^2}{8T}\right), \end{aligned} \tag{9.7}$$

where $D = 1$ means independence, and $D = 0$ means dependence. We have $\lim_{T\to\infty} D(\mathbf{x}_1', \mathbf{x}_2', T) \to 1$, implying the smooth condition.

Checking the smooth condition case by case can be quite math-heavy, as we demonstrate in Appendix A.6.1. We now show that dispersion relations can elegantly characterize the smooth condition. The dispersion relation relates a wave's wave number k to its frequency ω. All linear PDEs $\hat{L}\phi(\mathbf{x}, t) = 0$ have wave solutions: $\phi(\mathbf{x}, t) \propto \exp(-i\omega t)\exp(i\mathbf{k}\cdot\mathbf{x})$. Substituting the wave ansatz into the PDE gives the dispersion relation $\omega(\mathbf{k})$. For example, the diffusion equation $\phi_t - \nabla^2\phi = 0$ has $\omega = -ik^2 (k \equiv |\mathbf{k}|)$, where w is purely imaginary, so the time factor $\exp(-i\omega t) \sim \exp(-k^2 t)$ decays in time. In contrast, the wave equation $\phi_{tt} - \nabla^2\phi = 0$ has $\omega(k) = \pm k$, where ω is real, so the time factor $\exp(\pm ikt)$ oscillates in time without any decay.

In Appendix A.6.3, we prove that the smooth condition is, in fact equivalent to the

following condition:

$$\text{Im } \omega(k) < \text{Im } \omega(0), \quad \forall k > 0 \tag{9.8}$$

Note that although dispersion relations may have multiple branches, we only require one branch to satisfy Eq. (9.8). For example, dispersion relations of the Poisson equation have two branches $\omega = \pm ik$, one of which ($\omega = -ik$) satisfies Eq. (9.8). Intuitively, the smooth condition requires that details of the initial distribution smooth out as time elapses, which means that high-frequency modes should decay faster than low-frequency ones. Eq. (9.8) states that non-zero frequency modes should decay faster than the zero frequency mode. If we define $\hat{p}(\mathbf{k})$ as the spatial Fourier transform of $p(\mathbf{x})$, this condition simply means that spatial oscillations with $k \equiv |\mathbf{k}| > 0$ decay faster than the total probability/mass $\hat{p}(\mathbf{k} = 0, t) = \int p(\mathbf{x}, t) \mathrm{d}^N \mathbf{x}$. In Table 9.1, we list the dispersion relations $\omega(k)$ of several physical PDEs, which we elaborate below.

9.4 Applications.

We will examine several representative PDEs in physics, including the diffusion equation, the Poisson equation, the wave equation (ideal or dissipative), the Helmholtz equation, the screened Poisson equation and the Schrödinger equation. When a physical process p satisfies both the density flow condition and the smooth condition, we call it smooth density flow. In summary: (1) p is smooth density flow. Examples: p = diffusion, Poisson, screened Poisson (Yukawa). (2) p is conditionally smooth density flow. Examples: p = dissipative wave, Helmholtz. (3) p is not a smooth density flow. Examples: p = ideal wave, Schrödinger.

For each PDE, we rewrite it into the form of a density flow (figuring out its corresponding $(p, \mathbf{v}, R)$), calculate the Green's function and dispersion relation. In the following, we focus on their physical intuition, with detailed results summarized in TABLE 9.1.

The diffusion equation is $\phi_t - \nabla^2 \phi = 0$, describing how densities (e.g., mass, heat, charge) transport in space. In a steady environment, a system would reach thermal equilibrium, satisfying the smoothing condition.

The Poisson equation is $\nabla^2 \phi = 0$. If we interpret one of its spatial dimensions to be time, then the equation becomes $\phi_{tt} + \nabla^2 \phi = 0$. Remember, we interpret $p = -\phi_t$ as density,

and $\mathbf{v} = -\nabla\phi/\phi_t$, whose mental picture is charged particles moving along the electric field lines. Any compact distribution, when they go very far away along electric field lines, will become a uniform distribution on a hypersphere [50]. Equivalently, this is because any charge distribution, when viewed from far away, behaves like a point particle. In this sense, the Poisson equation satisfies the smoothing condition.

The ideal wave equation is $\phi_{tt} - \nabla^2\phi = 0$, describing the propagation of waves of sound, light, etc [229]. The wave equation preserves information in the sense that a wave front travels with a constant speed away from the source. Think of a stone dropped into water. The information preservation property goes against the smooth condition where we want smoothing effects.

The dissipative wave equation is $\phi_{tt} + 2\epsilon\phi_t - \nabla^2\phi = 0$ where ϵ is the damping coefficient [230]. It describes wave propagation with dissipation. Dissipation slows down the wave propagation, leading to behavior "interpolating" between wave and diffusion: $\epsilon \to 0$ recovers ideal waves, and $\epsilon \to \infty$ recovers diffusion. Choosing a large (small) enough ϵ will make the process quantitatively similar to diffusion (wave), so the smooth condition should hold (fail).

The Helmholtz equation $(\nabla^2_{\tilde{\mathbf{x}}} + k_0^2)\phi = 0$ is the single-frequency wave equation, where $\tilde{\mathbf{x}} = (\mathbf{x}, t)$. It explains the wave-like behavior (shown in Table 9.1) in its Green's function. Think of water ripples driven by a periodic source. $k_0 \to 0$ recovers the Poisson equation, so a small enough k_0 should be smooth density flow. However, if k_0 is too large, Green's function may become negative in the region of interest, violating the density flow condition.

The screened Poisson equation, a.k.a. Yukawa $(\nabla^2_{\tilde{\mathbf{x}}} - m^2)\phi = 0$, where m expresses the "screening". Compared to the Poisson potential (the solution of the Poisson equation), the screened Poisson potential is more short-ranged with a length scale m^{-1}, due to screening effects. When $N = 3$, ϕ is the well-known Yukawa potential [231]. Note that $m = 0$ recovers the Poisson equation. The solution is qualitatively similar to that of the Poisson equation, although decreases faster with distance.

The Schrödinger equation of a free particle is $i\phi_t = -\nabla^2\phi + V(\mathbf{x})\phi$, where ϕ is the (complex-valued) wave function. It describes the evolution of a quantum particle [232]. The Schrödinger equation describes the wave nature of the particles, based on the idea of wave-

particle duality. For $V(\mathbf{x}) = 0$, the free particle PDE implies $\frac{\partial|\phi|^2}{\partial t} + \nabla \cdot [|\phi|^2(2\text{Im}\nabla\log\phi)] = 0$ (see Appendix A.6.1 for details), so $p = |\phi|^2$, $\mathbf{v} = 2\ \text{Im}\nabla\log\phi$ and $R = 0$. The Green's function $G(\mathbf{x}, t; \mathbf{x}') = \frac{1}{(4it)^{N/2}}\exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{4it})$. $p(\mathbf{x}, T)$ oscillates restlessly even for large T and depends on the initial distribution, violating the smooth condition. However, to have the final distribution independent of the initial distribution, it is possible to consider the subsystem quantum dynamics under the Schrödinger evolution such that it thermalizes or reaches a steady state in the open quantum system formulation.

Physics	PDE	Dispersion Relation
Mixed diffusion Poisson	$a\phi_{tt} - b\phi_t + \nabla^2\phi = 0\ (a > 0, b > 0)$	$\omega = \frac{i}{2a}(b \pm \sqrt{b^2 + 4ak^2})$
Fractional diffusion	$\phi_t + (-\Delta)^\beta\phi = 0\ (\beta > 0)$	$\omega = -ik^{2\beta}$
Third-order diffusion	$\phi_{ttt} - \Delta u = 0$	$\omega = (-i, e^{i\frac{\pi}{6}}, e^{i\frac{5\pi}{6}})k^{\frac{2}{3}}$
Elasticity (Biharmonic)	$\phi_t + \nabla^2\nabla^2\phi = 0$	$\omega = -ik^4$

Table 9.2: Dispersion relation suggests new generative models

PDEs for a new family of smooth density flow We can now simply invent generative models by constructing PDEs whose dispersion relations satisfy the condition Eq. (9.8). We list a few examples in Table 9.2. The mixed diffusion Poisson equation interpolates between the diffusion equation and the Poisson equation. The fractal diffusion and the third-order diffusion equations characterize more exotic diffusion beyond Brownian motion. The elasticity equation (a.k.a bi-harmonic equation), widely used in continuum mechanics, can also be converted to a smooth density flow.

9.5 Conclusion

We have developed a novel framework that can construct generative models from physical processes and lead to the discovery of a new family of smooth density flow. Our framework establishes the foundation for first-principle understanding and design of machine learning models based on physics theories. Meanwhile, it also opens up a number of exciting opportunities that are beyond the current analysis, including: (1) From smoothing to non-smoothing PDEs. There exists non-smooth density flow model which can also provide useful generative modeling, such as the case in quantum machine learning with dynamics based on the

Schrödinger equation and quantum circuits. (2) From linear to nonlinear PDEs, e.g., the Navier-Stokes equation, the reaction-diffusion equation, and the Bose-Einstein condensation, etc. (3) From symmetric to general PDEs. We only discussed PDEs with translational or rotational symmetry with analytical accessibility of the Green's function, but generic PDEs without such symmetries may offer additional power and flexibility. For example, the imaginary time evolution of a Schrödinger equation with position-dependent potential gives rise to the birth/death process in the density flow. (4) From time-independent to time-dependent PDEs. This opens up connections to a broader class of dynamical processes in nature, such as non-equilibrium dynamics, quench experiments and annealing. For example, it is also known that adiabatic quantum dynamics can achieve universal quantum computation [233], which can be leveraged for generic probability modeling. Our work unlocks numerous possibilities for advancing machine learning via physics, heralding a new era of interdisciplinary innovation in both fields.

Chapter 10

Conclusion

Physics-inspired generative models, exemplified by diffusion models, have emerged as a strong family in generative modeling. Their underlying training objective is essentially fitting a fixed vector field (*e.g.,* score function in diffusion models) and requires no coordination among multiple neural networks as in VAEs [85] or GANs [14], [79], resulting in a more stable learning process. In addition, they are less constrained by architectural requirements compared to normalizing flows [16], [234] and employ an iterative backward process similar to the concatenation of neural networks, thereby possessing high capacity.

Taken together, training stability and capacity are likely the key characteristics that primarily account for the dominance of diffusion models in tackling larger problems and across various modalities. For example, in the vision domain, diffusion models have achieved remarkable performance on text-to-image generation [2], [235], text-to-3D generation [130], [133] and text-to-video generation [127], [128]. Diffusion models also dominate many generative modeling tasks in the other domains, including biology (docking [236] and molecular conformation generation [168], [184]) and speech synthesis [237], [238]. Moreover, diffusion models excel at solving challenging ill-posed inverse problems commonly encountered in science and engineering, such as super-resolution [239] and medical image reconstruction [240].

In this thesis, we have presented improved techniques in both the training and sampling of diffusion models. We enhance the training of diffusion models to reduce the variance of training targets (Chapter 3) and the curvature of generative trajectories (Chapter 4). To accelerate the generation speed of diffusion models, we design a new sampler that better

utilizes the stochasticity in the sampling process, achieving superior performance in all network function evaluation (NFE) regimes (Chapter 5). We further exert mutually repulsive force in the sampling process to promote the diversity among mini-batch samples (Chapter 6).

Despite the clear improvements demonstrated by these techniques in this thesis, several research questions remain unexplored. First of all, our training techniques have been tested on standard benchmarks like ImageNet. With additional computing resources, there is potential to apply these ideas to larger scale tasks, such as text-to-image generation. Secondly, the proposed sampling technique is not optimized for extremely low NFEs, and it is anticipated that more efficient noise utilization could enhance models in this regime. Lastly, we currently lack a precise characterization of the final joint distribution when applying the repulsive force in the sampling of diffusion models, suggesting an avenue for further research.

Additionally, we introduced a new family of generative models, Poisson Flow Generative Models (PFGM), which draw inspiration from electrostatics theory. We demonstrate that PFGM outperforms previous diffusion models while offering a faster and more robust generation process. Since the introduction of this model, many researchers have applied it to other fields, such as antibody generation [241] and medical image (CT scan) [242] generation. We further expand the electrostatics theory used in PFGM, unifying diffusion models and PFGM. More intriguingly, interpolation between the two models reveals a sweet spot with new state-of-the-art performance in image generation (Chapter 8). Finally, we present a systematic approach to convert physical processes into generative models, to expand the design space of generative models (Chapter 9).

Although we address the question of *how to construct generative models based on physical processes*, the question of *which physical processes are optimal as forward processes* remains unanswered. To answer the question, we can encapsulate these physical processes within a space that can be optimized for generative modeling, enabling the selection of the most suitable process for specific tasks or architectures. Physical processes introduce natural hyper-parameters that characterize the generative process. For instance, diffusion models and Poisson Flow Generative Models (PFGM) can be viewed as methods that progressively smooth the data distribution using different time-dependent kernel functions. We can integrate optimizable parameters within these valid kernel functions and assess the performance of the

resulting generative models by evaluating sample quality, sampling robustness, and ease of learning, among other criteria.

Another interesting question is *are vector fields arising from physical forward models easier to learn when the data come from physical processes?* Physics offers bridges between complex data and simple prior, with fixed vector fields to be learned. The fixed vector fields lead to a stable training process and underpin the scalability of physics-inspired generative models. When data comes from physical processes, the advantage of physics-inspired generative models may depend on whether neural networks naturally learn such vector fields more effectively than other models. The suitability might depend on whether the neural network's inductive biases align well with the underlying vector fields in physical data, which tends to be smooth and meaningful. For comparison, it seems unnatural to sequentially factorize data and require a neural network to model each conditional distribution in auto-regressive models. We will reserve these questions for future exploration.

Overall, this thesis provides new progress in three key areas of physics-inspired generative models: training, sampling, and the development of a new family of such models. Looking forward, we anticipate the emergence of more research ideas in this field, which will further advance data-driven techniques for generative modeling as a whole and benefit downstream applications like artwork creation, drug discovery, and speech synthesis.

Appendix A

Additional Proofs and Derivations

A.1 Chapter 3

A.1.1 Derivation of Equation 3.6

Recall that the STF objective (Equation 3.5) at time t is

$$\ell_{\text{STF}}(\theta, t) = \mathbb{E}_{\{\mathbf{x}_i\}_{i=1}^n \sim p_0^n} \mathbb{E}_{\mathbf{x}(t) \sim p_{t|0}(\cdot|\mathbf{x}_1)} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right\|_2^2 \right].$$

Swapping the sampling order and we get

$$\ell_{\text{STF}}(\theta, t) = \mathbb{E}_{\mathbf{x}(t) \sim p_t} \mathbb{E}_{\{\mathbf{x}_i\}_{i=1}^n \sim p_{0|t}(\cdot|\tilde{\mathbf{x}}) p_0^{n-1}} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right\|_2^2 \right].$$

This means that with input $\tilde{\mathbf{x}}$ and t, the model is optimized with

$$\mathbb{E}_{\{\mathbf{x}_i\}_{i=1}^n \sim p_{0|t}(\cdot|\tilde{\mathbf{x}}) p_0^{n-1}} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right\|_2^2 \right].$$

Taking its derivative w.r.t. $\mathbf{s}_\theta(\tilde{\mathbf{x}}, t)$ results in

$$\mathbb{E}_{\{\mathbf{x}_i\}_{i=1}^n \sim p_{0|t}(\cdot|\tilde{\mathbf{x}}) p_0^{n-1}} \left[2 \left(\mathbf{s}_\theta(\mathbf{x}(t), t) - \sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right) \right].$$

Setting it to 0, we have

$$\mathbf{s}_\theta(\mathbf{x}(t), t) - \mathbb{E}_{\{\mathbf{x}_i\}_{i=1}^n \sim p_{0|t}(\cdot|\tilde{\mathbf{x}}) p_0^{n-1}} \left[\sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\mathbf{x}(t)} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right] = 0.$$

Thus, we arrive at the minimizer of the STF objective (Equation 3.6):

$$\mathbf{s}^*_{\mathrm{STF}}(\tilde{\mathbf{x}}, t) = \mathbb{E}_{\mathbf{x}_1 \sim p_{0|t}(\cdot|\tilde{\mathbf{x}})} \mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}} \left[\sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_j p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} \nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \right].$$

A.1.2 Proof for Theorem 1

Theorem 1. *Suppose* $\forall t \in [0,1], 0 < \sigma_t < \infty$, *then*

$$\sqrt{n} \left(\mathbf{s}^*_{STF}(\tilde{\mathbf{x}}, t) - \nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}}) \right) \xrightarrow{d} \mathcal{N} \left(\mathbf{0}, \frac{\mathrm{Cov}(\nabla_{\tilde{\mathbf{x}}} p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}))}{p_t(\tilde{\mathbf{x}})^2} \right) \tag{3.7}$$

Proof. Recall that $\mathbf{s}^*_{\mathrm{STF}}(\tilde{\mathbf{x}}, t)$ is calculated via Equation 3.6. The mean of the denominator in the expectation $\frac{1}{n} \sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)$, for large n, approximate $\frac{1}{n-1} \sum_{j=2}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)$, which in turn, $\frac{1}{n-1} \sum_{j=2}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j) \xrightarrow{p} p_t(\tilde{\mathbf{x}})$ by WLLN.

Similarly, for the mean of the remaining terms in the expectation, by CLT, we have

$$\sqrt{n} \left(\frac{1}{n} \sum_{k=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \nabla_{\tilde{\mathbf{x}}} \log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) - p_t(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log p_t(\tilde{\mathbf{x}}) \right) \xrightarrow{d} \mathcal{N} \left(\mathbf{0},\ \mathrm{Cov}(\nabla_{\tilde{\mathbf{x}}} p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})) \right)$$

Putting them together via Slutsky's theorem, we conclude the proof. □

A.1.3 Proof for Theorem 2

Theorem 2. *Suppose* $\forall t \in [0,1], 0 < \sigma_t < \infty$, *then*

$$V_{STF}(t) \leq \frac{1}{n-1}\left(V_{DSM}(t) + \frac{\sqrt{3}d}{\sigma_t^2}\sqrt{\mathbb{E}_{p_t(\tilde{\mathbf{x}})}D_f\left(p_0(\mathbf{x}) \parallel p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})\right)}\right) + O\left(\frac{1}{n^2}\right),$$

where D_f *is an f-divergence with* $f(y) = \begin{cases} (1/y-1)^2 & (y < 1.5) \\ 8y/27 - 1/3 & (y \geq 1.5) \end{cases}$. *Further, when* $n \gg d$ *and* $p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}}) \approx p_0(\mathbf{x})$ *for all* $\tilde{\mathbf{x}}$, $V_{STF}(t) \lessapprox \frac{V_{DSM}(t)}{n-1}$.

Proof. **Step 1: Make the likelihood weighting coefficients "independent"**

We first apply Hoeffding's inequality for the set $\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}$ to make the summation $\sum_{j=2}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)$ concentrate to its expectation $(n-1)p_t(\mathbf{x}(t))$. Since $p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j) \in (0, \frac{1}{(\sqrt{2\pi}\sigma_t)^d})$, we have

$$\Pr\left[\left|\sum_{j=2}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j) - (n-1)p_t(\mathbf{x}(t))\right| \geq n^{\gamma_1}\right] \leq 2e^{\frac{-2n^{2\gamma_1}(\sqrt{2\pi}\sigma_t)^d}{(n-1)}},$$

$\forall \gamma_1 \in (\frac{1}{2}, 1)$.

Thus the summation can be re-expressed as:

$$\begin{aligned}\sum_{j=2}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j) &= (1 - O(2e^{\frac{-2n^{2\gamma_1}(\sqrt{2\pi}\sigma_t)^d}{(n-1)}}))[(n-1)p_t(\mathbf{x}(t)) + O(n^{\gamma_1})] \\ &\quad + O(2e^{\frac{-2n^{2\gamma_1}(\sqrt{2\pi}\sigma_t)^d}{(n-1)}})O((n-1)\frac{1}{(\sqrt{2\pi}\sigma_t)^d}) \\ &= (n-1)p_t(\mathbf{x}(t)) + O(ne^{-2n^{2\gamma_1-1}(\sqrt{2\pi}\sigma_t)^d})\end{aligned}$$

The coefficient for $\mathbf{x}_1$ is then

$$\begin{aligned}\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_1)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} &= \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_1)}{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_1) + \sum_{j=2}^n p_{t|0}(\tilde{\mathbf{x}}|x_j)} \\ &= \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_1)}{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_1) + (n-1)p_t(\mathbf{x}(t)) + O(ne^{-2n^{2\gamma_1-1}(\sqrt{2\pi}\sigma_t)^d})} \\ &= O(\frac{1}{n})\end{aligned}$$

The coefficient for $x_k, k \in \{2, \ldots, n\}$ is:

$$\begin{aligned}\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)} &= \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{(n-1)p_t(\mathbf{x}(t)) + p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_1) + O(ne^{-2n^{2\gamma_1-1}(\sqrt{2\pi}\sigma_t)^d})} \\ &= \frac{1}{(n-1)}\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{p_t(\mathbf{x}(t))} + O(\frac{1}{n^2}) + O(ne^{-2n^{2\gamma_1-1}(\sqrt{2\pi}\sigma_t)^d})\end{aligned}$$

Step 2: Re-express the trace-of-covariance by the "independent" weights

Plugging in the above formulation of coefficients, we can rewrite the trace-of-covariance for the new target as:

$$\begin{aligned}&V_{\mathrm{STF}}(\mathbf{x}(t), t) \\ &= \mathbb{E}_{\mathbf{x}_1\sim p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n\sim p^{n-1}(\mathbf{x})}\|\sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|x_j)}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \\ &\qquad - \mathbb{E}_{\mathbf{x}_1\sim p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n\sim p^{n-1}(\mathbf{x})}\sum_{k=1}^n \frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{\sum_{j=1}^n p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_j)}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)\|_2^2 \\ &\leq \mathbb{E}_{\mathbf{x}_1\sim p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n\sim p^{n-1}(\mathbf{x})}\|\sum_{k=2}^n \frac{1}{(n-1)}\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \\ &\qquad - \mathbb{E}_{\mathbf{x}_1\sim p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n\sim p^{n-1}(\mathbf{x})}\sum_{k=2}^n \frac{1}{(n-1)}\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) \\ &\qquad + \sum_{k=2}^n O(\frac{1}{n^2})\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k) - O(\frac{1}{n})\mathbb{E}_{p(\mathbf{x})}[\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})]\|_2^2 + O(\frac{1}{n^2}) \\ &= \frac{1}{(n-1)^2}\sum_{k=2}^n \mathrm{Tr}\left(\mathrm{Cov}_{\mathbf{x}_k\sim p(\mathbf{x})}(\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k)}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}_k))\right) + O(\frac{1}{n^2}) \\ &= \frac{1}{(n-1)}\mathrm{Tr}\left(\mathrm{Cov}_{\mathbf{x}\sim p(\mathbf{x})}(\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}))\right) + O(\frac{1}{n^2}) \qquad \text{(A.1)}\end{aligned}$$

Step 3: Upper bound the new trace-of-covariance term

Next, we examine the new trace-of-covariance term:

$$
\begin{aligned}
&\operatorname{Tr}\left(\operatorname{Cov}_{\mathbf{x}\sim p_0(\mathbf{x})}\left(\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)\right) \\
=& \sum_{i=1}^{d}\mathbb{E}_{p_0(\mathbf{x})}\left[\left(\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)_i\right)^2\right] \\
&\qquad -\left(\mathbb{E}_{p_0(\mathbf{x})}\left[\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)_i\right]\right)^2 \\
=& \sum_{i=1}^{d}\mathbb{E}_{p_0(\mathbf{x})}\left[\left(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p_0(\mathbf{x})}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)_i\right)^2\right] \\
&\qquad -\left(\mathbb{E}_{p_0(\mathbf{x})}\left[\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p_0(\tilde{\mathbf{x}})}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)_i\right]\right)^2 \\
=& \sum_{i=1}^{d}\mathbb{E}_{p_0(\mathbf{x})}\left[\left(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p_0(\mathbf{x})}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)_i\right)^2\right] - \left(\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}})_i\right)^2 \\
=& \sum_{i=1}^{d}\mathbb{E}_{p_0(\mathbf{x})}\left[\left(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p_0(\mathbf{x})}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})\right)_i\right)^2\right] - \left(\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}})_i\right)^2 \\
&\qquad +\mathbb{E}_{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\left[\left(\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i\right)^2\right] - \mathbb{E}_{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}\left[\left(\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i\right)^2\right] \\
=& V_{\mathrm{DSM}}(\mathbf{x}(t),t) + \sum_{i=1}^{d}\mathbb{E}_{p_{0|t}}\left[\left(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p(\mathbf{x})}-1\right)\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i^2\right] \\
\leq& V_{\mathrm{DSM}}(\mathbf{x}(t),t) + \sum_{i=1}^{d}\sqrt{\mathbb{E}_{p_{0|t}}\left[\left(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p(\mathbf{x})}-1\right)^2\right]\mathbb{E}_{p_{0|t}}\left[\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i^4\right]} \qquad \text{(A.2)}
\end{aligned}
$$

We can further upper bound the trace-of-covariance term in Equation A.1:

$$
\begin{aligned}
&V_{\mathrm{STF}}(\mathbf{x}(t),t) \\
=& \frac{1}{(n-1)}\operatorname{Tr}(\operatorname{Cov}_{\mathbf{x}\sim p(\mathbf{x})}(\frac{p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})}{p_t(\mathbf{x}(t))}\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}))) + O(\frac{1}{n^2}) \\
\leq& \frac{1}{n-1}\left(V_{\mathrm{DSM}}(\mathbf{x}(t),t) + \sum_{i=1}^{d}\sqrt{\mathbb{E}_{p_{0|t}}\left[\left(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p(\mathbf{x})}-1\right)^2\right]\mathbb{E}_{p_{0|t}}\left[\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i^4\right]}\right) \\
&\qquad + O(\frac{1}{n^2})
\end{aligned}
$$

Taking the expectation w.r.t $p_t(\mathbf{x}(t))$ for both sides, we get

$$
\begin{aligned}
&V_{\mathrm{STF}}(t) \\
&\leq \mathbb{E}_{p_t(\mathbf{x}(t))}\Bigg[\frac{1}{n-1}\Bigg(V_{\mathrm{DSM}}(\mathbf{x}(t),t) \\
&\qquad + \sum_{i=1}^{d}\sqrt{\mathbb{E}_{p_{0|t}}\left[(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p(\mathbf{x})}-1)^2\right]\mathbb{E}_{p_{0|t}}\left[\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i^4\right]}\Bigg) + O(\frac{1}{n^2})\Bigg] \\
&\leq \frac{1}{n-1}\left(V_{\mathrm{DSM}}(t) + \sum_{i=1}^{d}\mathbb{E}_{p_t(\mathbf{x}(t))}\sqrt{\mathbb{E}_{p_{0|t}}\left[(\frac{p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})}{p(\mathbf{x})}-1)^2\right]\mathbb{E}_{p_{0|t}}\left[\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i^4\right]}\right) \\
&\qquad + O(\frac{1}{n^2}) \\
&\leq \frac{1}{n-1}\left(V_{\mathrm{DSM}}(t) + \sum_{i=1}^{d}\sqrt{\mathbb{E}_{p_t(\tilde{\mathbf{x}})}D_f\left(p_0(\mathbf{x})\parallel p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})\right)}\sqrt{\mathbb{E}_{p_{0,t}}\left[\nabla_{\mathbf{x}(t)}\log p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x})_i^4\right]}\right) \\
&\qquad + O(\frac{1}{n^2}) \qquad\qquad \text{(Concavity of } x^{\frac{1}{2}}\text{, Cauchy's inequality)} \\
&\leq \frac{1}{n-1}\left(V_{\mathrm{DSM}}(t) + d\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}[\frac{z^4}{\sigma_t^8}]}\sqrt{\mathbb{E}_{p_t(\tilde{\mathbf{x}})}D_f\left(p_0(\mathbf{x})\parallel p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})\right)}\right) + O(\frac{1}{n^2}) \\
&\leq \frac{1}{n-1}\left(V_{\mathrm{DSM}}(t) + \frac{\sqrt{3}d}{\sigma_t^2}\sqrt{\mathbb{E}_{p_t(\tilde{\mathbf{x}})}D_f\left(p_0(\mathbf{x})\parallel p_{0|t}(\mathbf{x}|\tilde{\mathbf{x}})\right)}\right) + O(\frac{1}{n^2})
\end{aligned}
$$

where D_f is an f-divergence with $f(y) = \begin{cases}(1/y-1)^2 & (y<1.5)\\ 8y/27-1/3 & (y\geq 1.5)\end{cases}$. Note that we choose this particular form of $f(y)$ since it is the convex function with the tightest upper bound on $(\frac{1}{y}-1)^2$. □

A.1.4 STF Specified with Popular SGMs

Here, we detail the practically used STF objectives in Section 3.5, which are built on the popular instances of SGMs, *e.g.*VE, VP [35], and EDM [27].

VE and EDM For VE and EDM, the transition kernel is in the form of

$$p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}\left(\mathbf{x},\ \sigma_t^2\boldsymbol{I}\right) \quad t\in[0,1].$$

VE has $\sigma_t = \sigma_m \left(\frac{\sigma_M}{\sigma_m}\right)^t$ for some fixed σ_m and σ_M. EDM has $\sigma_t = (t\sigma_M^{\frac{1}{\rho}} + (1-t)\sigma_m^{\frac{1}{\rho}})^\rho$ for some σ_m and σ_M, with ρ set to 7 in practice. The STF objective for both VE and EDM at $\tilde{\mathbf{x}}$ is then in the following form:

$$\mathbb{E}_{\mathbf{x}_1 \sim p_{0|t}(\cdot|\tilde{\mathbf{x}})} \mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \frac{1}{\sigma_t^2} \sum_{k=1}^n \frac{\exp\left(-\frac{\|\mathbf{x}(t)-\mathbf{x}_k\|_2^2}{2\sigma_t^2}\right)}{\sum_j \exp\left(-\frac{\|\mathbf{x}(t)-\mathbf{x}_j\|_2^2}{2\sigma_t^2}\right)} (\mathbf{x}_k - \mathbf{x}(t)) \right\|_2^2 \right].$$

VP VP in its original formulation has the transition kernel as

$$p_{t|0}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}\left(e^{-\frac{1}{4}t^2(\beta_M-\beta_m)-\frac{1}{2}t\beta_m}\mathbf{x},\ \boldsymbol{I} - \boldsymbol{I}e^{-\frac{1}{2}t^2(\beta_M-\beta_m)-t\beta_m}\right),$$

for some β_m and β_M. The STF objective for VP at $\tilde{\mathbf{x}}$ is

$$\mathbb{E}_{\mathbf{x}_1 \sim p_{0|t}} \mathbb{E}_{\{\mathbf{x}_i\}_{i=2}^n \sim p_0^{n-1}} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \frac{1}{\sigma_t^2} \sum_{k=1}^n \frac{\exp\left(-\frac{\|\mathbf{x}(t)-e^{\beta_t}\mathbf{x}_k\|_2^2}{2\sigma_t^2}\right)}{\sum_j \exp\left(-\frac{\|\mathbf{x}(t)-e^{\beta_t}\mathbf{x}_j\|_2^2}{2\sigma_t^2}\right)} \left(e^{\beta_t}\mathbf{x}_k - \mathbf{x}(t)\right) \right\|_2^2 \right],$$

where $\beta_t = -\frac{1}{4}t^2(\beta_M - \beta_m) - \frac{1}{2}t\beta_m$, and $\sigma_t = \sqrt{1 - e^{2\beta_t}}$. Note that as shown in [27], VP's transition kernel can be reparameterized in the form of $\mathcal{N}(\mathbf{x}, \sigma_t^2 \boldsymbol{I})$ with a correspondingly revised sampling process. Adopting this formulation, we would have the STF objective for VP the same as the one for VE and EDM with a different σ_t.

A.2 Chapter 5

In this section, we provide proof of our main results. Below, we define some crucial notations that we will use throughout. We use ODE(...) to denote the backward ODE under exact score $\nabla \log p_t(\mathbf{x})$. More specifically, given any $\mathbf{x} \in \mathbb{R}^d$ and $s > r > 0$, let $\mathbf{x}_t$ denote the solution to the following ODE:

$$d\mathbf{x}_t = -t\nabla \log p_t(\mathbf{x}_t)dt. \tag{A.3}$$

$\text{ODE}(\mathbf{x}, s \to r)$ is defined as "the value of $\mathbf{x}_r$ when initialized at $\mathbf{x}_s = \mathbf{x}$". It will also be useful to consider a "time-discretized ODE with drift $t\mathbf{s}_\theta(\mathbf{x}, t)$": let δ denote the discretization step size and let k denote any integer. Let δ denote a step size, let $\overline{\mathbf{x}}_t$ denote the solution to

$$d\overline{\mathbf{x}}_t = -t\mathbf{s}_\theta(\mathbf{x}_{k\delta}, k\delta)dt, \tag{A.4}$$

where for any t, k is the unique integer such that $t \in ((k-1)\delta, k\delta]$. We verify that the dynamics of Equation A.4 is equivalent to the following discrete-time dynamics for $t = k\delta, k \in \mathbb{Z}$:

$$\overline{\mathbf{x}}_{(k-1)\delta} = \overline{\mathbf{x}}_{k\delta} - \frac{1}{2}\left(((k-1)\delta)^2 - (k\delta)^2\right)\mathbf{s}_\theta(\mathbf{x}_{k\delta}, k\delta).$$

We similarly denote the value of $\overline{\mathbf{x}}_r$ when initialized at $\overline{\mathbf{x}}_s = x$ as $\text{ODE}_\theta(\mathbf{x}, s \to r)$. Analogously, we let $\text{SDE}(\mathbf{x}, s \to r)$ and $\text{SDE}_\theta(\mathbf{x}, s \to r)$ denote solutions to

$$\begin{aligned} d\mathbf{y}_t &= -2t\nabla \log p_t(\mathbf{y}_t)dt + \sqrt{2t}dB_t \\ d\overline{\mathbf{y}}_t &= -2t\mathbf{s}_\theta(\overline{\mathbf{y}}_t, t)dt + \sqrt{2t}dB_t \end{aligned}$$

respectively. Finally, we will define the Restart_θ process as follows:

$$\begin{aligned} &\text{(Restart_θ forward process)} \qquad \mathbf{x}^{i+1}_{t_{\max}} = \mathbf{x}^i_{t_{\min}} + \varepsilon^i_{t_{\min} \to t_{\max}} \\ &\text{(Restart_θ backward process)} \qquad \mathbf{x}^{i+1}_{t_{\min}} = \text{ODE}_\theta(\mathbf{x}^{i+1}_{t_{\max}}, t_{\max} \to t_{\min}), \end{aligned} \tag{A.5}$$

where $\varepsilon^i_{t_{\min}\to t_{\max}} \sim \mathcal{N}\left(\mathbf{0}, (t^2_{\max} - t^2_{\min})\, \boldsymbol{I}\right)$. We use $\text{Restart}_\theta(\mathbf{x}, K)$ to denote $\mathbf{x}^K_{t_{\min}}$ in the above processes, initialized at $\mathbf{x}^0_{t_{\min}} = x$. In various theorems, we will refer to a function $Q(r) : \mathbb{R}^+ \to [0, 1/2)$, defined as the Gaussian tail probability $Q(r) = Pr(a \geq r)$ for $a \sim \mathcal{N}(0, 1)$.

A.2.1 Proof for Theorem 3

Theorem 9. *[Formal version of Theorem 3] Let t_{max} be the initial noise level. Let the initial random variables $\overline{\mathbf{x}}_{t_{max}} = \overline{\mathbf{y}}_{t_{max}}$, and*

$$\overline{\mathbf{x}}_{t_{min}} = ODE_\theta(\overline{\mathbf{x}}_{t_{max}}, t_{max} \to t_{min})$$
$$\overline{\mathbf{y}}_{t_{min}} = SDE_\theta(\overline{\mathbf{y}}_{t_{max}}, t_{max} \to t_{min}),$$

Let p_t denote the true population distribution at noise level t. Let $p_t^{ODE_\theta}, p_t^{SDE_\theta}$ denote the distributions for $\mathbf{x}_t, y_t$ respectively. Assume that for all $\mathbf{x}, \mathbf{y}, s, t$, $\mathbf{s}_\theta(\mathbf{x}, t)$ satisfies $\|ts_\theta(\mathbf{x}, t) - t\mathbf{s}_\theta(\mathbf{x}, s)\| \leq L_0|s - t|$, $\|t\mathbf{s}_\theta(\mathbf{x}, t)\| \leq L_1$, $\|ts_\theta(\mathbf{x}, t) - t\mathbf{s}_\theta(\mathbf{y}, t)\| \leq L_2 \|\mathbf{x} - y\|$, and the approximation error $\|t\mathbf{s}_\theta(\mathbf{x}, t) - t\nabla \log p_t(\mathbf{x})\| \leq \epsilon_{approx}$. Assume in addition that $\forall t \in [t_{min}, t_{max}]$, $\|\mathbf{x}_t\| < B/2$ for any $\mathbf{x}_t$ in the support of p_t, $p_t^{ODE_\theta}$ or $p_t^{SDE_\theta}$, and $K \leq \frac{C}{L_2(t_{max} - t_{min})}$ for some universal constant C. Then

$$\begin{aligned} W_1(p^{ODE_\theta}_{t_{min}}, p_{t_{min}}) &\leq B \cdot TV\left(p^{ODE_\theta}_{t_{max}}, p_{t_{max}}\right) \\ &\quad + e^{L_2(t_{max} - t_{min})} \cdot \left(\delta(L_2 L_1 + L_0) + \epsilon_{approx}\right)(t_{max} - t_{min}) \qquad \text{(A.6)} \\ W_1(p^{SDE_\theta}_{t_{min}}, p_{t_{min}}) &\leq B \cdot \left(1 - \lambda e^{-BL_1/t_{min} - L_1^2 t^2_{max}/t^2_{min}}\right) TV(p^{SDE_\theta}_{t_{max}}, p_{t_{max}}) \\ &\quad + e^{2L_2(t_{max} - t_{min})}\left(\epsilon_{approx} + \delta L_0 + L_2\left(\delta L_1 + \sqrt{2\delta dt_{max}}\right)\right)(t_{max} - t_{min}) \qquad \text{(A.7)} \end{aligned}$$

where $\lambda := 2Q\left(\frac{B}{2\sqrt{t^2_{max} - t^2_{min}}}\right)$.

Proof. Let us define $\mathbf{x}_{t_{\max}} \sim p_{t_{\max}}$, and let $\mathbf{x}_{t_{\min}} = \text{ODE}(\mathbf{x}_{t_{\max}}, t_{\max} \to t_{\min})$. We verify that $\mathbf{x}_{t_{\min}}$ has density $p_{t_{\min}}$. Let us also define $\hat{\mathbf{x}}_{t_{\min}} = \text{ODE}_\theta(\mathbf{x}_{t_{\max}}, t_{\max} \to t_{\min})$. We would like to bound the Wasserstein distance between $\bar{\mathbf{x}}_{t_{\min}}$ and $\mathbf{x}_{t_{\min}}$ (*i.e.*, $p^{\text{ODE}_\theta}_{t_{\min}}$ and $p_{t_{\min}}$), by the

following triangular inequality:

$$W_1(\bar{\mathbf{x}}_{t_{\min}}, \mathbf{x}_{t_{\min}}) \leq W_1(\bar{\mathbf{x}}_{t_{\min}}, \hat{\mathbf{x}}_{t_{\min}}) + W_1(\hat{\mathbf{x}}_{t_{\min}}, \mathbf{x}_{t_{\min}}) \tag{A.8}$$

By Lemma 2, we know that

$$\|\hat{\mathbf{x}}_{t_{\min}} - \mathbf{x}_{t_{\min}}\| \leq e^{(t_{\max}-t_{\min})L_2} \left(\delta(L_2L_1 + L_0) + \epsilon_{approx}\right)(t_{\max} - t_{\min}),$$

where we use the fact that $\|\hat{\mathbf{x}}_{t_{\max}} - \mathbf{x}_{t_{\max}}\| = 0$. Thus, we immediately have

$$W_1(\hat{\mathbf{x}}_{t_{\min}}, \mathbf{x}_{t_{\min}}) \leq e^{(t_{\max}-t_{\min})L_2} \left(\delta(L_2L_1 + L_0) + \epsilon_{approx}\right)(t_{\max} - t_{\min}) \tag{A.9}$$

On the other hand,

$$\begin{aligned} W_1(\hat{\mathbf{x}}_{t_{\min}}, \overline{\mathbf{x}}_{t_{\min}}) \leq & B \cdot TV\left(\hat{\mathbf{x}}_{t_{\min}}, \overline{\mathbf{x}}_{t_{\min}}\right) \\ \leq & B \cdot TV\left(\hat{\mathbf{x}}_{t_{\max}}, \overline{\mathbf{x}}_{t_{\max}}\right) \end{aligned} \tag{A.10}$$

where the last equality is due to the data-processing inequality. Combining Equation A.9 , Equation A.10 and the triangular inequality Equation A.8, we arrive at the upper bound for ODE (Equation A.6). The upper bound for SDE (Equation A.7) shares a similar proof approach. First, let $\mathbf{y}_{t_{\max}} \sim p_{t_{\max}}$. Let $\hat{\mathbf{y}}_{t_{\min}} = \text{SDE}_\theta(\mathbf{y}_{t_{\max}}, t_{\max} \to t_{\min})$. By Lemma 5,

$$TV\left(\hat{\mathbf{y}}_{t_{\min}}, \overline{\mathbf{y}}_{t_{\min}}\right) \leq \left(1 - 2Q\left(\frac{B}{2\sqrt{t_{\max}^2 - t_{\min}^2}}\right) \cdot e^{-BL_1/t_{\min} - L_1^2 t_{\max}^2/t_{\min}^2}\right) \cdot TV\left(\hat{\mathbf{y}}_{t_{\max}}, \overline{\mathbf{y}}_{t_{\max}}\right)$$

On the other hand, by Lemma 4,

$$\mathbb{E}\left[\|\hat{\mathbf{y}}_{t_{\min}} - \mathbf{y}_{t_{\min}}\|\right] \leq e^{2L_2(t_{\max}-t_{\min})} \left(\epsilon_{approx} + \delta L_0 + L_2\left(\delta L_1 + \sqrt{2\delta d t_{\max}}\right)\right)(t_{\max} - t_{\min}).$$

The SDE triangular upper bound on $W_1(\bar{\mathbf{y}}_{t_{\min}}, \mathbf{y}_{t_{\min}})$ follows by multiplying the first inequality by B (to bound $W_1(\bar{\mathbf{y}}_{t_{\min}}, \hat{\mathbf{y}}_{t_{\min}})$) and then adding the second inequality (to bound $W_1(\mathbf{y}_{t_{\min}}, \hat{\mathbf{y}}_{t_{\min}})$). Notice that by definition, $TV\left(\hat{\mathbf{y}}_{t_{\max}}, \overline{\mathbf{y}}_{t_{\max}}\right) = TV\left(\mathbf{y}_{t_{\max}}, \overline{\mathbf{y}}_{t_{\max}}\right)$. Finally,

because of the assumption that $K \leq \frac{C}{L_2(t_{\max}-t_{\min})}$ for some universal constant, we summarize the second term in the Equation A.6 and Equation A.7 into the big O in the informal version Theorem 3. □

A.2.2 Proof for Theorem 4

Theorem 10. *[Formal version of Theorem 4] Consider the same setting as Theorem 9. Let* $p_{t_{min}}^{Restart_\theta,i}$ *denote the distributions after* i^{th} *Restart iteration,* i.e., *the distribution of* $\overline{\mathbf{x}}_{t_{min}}^i = Restart_\theta(\overline{\mathbf{x}}_{t_{min}}^0, i)$. *Given initial* $\overline{\mathbf{x}}_{t_{max}}^0 \sim p_{t_{max}}^{Restart,0}$, *let* $\overline{\mathbf{x}}_{t_{min}}^0 = ODE_\theta(\overline{\mathbf{x}}_{t_{max}}^0, t_{max} \to t_{min})$. *Then*

$$W_1(p_{t_{min}}^{Restart_\theta,K}, p_{t_{min}}) \leq \underbrace{B \cdot (1-\lambda)^K \, TV(p_{t_{max}}^{Restart,0}, p_{t_{max}})}_{\text{upper bound on contracted error}} + \underbrace{e^{(K+1)L_2(t_{max}-t_{min})}(K+1)\left(\delta(L_2L_1+L_0)+\epsilon_{approx}\right)(t_{max}-t_{min})}_{\text{upper bound on additional sampling error}} \tag{A.11}$$

where $\lambda = 2Q\left(\frac{B}{2\sqrt{t_{max}^2 - t_{min}^2}}\right)$.

Proof. Let $\mathbf{x}_{t_{\max}}^0 \sim p_{t_{\max}}$. Let $\mathbf{x}_{t_{\min}}^K = \text{Restart}(\mathbf{x}_{t_{\min}}^0, K)$. We verify that $\mathbf{x}_{t_{\min}}^K$ has density $p_{t_{\min}}$. Let us also define $\hat{\mathbf{x}}_{t_{\min}}^0 = \text{ODE}_\theta(\mathbf{x}_{t_{\max}}^0, t_{\max} \to t_{\min})$ and $\hat{\mathbf{x}}_{t_{\min}}^K = \text{Restart}_\theta(\hat{\mathbf{x}}_{t_{\min}}^0, K)$.

By Lemma 1,

$$\begin{aligned} TV\left(\overline{\mathbf{x}}_{t_{\min}}^K, \hat{\mathbf{x}}_{t_{\min}}^K\right) &\leq \left(1 - 2Q\left(\frac{B}{2\sqrt{t_{\max}^2 - t_{\min}^2}}\right)\right)^K TV\left(\overline{\mathbf{x}}_{t_{\min}}^0, \hat{\mathbf{x}}_{t_{\min}}^0\right) \\ &\leq \left(1 - 2Q\left(\frac{B}{2\sqrt{t_{\max}^2 - t_{\min}^2}}\right)\right)^K TV\left(\overline{\mathbf{x}}_{t_{\max}}^0, \hat{\mathbf{x}}_{t_{\max}}^0\right) \\ &= \left(1 - 2Q\left(\frac{B}{2\sqrt{t_{\max}^2 - t_{\min}^2}}\right)\right)^K TV\left(\overline{\mathbf{x}}_{t_{\max}}^0, \mathbf{x}_{t_{\max}}^0\right) \end{aligned}$$

The second inequality is held by data processing inequality. The above can be used to bound

the 1-Wasserstein distance as follows:

$$W_1\left(\overline{\mathbf{x}}^K_{t_{\min}}, \hat{\mathbf{x}}^K_{t_{\min}}\right) \leq B \cdot TV\left(\overline{\mathbf{x}}^K_{t_{\min}}, \hat{\mathbf{x}}^K_{t_{\min}}\right) \leq \left(1 - 2Q\left(\frac{B}{2\sqrt{t^2_{\max} - t^2_{\min}}}\right)\right)^K TV\left(\overline{\mathbf{x}}^0_{t_{\max}}, \mathbf{x}^0_{t_{\max}}\right) \tag{A.12}$$

On the other hand, using Lemma 3,

$$\begin{aligned} W_1\left(\mathbf{x}^K_{t_{\min}}, \hat{\mathbf{x}}^K_{t_{\min}}\right) \leq & \left\|\mathbf{x}^K_{t_{\min}} - \hat{\mathbf{x}}^K_{t_{\min}}\right\| \\ \leq & e^{(K+1)L_2(t_{\max}-t_{\min})}(K+1)\left(\delta(L_2L_1 + L_0) + \epsilon_{approx}\right)(t_{\max} - t_{\min}) \end{aligned} \tag{A.13}$$

We arrive at the result by combining the two bounds above (Equation A.12, Equation A.13) with the following triangular inequality,

$$W_1(\overline{\mathbf{x}}^K_{t_{\min}}, \mathbf{x}^K_{t_{\min}}) \leq W_1(\overline{\mathbf{x}}^K_{t_{\min}}, \hat{\mathbf{x}}^K_{t_{\min}}) + W_1(\hat{\mathbf{x}}^K_{t_{\min}}, \mathbf{x}^K_{t_{\min}})$$

□

Below we prove the lemmas used above.

Mixing under Restart with Exact ODE

Lemma 1. *Consider the same setup as Theorem 10. Consider the $Restart_\theta$ process defined in equation A.5. Let*

$$\begin{aligned} \mathbf{x}^i_{t_{min}} &= Restart_\theta(\mathbf{x}^0_{t_{min}}, i) \\ \mathbf{y}^i_{t_{min}} &= Restart_\theta(\mathbf{y}^0_{t_{min}}, i). \end{aligned}$$

Let $p_t^{Restart_\theta(i)}$ and $q_t^{Restart_\theta(i)}$ denote the densities of $\mathbf{x}^i_t$ and $\mathbf{y}^i_t$ respectively. Then

$$TV\left(p^{Restart_\theta(K)}_{t_{min}}, q^{Restart_\theta(K)}_{t_{min}}\right) \leq (1-\lambda)^K TV\left(p^{Restart_\theta(0)}_{t_{min}}, q^{Restart_\theta(0)}_{t_{min}}\right),$$

where $\lambda = 2Q\left(\frac{B}{2\sqrt{t^2_{max} - t^2_{min}}}\right)$.

Proof. Conditioned on $\mathbf{x}^i_{t_{\min}}, \mathbf{y}^i_{t_{\min}}$, let $\mathbf{x}^{i+1}_{t_{\max}} = \mathbf{x}^i_{t_{\min}} + \sqrt{t^2_{\max} - t^2_{\min}}\xi^{\mathbf{x}}_i$ and $\mathbf{y}^{i+1}_{t_{\max}} = \mathbf{y}^i_{t_{\min}} + \sqrt{t^2_{\max} - t^2_{\min}}\xi^{\mathbf{y}}_i$. We now define a coupling between $\mathbf{x}^{i+1}_{t_{\min}}$ and $\mathbf{y}^{i+1}_{t_{\min}}$ by specifying the joint distribution over $\xi^{\mathbf{x}}_i$ and $\xi^{\mathbf{y}}_i$.

If $\mathbf{x}^i_{t_{\min}} = y^i_{t_{\min}}$, let $\xi^{\mathbf{x}}_i = \xi^{\mathbf{y}}_i$, so that $\mathbf{x}^{i+1}_{t_{\min}} = y^{i+1}_{t_{\min}}$. On the other hand, if $\mathbf{x}^i_{t_{\min}} \neq y^i_{t_{\min}}$, let $\mathbf{x}^{i+1}_{t_{\max}}$ and $\mathbf{y}^{i+1}_{t_{\max}}$ be coupled as described in the proof of Lemma 7, with $\mathbf{x}' = \mathbf{x}^{i+1}_{t_{\max}}, \mathbf{y}' = \mathbf{y}^{i+1}_{t_{\max}}, \sigma = \sqrt{t^2_{\max} - t^2_{\min}}$. Under this coupling, we verify that,

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{1}\left\{\mathbf{x}^{i+1}_{t_{\min}} \neq \mathbf{y}^{i+1}_{t_{\min}}\right\}\right] \\
\leq &\mathbb{E}\left[\mathbb{1}\left\{\mathbf{x}^{i+1}_{t_{\max}} \neq \mathbf{y}^{i+1}_{t_{\max}}\right\}\right] \\
\leq &\mathbb{E}\left[\left(1 - 2Q\left(\frac{\left\|\mathbf{x}^i_{t_{\min}} - \mathbf{y}^i_{t_{\min}}\right\|}{2\sqrt{t^2_{\max} - t^2_{\min}}}\right)\right)\mathbb{1}\left\{\mathbf{x}^i_{t_{\min}} \neq \mathbf{y}^i_{t_{\min}}\right\}\right] \\
\leq &\left(1 - 2Q\left(\frac{B}{2\sqrt{t^2_{\max} - t^2_{\min}}}\right)\right)\mathbb{E}\left[\mathbb{1}\left\{\mathbf{x}^i_{t_{\min}} \neq \mathbf{y}^i_{t_{\min}}\right\}\right].
\end{aligned}$$

Applying the above recursively,

$$\mathbb{E}\left[\mathbb{1}\left\{\mathbf{x}^K_{t_{\min}} \neq \mathbf{y}^K_{t_{\min}}\right\}\right] \leq \left(1 - 2Q\left(\frac{B}{2\sqrt{t^2_{\max} - t^2_{\min}}}\right)\right)^K \mathbb{E}\left[\mathbb{1}\left\{\mathbf{x}^0_{t_{\min}} \neq \mathbf{y}^0_{t_{\min}}\right\}\right].$$

The conclusion follows by noticing that $TV\left(p^{\text{Restart}_\theta(K)}_{t_{\min}}, q^{\text{Restart}_\theta(K)}_{t_{\min}}\right) \leq Pr\left(\mathbf{x}^K_{t_{\min}} \neq \mathbf{y}^K_{t_{\min}}\right) = \mathbb{E}\left[\mathbb{1}\left\{\mathbf{x}^K_{t_{\min}} \neq \mathbf{y}^K_{t_{\min}}\right\}\right]$, and by selecting the initial coupling so that $Pr\left(\mathbf{x}^0_{t_{\min}} \neq \mathbf{y}^0_{t_{\min}}\right) = TV\left(p^{\text{Restart}_\theta(0)}_{t_{\min}}, q^{\text{Restart}_\theta(0)}_{t_{\min}}\right)$. □

W_1 Discretization Bound

Lemma 2 (Discretization bound for ODE)**.** *Let* $\mathbf{x}_{t_{min}} = ODE(\mathbf{x}_{t_{max}}, t_{max} \to t_{min})$ *and let* $\overline{\mathbf{x}}_{t_{min}} = ODE_\theta\left(\overline{\mathbf{x}}_{t_{max}}, t_{max} \to t_{min}\right)$*. Assume that for all* $\mathbf{x}, \mathbf{y}, s, t$, $\mathbf{s}_\theta(\mathbf{x}, t)$ *satisfies* $\|ts_\theta(\mathbf{x}, t) - t\mathbf{s}_\theta(\mathbf{x}, s)\| \leq L_0|s - t|$, $\|t\mathbf{s}_\theta(\mathbf{x}, t)\| \leq L_1$ *and* $\|ts_\theta(\mathbf{x}, t) - t\mathbf{s}_\theta(\mathbf{y}, t)\| \leq L_2\|\mathbf{x} - y\|$*. Then*

$$\|\mathbf{x}_{t_{min}} - \overline{\mathbf{x}}_{t_{min}}\| \leq e^{(t_{max} - t_{min})L_2}\left(\|\mathbf{x}_{t_{max}} - \overline{\mathbf{x}}_{t_{max}}\| + \left(\delta(L_2L_1 + L_0) + \epsilon_{approx}\right)(t_{max} - t_{min})\right)$$

Proof. Consider some fixed arbitrary k, and recall that δ is the step size. Recall that by

definition of ODE and ODE$_\theta$, for $t \in ((k-1)\delta, k\delta]$,

$$d\mathbf{x}_t = -t\nabla \log p_t(\mathbf{x}_t)dt$$
$$d\overline{\mathbf{x}}_t = -t\mathbf{s}_\theta(\overline{\mathbf{x}}_{k\delta}, k\delta)dt.$$

For $t \in [t_{\min}, t_{\max}]$, let us define a time-reversed process $\mathbf{x}_t^{\leftarrow} := \mathbf{x}_{-t}$. Let $v(\mathbf{x}, t) := \nabla \log p_{-t}(\mathbf{x})$. Then for $t \in [-t_{\max}, -t_{\min}]$

$$d\mathbf{x}_t^{\leftarrow} = tv(\mathbf{x}_t^{\leftarrow}, t)ds.$$

Similarly, define $\overline{\mathbf{x}}_t^{\leftarrow} := \overline{\mathbf{x}}_{-t}$ and $\overline{v}(\mathbf{x}, t) := \mathbf{s}_\theta(\mathbf{x}, -t)$. It follows that

$$d\overline{\mathbf{x}}_t^{\leftarrow} = t\overline{v}(\overline{\mathbf{x}}_{k\delta}^{\leftarrow}, k\delta)ds,$$

where k is the unique (negative) integer satisfying $t \in [k\delta, (k+1)\delta)$. Following these definitions,

$$\begin{aligned}
&\frac{d}{dt}\left\|\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}\right\| \\
\leq &\left\|tv(\mathbf{x}_t^{\leftarrow}, t) - t\overline{v}(\mathbf{x}_t^{\leftarrow}, t)\right\| \\
&\quad + \left\|t\overline{v}(\mathbf{x}_t^{\leftarrow}, t) - t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, t)\right\| \\
&\quad + \left\|t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, t) - t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, k\delta)\right\| \\
&\quad + \left\|t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, k\delta) - t\overline{v}(\overline{\mathbf{x}}_{k\delta}^{\leftarrow}, k\delta)\right\| \\
\leq &\epsilon_{approx} + L_2\left\|\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}\right\| + \delta L_0 + L_2\left\|\overline{\mathbf{x}}_t^{\leftarrow} - \overline{\mathbf{x}}_{k\delta}^{\leftarrow}\right\| \\
\leq &\epsilon_{approx} + L_2\left\|\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}\right\| + \delta L_0 + \delta L_2 L_1.
\end{aligned}$$

Applying Gronwall's Lemma over the interval $t \in [-t_{\max}, -t_{\min}]$,

$$\begin{aligned}
&\left\|\mathbf{x}_{t_{\min}} - \overline{\mathbf{x}}_{t_{\min}}\right\| \\
= &\left\|\mathbf{x}_{-t_{\min}}^{\leftarrow} - \overline{\mathbf{x}}_{-t_{\min}}^{\leftarrow}\right\| \\
\leq &e^{L_2(t_{\max}-t_{\min})}\left(\left\|\mathbf{x}_{-t_{\max}}^{\leftarrow} - \overline{\mathbf{x}}_{-t_{\max}}^{\leftarrow}\right\| + (\epsilon_{approx} + \delta L_0 + \delta L_2 L_1)(t_{\max} - t_{\min})\right) \\
= &e^{L_2(t_{\max}-t_{\min})}\left(\left\|\mathbf{x}_{t_{\max}} - \overline{\mathbf{x}}_{t_{\max}}\right\| + (\epsilon_{approx} + \delta L_0 + \delta L_2 L_1)(t_{\max} - t_{\min})\right).
\end{aligned}$$

□

Lemma 3. *Given initial* $\mathbf{x}^0_{t_{max}}$*, let* $\mathbf{x}^0_{t_{min}} = ODE\left(\mathbf{x}^0_{t_{max}}, t_{max} \to t_{min}\right)$*, and let* $\hat{\mathbf{x}}^0_{t_{min}} = ODE_\theta\left(\mathbf{x}^0_{t_{max}}, t_{max} \to t_{min}\right)$*. We further denote the variables after* K *Restart iterations as* $\mathbf{x}^K_{t_{min}} = Restart(\mathbf{x}^0_{t_{min}}, K)$ *and* $\hat{\mathbf{x}}^K_{t_{min}} = Restart_\theta(\hat{\mathbf{x}}^0_{t_{min}}, K)$*, with true field and learned field respectively. Then there exists a coupling between* $\mathbf{x}^K_{t_{min}}$ *and* $\hat{\mathbf{x}}^K_{t_{min}}$ *such that*

$$\left\|\mathbf{x}^K_{t_{min}} - \hat{\mathbf{x}}^K_{t_{min}}\right\| \leq e^{(K+1)L_2(t_{max}-t_{min})}(K+1)\left(\delta(L_2L_1+L_0)+\epsilon_{approx}\right)\left(t_{max}-t_{min}\right).$$

Proof. We will couple $\mathbf{x}^i_{t_{\min}}$ and $\hat{\mathbf{x}}^i_{t_{\min}}$ by using the same noise $\varepsilon^i_{t_{\min}\to t_{\max}}$ in the Restart forward process for $i = 0 \dots K-1$ (see Equation A.5). For any i, let us also define $\mathbf{y}^{i,j}_{t_{\min}} := \text{Restart}_\theta\left(\mathbf{x}^i_{t_{\min}}, j-i\right)$, and this process uses the same noise $\varepsilon^i_{t_{\min}\to t_{\max}}$ as previous ones. From this definition, $\mathbf{y}^{K,K}_{t_{\min}} = \mathbf{x}^K_{t_{\min}}$. We can thus bound

$$\left\|\mathbf{x}^K_{t_{\min}}, \hat{\mathbf{x}}^K_{t_{\min}}\right\| \leq \left\|\mathbf{y}^{0,K}_{t_{\min}} - \hat{\mathbf{x}}^K_{t_{\min}}\right\| + \sum_{i=0}^{K-1}\left\|\mathbf{y}^{i,K}_{t_{\min}} - y^{i+1,K}_{t_{\min}}\right\| \tag{A.14}$$

Using the assumption that $t\mathbf{s}_\theta(\cdot, t)$ is L_2 Lipschitz,

$$\begin{aligned}
&\left\|\mathbf{y}^{0,i+1}_{t_{\min}} - \hat{\mathbf{x}}^{i+1}_{t_{\min}}\right\| \\
=&\left\|\text{ODE}_\theta(\mathbf{y}^{0,i}_{t_{\max}}, t_{\max} \to t_{\min}) - \text{ODE}_\theta(\hat{\mathbf{x}}^i_{t_{\max}}, t_{\max} \to t_{\min})\right\| \\
\leq& e^{L_2(t_{\max}-t_{\min})}\left\|\mathbf{y}^{0,i}_{t_{\max}} - \hat{\mathbf{x}}^i_{t_{\max}}\right\| \\
=& e^{L_2(t_{\max}-t_{\min})}\left\|\mathbf{y}^{0,i}_{t_{\min}} - \hat{\mathbf{x}}^i_{t_{\min}}\right\|,
\end{aligned}$$

where the last equality is because we add the same additive Gaussian noise $\varepsilon^i_{t_{\min}\to t_{\max}}$ to $\mathbf{y}^{0,i}_{t_{\min}}$ and $\hat{\mathbf{x}}^i_{t_{\min}}$ in the Restart forward process. Applying the above recursively, we get

$$\begin{aligned}
\left\|\mathbf{y}^{0,K}_{t_{\min}} - \hat{\mathbf{x}}^K_{t_{\min}}\right\| \leq& e^{KL_2(t_{\max}-t_{\min})}\left\|\mathbf{y}^{0,0}_{t_{\min}} - \hat{\mathbf{x}}^0_{t_{\min}}\right\| \\
\leq& e^{KL_2(t_{\max}-t_{\min})}\left\|\mathbf{x}^0_{t_{\min}} - \hat{\mathbf{x}}^0_{t_{\min}}\right\| \\
\leq& e^{(K+1)L_2(t_{\max}-t_{\min})}\left(\delta(L_2L_1+L_0)+\epsilon_{approx}\right)\left(t_{\max}-t_{\min}\right),
\end{aligned} \tag{A.15}$$

where the last line follows by Lemma 2 when setting $\mathbf{x}_{t_{\max}} = \bar{\mathbf{x}}_{t_{\max}}$. We will now bound

$\left\|\mathbf{y}_{t_{\min}}^{i,K} - y_{t_{\min}}^{i+1,K}\right\|$ for some $i \leq K$. It follows from definition that

$$\begin{aligned}\mathbf{y}_{t_{\min}}^{i,i+1} &= \mathrm{ODE}_\theta\left(\mathbf{x}_{t_{\max}}^i, t_{\max} \to t_{\min}\right)\\ \mathbf{y}_{t_{\min}}^{i+1,i+1} &= \mathbf{x}_{t_{\min}}^{i+1} = \mathrm{ODE}\left(\mathbf{x}_{t_{\max}}^i, t_{\max} \to t_{\min}\right).\end{aligned}$$

By Lemma 2,

$$\left\|\mathbf{y}_{t_{\min}}^{i,i+1} - \mathbf{y}_{t_{\min}}^{i+1,i+1}\right\| \leq e^{L_2(t_{\max}-t_{\min})}\left(\delta(L_2L_1+L_0)+\epsilon_{approx}\right)(t_{\max}-t_{\min})$$

For the remaining steps from $i+2 \ldots K$, both $\mathbf{y}^{i,\cdot}$ and $\mathbf{y}^{i+1,\cdot}$ evolve with ODE_θ in each step. Again using the assumption that $t\mathbf{s}_\theta(\cdot,t)$ is L_2 Lipschitz,

$$\left\|\mathbf{y}_{t_{\min}}^{i,K} - \mathbf{y}_{t_{\min}}^{i+1,K}\right\| \leq e^{(K-i)L_2(t_{\max}-t_{\min})}\left(\delta(L_2L_1+L_0)+\epsilon_{approx}\right)(t_{\max}-t_{\min})$$

Summing the above for $i = 0 \ldots K-1$, and combining with Equation A.14 and Equation A.15 gives

$$\left\|\mathbf{x}_{t_{\min}}^K - \hat{\mathbf{x}}_{t_{\min}}^K\right\| \leq e^{(K+1)L_2(t_{\max}-t_{\min})}(K+1)\left(\delta(L_2L_1+L_0)+\epsilon_{approx}\right)(t_{\max}-t_{\min}).$$

□

Lemma 4. *Consider the same setup as Theorem 9. Let* $\mathbf{x}_{t_{min}} = SDE(\mathbf{x}_{t_{max}}, t_{max} \to t_{min})$ *and let* $\overline{\mathbf{x}}_{t_{min}} = SDE(\overline{\mathbf{x}}_{t_{max}}, t_{max} \to t_{min})$*. Then there exists a coupling between* $\mathbf{x}_t$ *and* $\overline{\mathbf{x}}_t$ *such that*

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{x}_{t_{min}} - \overline{\mathbf{x}}_{t_{min}}\|\right] \leq\ & e^{2L_2(t_{max}-t_{min})}\mathbb{E}\left[\|\mathbf{x}_{t_{max}} - \overline{\mathbf{x}}_{t_{max}}\|\right]\\ &+ e^{2L_2(t_{max}-t_{min})}\left(\epsilon_{approx} + \delta L_0 + L_2\left(\delta L_1 + \sqrt{2\delta dt_{max}}\right)\right)(t_{max}-t_{min})\end{aligned}$$

Proof. Consider some fixed arbitrary k, and recall that δ is the stepsize. By definition of SDE and SDE_θ, for $t \in ((k-1)\delta, k\delta]$,

$$\begin{aligned}d\mathbf{x}_t &= -2t\nabla \log p_t(\mathbf{x}_t)dt + \sqrt{2t}dB_t\\ d\overline{\mathbf{x}}_t &= -2t\mathbf{s}_\theta(\overline{\mathbf{x}}_{k\delta}, k\delta)dt + \sqrt{2t}dB_t.\end{aligned}$$

Let us define a coupling between $\mathbf{x}_t$ and $\overline{\mathbf{x}}_t$ by identifying their respective Brownian motions. It will be convenient to define the time-reversed processes $\mathbf{x}_t^{\leftarrow} := \mathbf{x}_{-t}$, and $\overline{\mathbf{x}}_t^{\leftarrow} := \overline{\mathbf{x}}_{-t}$, along with $v(\mathbf{x}, t) := \nabla \log p_{-t}(\mathbf{x})$ and $\overline{v}(\mathbf{x}, t) := \mathbf{s}_\theta(\mathbf{x}, -t)$. Then there exists a Brownian motion $B_t^{\leftarrow}$, such that for $t \in [-t_{\max}, -t_{\min}]$,

$$\begin{aligned}
& d\mathbf{x}_t^{\leftarrow} = -2tv(\mathbf{x}_t^{\leftarrow}, t)dt + \sqrt{-2t}dB_t^{\leftarrow} \\
& d\overline{\mathbf{x}}_t^{\leftarrow} = -2t\overline{v}(\overline{\mathbf{x}}_{k\delta}^{\leftarrow}, k\delta)dt + \sqrt{-2t}dB_t^{\leftarrow} \\
\Rightarrow \qquad & d(\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}) = -2t\left(v(\mathbf{x}_t^{\leftarrow}, t) - \overline{v}(\overline{\mathbf{x}}_{k\delta}^{\leftarrow}, k\delta)\right)dt,
\end{aligned}$$

where k is the unique negative integer such that $t \in [k\delta, (k+1)\delta)$. Thus

$$\begin{aligned}
& \frac{d}{dt}\mathbb{E}\left[\|\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}\|\right] \\
\leq & 2\left(\mathbb{E}\left[\|tv(\mathbf{x}_t^{\leftarrow}, t) - t\overline{v}(\mathbf{x}_t^{\leftarrow}, t)\|\right] + \mathbb{E}\left[\|t\overline{v}(\mathbf{x}_t^{\leftarrow}, t) - t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, t)\|\right]\right) \\
& \quad + 2\left(\mathbb{E}\left[\|t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, t) - t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, k\delta)\|\right] + \mathbb{E}\left[\|t\overline{v}(\overline{\mathbf{x}}_t^{\leftarrow}, k\delta) - t\overline{v}(\overline{\mathbf{x}}_{k\delta}^{\leftarrow}, k\delta)\|\right]\right) \\
\leq & 2\left(\epsilon_{approx} + L_2\mathbb{E}\left[\|\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}\|\right] + \delta L_0 + L_2\mathbb{E}\left[\|\overline{\mathbf{x}}_t^{\leftarrow} - \overline{\mathbf{x}}_{k\delta}^{\leftarrow}\|\right]\right) \\
\leq & 2\left(\epsilon_{approx} + L_2\mathbb{E}\left[\|\mathbf{x}_t^{\leftarrow} - \overline{\mathbf{x}}_t^{\leftarrow}\|\right] + \delta L_0 + L_2\left(\delta L_1 + \sqrt{2\delta dt_{\max}}\right)\right).
\end{aligned}$$

By Gronwall's Lemma,

$$\begin{aligned}
& \mathbb{E}\left[\|\mathbf{x}_{t_{\min}} - \overline{\mathbf{x}}_{t_{\min}}\|\right] \\
= & \mathbb{E}\left[\left\|\mathbf{x}_{-t_{\min}}^{\leftarrow} - \overline{\mathbf{x}}_{-t_{\min}}^{\leftarrow}\right\|\right] \\
\leq & e^{2L_2(t_{\max} - t_{\min})} \\
& \quad \left(\mathbb{E}\left[\left\|\mathbf{x}_{-t_{\max}}^{\leftarrow} - \overline{\mathbf{x}}_{-t_{\max}}^{\leftarrow}\right\|\right] + \left(\epsilon_{approx} + \delta L_0 + L_2\left(\delta L_1 + \sqrt{2\delta dt_{\max}}\right)\right)(t_{\max} - t_{\min})\right) \\
= & e^{2L_2(t_{\max} - t_{\min})} \\
& \quad \left(\mathbb{E}\left[\|\mathbf{x}_{t_{\max}} - \overline{\mathbf{x}}_{t_{\max}}\|\right] + \left(\epsilon_{approx} + \delta L_0 + L_2\left(\delta L_1 + \sqrt{2\delta dt_{\max}}\right)\right)(t_{\max} - t_{\min})\right)
\end{aligned}$$

□

Mixing Bounds

Lemma 5. *Consider the same setup as Theorem 9. Assume that $\delta \leq t_{min}$. Let*

$$\mathbf{x}_{t_{min}} = SDE_\theta\left(\mathbf{x}_{t_{max}}, t_{max} \to t_{min}\right)$$
$$\mathbf{y}_{t_{min}} = SDE_\theta\left(\mathbf{y}_{t_{max}}, t_{max} \to t_{min}\right).$$

Then there exists a coupling between $\mathbf{x}_s$ and $\mathbf{y}_s$ such that

$$TV\left(\mathbf{x}_{t_{min}}, \mathbf{y}_{t_{min}}\right) \leq \left(1 - 2Q\left(\frac{B}{2\sqrt{t_{max}^2 - t_{min}^2}}\right) \cdot e^{-BL_1/t_{min} - L_1^2 t_{max}^2/t_{min}^2}\right) TV\left(\mathbf{x}_{t_{max}}, \mathbf{y}_{t_{max}}\right)$$

Proof. We will construct a coupling between $\mathbf{x}_t$ and $\mathbf{y}_t$. First, let $(\mathbf{x}_{t_{\max}}, \mathbf{y}_{t_{\max}})$ be sampled from the optimal TV coupling, *i.e.*, $Pr(\mathbf{x}_{t_{\max}} \neq \mathbf{y}_{t_{\max}}) = TV(\mathbf{x}_{t_{\max}}, \mathbf{y}_{t_{\max}})$. Recall that by definition of SDE_θ, for $t \in ((k-1)\delta, k\delta]$,

$$d\mathbf{x}_t = -2t\mathbf{s}_\theta(\mathbf{x}_{k\delta}, k\delta)dt + \sqrt{2t}dB_t.$$

Let us define a time-rescaled version of $\mathbf{x}_t$: $\overline{\mathbf{x}}_t := \mathbf{x}_{t^2}$. We verify that

$$d\overline{\mathbf{x}}_t = -\mathbf{s}_\theta(\overline{\mathbf{x}}_{(k\delta)^2}, k\delta)dt + dB_t,$$

where k is the unique integer satisfying $t \in [((k-1)\delta)^2, k^2\delta^2)$. Next, we define the time-reversed process $\overline{\mathbf{x}}_t^{\leftarrow} := \overline{\mathbf{x}}_{-t}$, and let $v(\mathbf{x}, t) := \mathbf{s}_\theta(\mathbf{x}, -t)$. We verify that there exists a Brownian motion $B_t^{\mathbf{x}}$ such that, for $t \in [-t_{\max}^2, -t_{\min}^2]$,

$$d\overline{\mathbf{x}}_t^{\leftarrow} = v_t^{\mathbf{x}}dt + dB_t^{\mathbf{x}},$$

where $v_t^{\mathbf{x}} = \mathbf{s}_\theta(\overline{\mathbf{x}}_{-(k\delta)^2}^{\leftarrow}, -k\delta)$, where k is the unique positive integer satisfying $-t \in (((k-1)\delta)^2, (k\delta)^2]$. Let $d\overline{\mathbf{y}}_t^{\leftarrow} = v_t^{\mathbf{y}}dt + dB_t^{\mathbf{y}}$, be defined analogously. For any positive integer k and

for any $t \in [-(k\delta)^2, -((k-1)\delta)^2)$, let us define

$$z_t = \overline{\mathbf{x}}^{\leftarrow}_{-k^2\delta^2} - \overline{\mathbf{y}}^{\leftarrow}_{-k^2\delta^2} + (2k-1)\delta^2\left(v^{\mathbf{x}}_{-(k\delta)^2} - v^{\mathbf{y}}_{-(k\delta)^2}\right) + \left(B^{\mathbf{x}}_t - B^{\mathbf{x}}_{-(k\delta)^2}\right) - \left(B^{\mathbf{y}}_t - B^{\mathbf{y}}_{-(k\delta)^2}\right).$$

Let $\gamma_t := \frac{z_t}{\|z_t\|}$. We will now define a coupling between $dB^{\mathbf{x}}_t$ and $dB^{\mathbf{y}}_t$ as

$$dB^{\mathbf{y}}_t = \left(I - 2\mathbb{1}\left\{t \le \tau\right\}\gamma_t\gamma_t^T\right) dB^{\mathbf{x}}_t,$$

where $\mathbb{1}\{\}$ denotes the indicator function, i.e. $\mathbb{1}\{t \le \tau\} = 1$ if $t \le \tau$, and τ is a stopping time given by the first hitting time of $z_t = 0$. Let $r_t := \|z_t\|$. Consider some $t \in (-i^2\delta^2, -(i-1)^2\delta^2)$, and Let $j := \frac{t_{\max}}{\delta}$ (assume w.l.o.g that this is an integer), then

$$\begin{aligned}
r_t - r_{-t^2_{\max}} &\le \sum_{k=i}^{j}(2k-1)\delta^2\left\|(v^{\mathbf{x}}_{-(k\delta)^2} - v^{\mathbf{y}}_{-(k\delta)^2})\right\| + \int_{-t^2_{\max}}^{t}\mathbb{1}\left\{t \le \tau\right\}2dB^1_s \\
&\le \sum_{k=i}^{j}\left(k^2 - (k-1)^2\right)\delta^2 2L_1/\left(t_{\min}\right) + \int_{-t^2_{\max}}^{t}\mathbb{1}\left\{t \le \tau\right\}2dB^1_t \\
&= \int_{-t^2_{\max}}^{-(i-1)\delta^2}\frac{2L_1}{t_{\min}}ds + \int_{-t^2_{\max}}^{t}\mathbb{1}\left\{t \le \tau\right\}2dB^1_s,
\end{aligned}$$

where $dB^1_s = \langle\gamma_t, dB^{\mathbf{x}}_s - dB^{\mathbf{y}}_s\rangle$ is a 1-dimensional Brownian motion. We also verify that

$$\begin{aligned}
r_{-t^2_{\max}} &= \left\|z_{-t^2_{\max}}\right\| \\
&= \left\|\overline{\mathbf{x}}^{\leftarrow}_{-t^2_{\max}} - \overline{\mathbf{y}}^{\leftarrow}_{-t^2_{\max}} + (2j-1)\delta^2\left(v^{\mathbf{x}}_{-t^2_{\max}} - v^{\mathbf{y}}_{-t^2_{\max}}\right) + \left(B^{\mathbf{x}}_t - B^{\mathbf{x}}_{-t^2_{\max}}\right) - \left(B^{\mathbf{y}}_t - B^{\mathbf{y}}_{-t^2_{\max}}\right)\right\| \\
&\le \left\|\overline{\mathbf{x}}^{\leftarrow}_{-t^2_{\max}} + (2j-1)\delta^2 v^{\mathbf{x}}_{-t^2_{\max}} + \left(B^{\mathbf{x}}_{-(j-1)^2\delta^2} - B^{\mathbf{x}}_{-t^2_{\max}}\right)\right\| \\
&\quad + \left\|\overline{\mathbf{y}}^{\leftarrow}_{-t^2_{\max}} + (2j-1)\delta^2 v^{\mathbf{y}}_{-t^2_{\max}} + \left(B^{\mathbf{x}}_{-(j-1)^2\delta^2} - B^{\mathbf{x}}_t + B^{\mathbf{y}}_t - B^{\mathbf{y}}_{-t^2_{\max}}\right)\right\| \le B
\end{aligned}$$

where the third relation is by adding and subtracting $B^{\mathbf{x}}_{-(j-1)^2\delta^2} - B^{\mathbf{x}}_t$ and using triangle inequality. The fourth relation is by noticing that $\overline{\mathbf{x}}^{\leftarrow}_{-t^2_{\max}} + (2j-1)\delta^2 v^{\mathbf{x}}_{-t^2_{\max}} + \left(B^{\mathbf{x}}_{-(j-1)^2\delta^2} - B^{\mathbf{x}}_{-t^2_{\max}}\right) = \overline{\mathbf{x}}^{\leftarrow}_{-(j-1)^2\delta^2}$ and that $\overline{\mathbf{y}}^{\leftarrow}_{-t^2_{\max}} + (2j-1)\delta^2 v^{\mathbf{y}}_{-t^2_{\max}} + \left(B^{\mathbf{x}}_{-(j-1)^2\delta^2} - B^{\mathbf{x}}_t + B^{\mathbf{y}}_t - B^{\mathbf{y}}_{-t^2_{\max}}\right) \overset{d}{=} \overline{\mathbf{y}}^{\leftarrow}_{-(j-1)^2\delta^2}$, and then using our assumption in the theorem statement that all processes are supported on a ball of radius $B/2$.

We now define a process s_t defined by $ds_t = 2L_1/t_{\min}dt + 2dB^1_t$, initialized at $s_{-t^2_{\max}} =$

$B \geq r_{-t^2_{\max}}$. We can verify that, up to time τ, $r_t \leq s_t$ with probability 1. Let τ' denote the first-hitting time of s_t to 0, then $\tau \leq \tau'$ with probability 1. Thus

$$Pr(\tau \leq -t^2_{\min}) \geq Pr(\tau' \leq -t^2_{\min}) \geq 2Q\left(\frac{B}{2\sqrt{t^2_{\max} - t^2_{\min}}}\right) \cdot e^{-BL_1/t_{\min} - L_1^2 t^2_{\max}/t^2_{\min}}$$

where we apply Lemma 6. The proof follows by noticing that, if $\tau \leq -t^2_{\min}$, then $\mathbf{x}_{t_{\min}} = y_{t_{\min}}$. This is because if $\tau \in [-k^2\delta^2, -(k-1)^2\delta^2]$, then $\overleftarrow{\mathbf{x}}_{-(k-1)^2\delta^2} = \overleftarrow{\mathbf{y}}_{-(k-1)^2\delta^2}$, and thus $\overleftarrow{\mathbf{x}}_t = \overleftarrow{\mathbf{y}}_t$ for all $t \geq -(k-1)^2\delta^2$, in particular, at $t = -t^2_{\min}$.

□

Lemma 6. *Consider the stochastic process*

$$dr_t = dB^1_t + cdt.$$

Assume that $r_0 \leq B/2$. Let τ denote the hitting time for $r_t = 0$. Then for any $T \in \mathbb{R}^+$,

$$Pr(\tau \leq T) \geq 2Q\left(\frac{B}{2\sqrt{T}}\right) \cdot e^{-ac - \frac{c^2 T}{2}},$$

where Q is the tail probability of a standard Gaussian defined in Definition 1.

Proof. We will use he following facts in our proof:

1. For $\mathbf{x} \sim \mathcal{N}(0, \sigma^2)$, $Pr(\mathbf{x} > r) = \frac{1}{2}\left(1 - erf\left(\frac{r}{\sqrt{2}\sigma}\right)\right) = \frac{1}{2} erfc\left(\frac{r}{\sqrt{2}\sigma}\right)$.
2. $\int_0^T \frac{a \exp\left(-\frac{a^2}{2t}\right)}{\sqrt{2\pi t^3}} dt = erfc\left(\frac{a}{\sqrt{2T}}\right) = 2Pr\left(\mathcal{N}(0,T) > a\right) = 2Q\left(\frac{a}{\sqrt{T}}\right)$ by definition of Q.

Let $dr_t = dB^1_t + cdt$, with $r_0 = a$. The density of the hitting time τ is given by

$$p(\tau = t) = f(a, c, t) = \frac{a \exp\left(-\frac{(a+ct)^2}{2t}\right)}{\sqrt{2\pi t^3}}. \tag{A.16}$$

(see e.g. [243]). From item 2 above,

$$\int_0^T f(a, 0, t) dt = 2Q\left(\frac{a}{\sqrt{T}}\right).$$

In the case of a general $c \neq 0$, we can bound $\frac{(a+ct)^2}{2t} = \frac{a^2}{2t} + ac + \frac{c^2 t}{2}$. Consequently,

$$f(a, c, t) \geq f(a, 0, t) \cdot e^{-ac - \frac{c^2 t}{2}}.$$

Therefore,

$$Pr(\tau \leq T) = \int_0^T f(a, c, t)dt \geq \int_0^T f(a, 0, t)dt e^{-c} = 2Q\left(\frac{B}{2\sqrt{T}}\right) \cdot e^{-ac - \frac{c^2 T}{2}}.$$

□

TV Overlap

Definition 1. *Let* $\mathbf{x}$ *be sampled from standard normal distribution* $\mathcal{N}(0, 1)$*. We define the Gaussian tail probability* $Q(a) := Pr(\mathbf{x} \geq a)$*.*

Lemma 7. *We verify that for any two random vectors* $\xi_{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ *and* $\xi_{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$*, each belonging to* $\mathbb{R}^d$*, the total variation distance between* $\mathbf{x}' = \mathbf{x} + \xi_{\mathbf{x}}$ *and* $\mathbf{y}' = \mathbf{y} + \xi_{\mathbf{y}}$ *is given by*

$$TV(\mathbf{x}', \mathbf{y}') = 1 - 2Q\left(r\right) \leq 1 - \frac{2r}{r^2 + 1} \frac{1}{\sqrt{2\pi}} e^{-r^2/2},$$

where $r = \frac{\|\mathbf{x}-\mathbf{y}\|}{2\sigma}$*, and* $Q(r) = Pr(\xi \geq r)$*, when* $\xi \sim \mathcal{N}(0, 1)$*.*

Proof. Let $\gamma := \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}$. We decompose $\mathbf{x}', \mathbf{y}'$ into the subspace/orthogonal space defined by γ:

$$\mathbf{x}' = \mathbf{x}^{\perp} + \xi_{\mathbf{x}}^{\perp} + \mathbf{x}^{\parallel} + \xi_{\mathbf{x}}^{\parallel}$$
$$\mathbf{y}' = \mathbf{y}^{\perp} + \xi_{\mathbf{y}}^{\perp} + \mathbf{y}^{\parallel} + \xi_{\mathbf{y}}^{\parallel}$$

where we define

$$\begin{aligned}
\mathbf{x}^{\parallel} &:= \gamma\gamma^T\mathbf{x} \qquad & \mathbf{x}^{\perp} &:= \mathbf{x} - \mathbf{x}^{\parallel} \\
\mathbf{y}^{\parallel} &:= \gamma\gamma^T\mathbf{y} \qquad & \mathbf{y}^{\perp} &:= \mathbf{y} - \mathbf{y}^{\parallel} \\
\xi_{\mathbf{x}}^{\parallel} &:= \gamma\gamma^T\xi_{\mathbf{x}} \qquad & \xi_{\mathbf{x}}^{\perp} &:= \xi_{\mathbf{x}} - \xi_{\mathbf{x}}^{\parallel} \\
\xi_{\mathbf{y}}^{\parallel} &:= \gamma\gamma^T\xi_{\mathbf{y}} \qquad & \xi_{\mathbf{y}}^{\perp} &:= \xi_{\mathbf{y}} - \xi_{\mathbf{y}}^{\parallel}
\end{aligned}$$

We verify the independence $\xi_{\mathbf{x}}^{\perp} \perp\!\!\!\perp \xi_{\mathbf{x}}^{\parallel}$ and $\xi_{\mathbf{y}}^{\perp} \perp\!\!\!\perp \xi_{\mathbf{y}}^{\parallel}$ as they are orthogonal decompositions of the standard Gaussian. We will define a coupling between $\mathbf{x}'$ and $\mathbf{y}'$ by setting $\xi_{\mathbf{x}}^{\perp} = \xi_{\mathbf{y}}^{\perp}$. Under this coupling, we verify that

$$\left(\mathbf{x}^{\perp} + \xi_{\mathbf{x}}^{\perp}\right) - \left(\mathbf{y}^{\perp} + \xi_{\mathbf{y}}^{\perp}\right) = \mathbf{x} - \mathbf{y} - \gamma\gamma^T(\mathbf{x} - \mathbf{y}) = 0$$

Therefore, $\mathbf{x}' = \mathbf{y}'$ if and only if $\mathbf{x}^{\parallel} + \xi_{\mathbf{x}}^{\parallel} = \mathbf{y}^{\parallel} + \xi_{\mathbf{y}}^{\parallel}$. Next, we draw (a, b) from the optimal coupling between $\mathcal{N}(0,1)$ and $\mathcal{N}(\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}, 1)$. We verify that $\mathbf{x}^{\parallel} + \xi_{\mathbf{x}}^{\parallel}$ and $\mathbf{y}^{\parallel} + \xi_{\mathbf{y}}^{\parallel}$ both lie in the span of γ. Thus it suffices to compare $\left\langle \gamma, \mathbf{x}^{\parallel} + \xi_{\mathbf{x}}^{\parallel} \right\rangle$ and $\left\langle \gamma, \mathbf{y}^{\parallel} + \xi_{\mathbf{y}}^{\parallel} \right\rangle$. We verify that $\left\langle \gamma, \mathbf{x}^{\parallel} + \xi_{\mathbf{x}}^{\parallel} \right\rangle = \left\langle \gamma, \mathbf{y}^{\parallel} \right\rangle + \left\langle \gamma, \mathbf{x}^{\parallel} - \mathbf{y}^{\parallel} \right\rangle + \left\langle \gamma, \xi_{\mathbf{x}}^{\parallel} \right\rangle \sim \mathcal{N}(\left\langle \gamma, \mathbf{y}^{\parallel} \right\rangle + \|\mathbf{x} - \mathbf{y}\|, \sigma^2) \stackrel{d}{=} \left\langle \gamma, \mathbf{y}^{\parallel} \right\rangle + \sigma b$. We similarly verify that $\left\langle \gamma, \mathbf{y}^{\parallel} + \xi_{\mathbf{y}}^{\parallel} \right\rangle = \left\langle \gamma, \mathbf{y}^{\parallel} \right\rangle + \left\langle \gamma, \xi_{\mathbf{y}}^{\parallel} \right\rangle \sim \mathcal{N}(\left\langle \gamma, \mathbf{y}^{\parallel} \right\rangle, \sigma^2) \stackrel{d}{=} \left\langle \gamma, \mathbf{y}^{\parallel} \right\rangle + \sigma a$.

Thus $TV(\mathbf{x}', \mathbf{y}') = TV(\sigma a, \sigma b) = 1 - 2Q\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{2\sigma}\right)$. The last inequality follows from

$$Pr(\mathcal{N}(0,1) \geq r) \geq \frac{r}{r^2+1}\frac{1}{\sqrt{2\pi}}e^{-r^2/2}$$

□

A.2.3 Heun's method as DPM-Solver-2

The first order ODE in DPM-Solver [26] (DPM-Solver-1) is in the form of:

$$\hat{\mathbf{x}}_{t_{i-1}} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\hat{\mathbf{x}}_{t_{i-1}} - (\hat{\sigma}_{t_{i-1}}\frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} - \hat{\sigma}_{t_i})\hat{\sigma}_{t_i}\nabla_{\mathbf{x}}\log p_{\hat{\sigma}_{t_i}}(\hat{\mathbf{x}}_{t_i}) \tag{A.17}$$

The first order ODE in EDM is in the form of

$$\mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_i} - (\sigma_{t_{i-1}} - \sigma_{t_i})\sigma_{t_i}\nabla_{\mathbf{x}} \log p_{\sigma_{t_i}}(\mathbf{x}_{t_i}) \tag{A.18}$$

When $\mathbf{x}_t = \frac{\hat{\mathbf{x}}_t}{\alpha_t}$, $\hat{\sigma}_t = \sigma_t \alpha_t$, we can rewrite the DPM-Solver-1 (Equation A.17) as:

$$\begin{aligned}
\mathbf{x}_{t_{i-1}} &= \mathbf{x}_{t_i} - (\sigma_{t_{i-1}} - \sigma_{t_i})\hat{\sigma}_{t_i}\nabla_{\mathbf{x}} \log p_{\hat{\sigma}_{t_i}}(\hat{\mathbf{x}}_{t_i}) \\
&= \mathbf{x}_{t_i} - (\sigma_{t_{i-1}} - \sigma_{t_i})\hat{\sigma}_{t_i}\nabla_{\mathbf{x}} \log p_{\sigma_{t_i}}(\mathbf{x}_{t_i})\frac{1}{\alpha_{t_i}} \qquad \text{(change-of-variable)} \\
&= \mathbf{x}_{t_i} - (\sigma_{t_{i-1}} - \sigma_{t_i})\sigma_{t_i}\nabla_{\mathbf{x}} \log p_{\sigma_{t_i}}(\mathbf{x}_{t_i})
\end{aligned}$$

where the expression is exactly the same as the ODE in EDM [27]. It indicates that the sampling trajectory in DPM-Solver-1 is equivalent to the one in EDM, up to a time-dependent scaling (α_t). As $\lim_{t\to 0} \alpha_t = 1$, the two solvers will lead to the same final points when using the same time discretization. Note that the DPM-Solver-1 is also equivalent to DDIM (*c.f.* Section 4.1 in [26]), as also used in this paper.

With that, we can further verify that Heun's method used in this paper corresponds to the DPM-Solver-2 when setting $r_1 = 1$.

A.3 Chapter 6

A.3.1 Proof for Theorem 5

In this section, we provide the proof of Theorem 5. First, we restate the Feynman-Kac theorem. Let $u: [0,T] \times \mathbb{R}^d$ such that for any $t \in [0,T]$ and $x \in \mathbb{R}^d$ we have

$$\partial_t u(t,x) + \langle b(t,x), \nabla u(t,x) \rangle + (1/2)\langle \Sigma(t,x), \nabla^2 u(t,x) \rangle - V(t,x)u(t,x) + \mathbf{f}(t,x) = 0, \quad \text{(A.19)}$$

with $u(T,x) = \Phi(T,x)$. Then, under integrability and regularity assumptions, see [244] for instance, we have

$$u(0,x) = \mathbb{E}[\int_0^T \exp[-\int_0^r V(\tau, \mathbf{X}_\tau)\mathrm{d}\tau]\mathbf{f}(r, \mathbf{X}_r)\mathrm{d}r + \exp[-\int_0^T V(\tau, \mathbf{X}_\tau)\mathrm{d}\tau]\Phi(T, \mathbf{X}_T) \mid \mathbf{X}_0 = x], \quad \text{(A.20)}$$

with $u(T,x) = \Phi(T,x)$ and $\mathrm{d}\mathbf{X}_t = b(t, \mathbf{X}_t)\mathrm{d}t + \Sigma(t, \mathbf{X}_t)\mathrm{d}\mathbf{B}_t$. In the rest of this section, we derive the specific case of Theorem 5.

We recall that the generative model with particle guidance is given by $(\hat{p}_t)_{t\in[0,T]}$ and is associated with the generative model

$$\mathrm{d}\hat{\mathbf{Y}}_t = \{-\mathbf{f}(\hat{\mathbf{Y}}_t, T-t) + g(T-t)^2(\mathbf{s}_\theta(\hat{\mathbf{Y}}_t, T-t) + \nabla \log \Phi_{T-t}(\hat{\mathbf{Y}}_t))\}\mathrm{d}t + g(T-t)\mathrm{d}\mathbf{w}. \quad \text{(A.21)}$$

We also recall that the generative model without particle guidance is given by $(q_t)_{t\in[0,T]}$ and is associated with the generative model

$$\mathrm{d}\mathbf{Y}_t = \{-\mathbf{f}(\mathbf{Y}_t, T-t) + g(T-t)^2\mathbf{s}_\theta(\mathbf{Y}_t, T-t)\}\mathrm{d}t + g(T-t)\mathrm{d}\mathbf{w}. \quad \text{(A.22)}$$

Using the Fokker-Planck equation associated with Equation A.22 we have for any $x \in (\mathbb{R}^d)^N$

$$\partial_t q_t(x) + \mathrm{div}(\{-\mathbf{f}(T-t,\cdot) + g(T-t)^2\mathbf{s}_\theta(T-t,\cdot)\}q_t)(x) - (g(T-t)^2/2)\Delta q_t(x) = 0. \quad \text{(A.23)}$$

This can also be rewritten as

$$\partial_t q_t(x) + \langle -\mathbf{f}(T-t,x) + g(T-t)^2 \mathbf{s}_\theta(x, T-t), \nabla q_t(x)\rangle - (g(T-t)^2/2)\Delta q_t(x) \tag{A.24}$$

$$+ \operatorname{div}(\{-\mathbf{f}(\cdot, T-t) + g(T-t)^2 \mathbf{s}_\theta(T-t, \cdot)\})(x) q_t(x) = 0 \tag{A.25}$$

Denoting $u_t = q_{T-t}$ we have

$$\partial_t u_t(x) + \langle \mathbf{f}(x,t) - g(t)^2 \mathbf{s}_\theta(x,t), \nabla u_t(x)\rangle + (g(t)^2/2)\Delta u_t(x) \tag{A.26}$$

$$- \operatorname{div}(\{-\mathbf{f}(t,\cdot) + g(t)^2 \mathbf{s}_\theta(\cdot, t)\})(x) u_t(x) = 0. \tag{A.27}$$

Note that since $u_t = q_{T-t}$, we have that $u_t = p_t$ with the conventions from 6.2. Now combining this result with Equation A.19 and Equation A.20 with $V(t,x) = \operatorname{div}(\{-\mathbf{f}(\cdot,t) + g(t)^2 \mathbf{s}_\theta(t,\cdot)\})(x)$ and $f = 0$ we have that

$$u_0(x) = \mathbb{E}[Z], \tag{A.28}$$

with

$$Z = \exp[-\textstyle\int_0^T V(\tau, \mathbf{X}_\tau) \mathrm{d}\tau] p_0(\mathbf{X}_T), \tag{A.29}$$

and

$$\mathrm{d}\mathbf{X}_t = \{\mathbf{f}(t, \mathbf{X}_t) - g(t)^2 \mathbf{s}_\theta(\mathbf{X}_t, t)\}\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}. \tag{A.30}$$

with $\mathbf{X}_0 = x$. We now consider a similar analysis in the case of the generative with particle guidance. Using the Fokker-Planck equation associated with Equation A.21 we have for any $x \in (\mathbb{R}^d)^N$

$$\partial_t \tilde{q}_t(x) + \operatorname{div}(\{-\mathbf{f}(\cdot, T-t) + g(T-t)^2(\mathbf{s}_\theta(\cdot, T-t) + \nabla \log \Phi_{T-t})\}\tilde{q}_t)(x) - (g(T-t)^2/2)\Delta \tilde{q}_t(x) = 0. \tag{A.31}$$

This can also be rewritten as

$$\partial_t \tilde{q}_t(x) + \langle -\mathbf{f}(x, T-t) + g(T-t)^2 \mathbf{s}_\theta(x, T-t), \nabla \tilde{q}_t(x) \rangle - (g(T-t)^2/2)\Delta \tilde{q}_t(x) \tag{A.32}$$

$$+ \operatorname{div}(\{-\mathbf{f}(\cdot, T-t) + g(T-t)^2 \mathbf{s}_\theta(\cdot, T-t)\})(x)\tilde{q}_t(x) \tag{A.33}$$

$$+ g(T-t)^2(\langle \log \Phi_{T-t}(x), \nabla \log \tilde{q}_t(x) \rangle + \Delta \log \Phi_{T-t}(x))\tilde{q}_t(x) = 0. \tag{A.34}$$

Denoting $\hat{u}_t = \tilde{q}_{T-t}$ we have

$$\partial_t \hat{u}_t(x) + \langle \mathbf{f}(t, x) - g(t)^2 \mathbf{s}_\theta(x, t), \nabla \hat{u}_t(x) \rangle + (g(t)^2/2)\Delta \hat{u}_t(x) \tag{A.35}$$

$$- \operatorname{div}(\{-\mathbf{f}(\cdot, t) + g(t)^2 \mathbf{s}_\theta(t, \cdot)\})(x)\hat{u}_t(x) \tag{A.36}$$

$$- g(t)^2(\langle \nabla \log \Phi_t(x), \nabla \log \tilde{q}_{T-t}(x) \rangle + \Delta \log \Phi_t(x))\hat{u}_t(x) = 0. \tag{A.37}$$

Following the convention of 6.2, we have that $\hat{u}_t = \hat{p}_0$. Now combining this result with Equation A.19 and Equation A.20 with $\hat{V}(t, x) = \operatorname{div}(\{-\mathbf{f}(\cdot, t) + g(t)^2 \mathbf{s}_\theta(\cdot, t)\})(x)$
$+g(t)^2(\langle \nabla \log \Phi_t(x), \nabla \log \tilde{p}_{T-t}(x) \rangle + \Delta \log \Phi_t(x))$ and $f = 0$ we have that

$$\hat{u}_0(x) = \mathbb{E}[\hat{Z}], \tag{A.38}$$

with

$$\hat{Z} = \exp[-\textstyle\int_0^T \hat{V}(\tau, \mathbf{X}_\tau) \mathrm{d}\tau] p_0(\mathbf{X}_T), \tag{A.39}$$

and

$$\mathrm{d}\mathbf{X}_t = \{\mathbf{f}(\mathbf{X}_t, t) - g(t)^2 \mathbf{s}_\theta(\mathbf{X}_t, t)\}\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}. \tag{A.40}$$

again with $\mathbf{X}_0 = x$. We conclude the proof upon noting that

$$\hat{Z} = Z \exp[-\textstyle\int_0^T g(t)^2(\langle \nabla \log \Phi_t(\mathbf{X}_t), \nabla \log \hat{u}_t(\mathbf{X}_t) \rangle + \Delta \log \Phi_t(\mathbf{X}_t))dt]. \tag{A.41}$$

Interpretation An interpretation of this reweighting term can be given through the lens of SVGD. We introduce the *Stein operator* as in [174] given by for any $\Psi : (\mathbb{R}^d)^N \to (\mathbb{R}^d)^N$ by

$$\mathcal{A}_{\hat{p}_t}(\Psi_t) = \nabla \log \hat{p}_t \Psi_t^\top + \nabla \Psi_t. \tag{A.42}$$

Using $\Psi_t = \nabla \log \Phi_t$, we get that

$$\mathrm{Tr}(\mathcal{A}_{\hat{p}_t}(\Psi_t)) = \langle \nabla \log \Phi_t, \nabla \log \hat{p}_t \rangle + \Delta \log \Phi_t. \tag{A.43}$$

The squared expectation of this quantity w.r.t. a distribution q on $(\mathbb{R}^d)^N$ is the *Kernel Stein Discrepancy* (KSD) between q and $\hat{p}_t$ given the kernel $\log \Phi_t$.

A.3.2 Sampling a Predefined Joint Distribution

For ease of derivation via the Doob h-transform, we temporarily reverse the time from t to $T - t$. Here, p_T is treated as the data distribution, and Φ_T is regarded as the potential, as specified by users. We now consider another model. Namely, we are looking for a generative model $\hat{p}_t$ with $t \in [0, T]$ such that for any $t \in [0, T]$ we have $\hat{p}_t = p_t \Phi_t$ with Φ_T given by the user. In layman's terms, this means that we are considering a *factorized* model for all times t with the additional requirement that at the final time T, the model is given by $p_T = \hat{p}_T \Phi_T$ with Φ_T known. This is to be compared with Theorem 5. Indeed in Theorem 5 while the update on the generative dynamics is explicit (particle guidance term), the update on the density is not. In what follows, we are going to see, using tools from Doob h-transform theory, that we can obtain an expression for the update of the drift in the generative process when considering models of the form $\hat{p}_t = p_t \Phi_t$.

More precisely, we consider the following model. Let $\hat{p}_T = p_T$ and for any $s, t \in [0, T]$ with $s < t$ and $\mathbf{x}_s^{1:n} = \{\mathbf{x}_t^i\}_{i=1}^n \in (\mathbb{R}^d)^n$ and $\mathbf{x}_t^{1:n} = \{\mathbf{x}_t^i\}_{i=1}^n \in (\mathbb{R}^d)^n$ we define

$$\hat{p}_{t|s}(\mathbf{x}_t^{1:n} | \mathbf{x}_s^{1:n}) = p_{t|s}(\mathbf{x}_t^{1:n} | \mathbf{x}_s^{1:n}) \Phi_t(\mathbf{x}_t^{1:n}) / \Phi_s(\mathbf{x}_s^{1:n}), \tag{A.44}$$

with Φ_t which satisfies for any $\mathbf{x}_t^{1:n} \in (\mathbb{R}^d)^n$

$$\partial_t \Phi_t(\mathbf{x}_t^{1:n}) + \langle -f_{T-t}(\mathbf{x}_t^{1:n}) + g(T-t)^2 \nabla \log p_t(\mathbf{x}_t^{1:n}), \nabla \Phi_t(\mathbf{x}_t^{1:n}) \rangle + (g(T-t)^2/2) \Delta \Phi_t(\mathbf{x}_t^{1:n}) = 0, \tag{A.45}$$

with Φ_T given. Note that Equation A.45 expresses that Φ_t satisfies the backward Kolmogorov equation. Under mild assumptions, using Doob h-theory, we get that there exists $(\hat{\mathbf{X}}_t)_{t \in [0,T]}$

such that for any $t \in [0, T]$ we have $\hat{\mathbf{Y}}_t \sim \hat{p}_t$ and for any $t \in [0, T]$

$$\mathrm{d}\hat{\mathbf{Y}}_t = \{-f_{T-t}(\hat{\mathbf{Y}}_t) + g(T-t)^2[\nabla \log p_t(\hat{\mathbf{Y}}_t) + \nabla \log \Phi_t(\hat{\mathbf{Y}}_t)]\}\mathrm{d}t + g(T-t)\mathrm{d}\mathbf{w}. \tag{A.46}$$

The main difficulty is to compute Φ_t for any $t \in [0, T]$. Under mild assumptions, solutions to the backward Kolmogorov Equation A.45 are for any $t \in [0, T]$ by

$$\Phi_t(\mathbf{x}_t^{1:n}) = \mathbb{E}[\Phi_T(\mathbf{Y}_T)|\mathbf{Y}_t = \mathbf{x}_t^{1:n}] = \int \Phi_T(\mathbf{Y}_T = \mathbf{x}_T^{1:n}) p_{T|t}(\mathbf{x}_T^{1:n}|\mathbf{x}_t^{1:n}) d\mathbf{x}_T^{1:n}, \tag{A.47}$$

where we have

$$\mathrm{d}\mathbf{Y}_t = \{-f_{T-t}(\mathbf{Y}_t) + g(T-t)^2 \nabla \log p_t(\mathbf{Y}_t)\}\mathrm{d}t + g(T-t)\mathrm{d}\mathbf{w}. \tag{A.48}$$

This means that $(\mathbf{Y}_t)_{t\in[0,T]}$ is given by the original generative model, with time-dependent marginals p_t. The expression Equation A.47, suggests to parameterize Φ_t by Φ_t^θ and to consider the loss function

$$\ell_t(\theta) = \mathbb{E}_{\mathbf{Y}_T}\mathbb{E}_{\mathbf{Y}_t \sim p_{t|T}(\cdot|\mathbf{Y}_T)}[\|\Phi_T(\mathbf{Y}_T) - \Phi_t^\theta(\mathbf{Y}_t)\|^2]. \tag{A.49}$$

Then, we can define a global loss function $\mathcal{L}(\theta) = \int_0^T \lambda(t)\ell_t(\theta)\mathrm{d}t$ where λ_t is some weight. One problem with this original loss function is that it requires sampling and integrating with respect to $\mathbf{Y}_t$ which requires sampling from the generative model.

Recall that we reverse the time from t to $T - t$ at the beginning. Reverse back to the original convention in the main text, Equation A.50 can be expressed as

$$\ell_t(\theta) = \mathbb{E}_{\mathbf{X}_0 \sim p_0}\mathbb{E}_{\mathbf{X}_t \sim p_{t|0}(\cdot|\mathbf{X}_T)}[\|\Phi_0(\mathbf{X}_0) - \Phi_t^\theta(\mathbf{X}_t)\|^2]. \tag{A.50}$$

A.3.3 Preserving Marginal Distribution

For arbitrary time evolving potentials $\Phi_t(\mathbf{x}_1, ..., \mathbf{x}_n)$ sampling using particle guidance does not preserve the marginals $p(\mathbf{x}_i) \neq \int_{\mathbf{x}_1,...,\mathbf{x}_{i-1},\mathbf{x}_{i+1},...,\mathbf{x}_n} \hat{p}(\mathbf{x}_1, ..., \mathbf{x}_n) d\mathbf{x}_1 ... d\mathbf{x}_{i-1} d\mathbf{x}_{i+1} ... d\mathbf{x}_n$. In many domains this is not required and none of the methods discussed in Section 6.3 have this

property for finite number of particles, however, in domains where, for example, one wants to obtain unbiased estimates of some function this property may be useful.

While the technique discussed in Section 6.5 allows us to use any potential $\Phi_0(\mathbf{x}_1, ..., \mathbf{x}_n)$ choosing Φ_0 in such a way that preserves marginals is hard for non-trivial potentials and distributions. Therefore, we propose to learn a non-trivial marginal preserving Φ_0^θ from the data in the following way. Let $\Phi_0^\theta(\mathbf{x}_1, ..., \mathbf{x}_n) = \Phi_0'(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_i \gamma_\theta(\mathbf{x}_i)$ where Φ_0' is some predefined joint potential that, for example, encourages diversity in the joint distribution and γ_θ is a learned scalar function operating on individual points that counterbalances the effect that Φ_0' has on marginals while maintaining its effect on sample diversity.

How do we learn such scaling function γ_θ to preserve marginals? By definition, the individual marginal distribution is (consider $i = 1$ w.l.o.g.):

$$\begin{aligned}\hat{p}_1(\mathbf{x}_1) =& \frac{\int_{\mathbf{x}_2,...,\mathbf{x}_n} \Phi_0'(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_i p(\mathbf{x}_i)\gamma_\theta(\mathbf{x}_i) d\mathbf{x}_2 ... d\mathbf{x}_n}{\int_{\mathbf{x}_1',...,\mathbf{x}_n'} \Phi_0'(\mathbf{x}_1', ..., \mathbf{x}_n') \prod_i p(\mathbf{x}_i')\gamma_\theta(\mathbf{x}_i') d\mathbf{x}_1' ... d\mathbf{x}_n'} = \\ =& \frac{p(\mathbf{x}_1) E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_{i>1} p(\mathbf{x}_i)} \left[\Phi_0'(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_i \gamma_\theta(\mathbf{x}_i)\right]}{E_{\mathbf{x}_1',...,\mathbf{x}_n' \sim \prod_i p(\mathbf{x}_i')} \left[\Phi_0'(\mathbf{x}_1', ..., \mathbf{x}_n') \prod_i \gamma_\theta(\mathbf{x}_i')\right]} = \\ =& p(\mathbf{x}_1) \frac{E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_{i>1} p(\mathbf{x}_i)} \left[\Phi_0^\theta(\mathbf{x}_1, ..., \mathbf{x}_n)\right]}{E_{\mathbf{x}_1',...,\mathbf{x}_n' \sim \prod_i p(\mathbf{x}_i')} \left[\Phi_0^\theta(\mathbf{x}_1', ..., \mathbf{x}_n')\right]}\end{aligned}$$

Therefore $\hat{p}_1 = p$ if and only if the fraction is always equal to 1 (intuitively for the marginal to be maintained on average the potential should have no effect). Assuming that the joint potential Φ_0' is invariant to the permutation to its inputs. Then one can also prove that, $\hat{p}_1 = p$ if and only if $E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_{i>1} p(\mathbf{x}_i)} \left[\Phi_0^\theta(\mathbf{x}_1, ..., \mathbf{x}_n)\right]$ is equal to a positive constant C for any x_1. We can then minimize the following regression loss to learn the scalar function γ_θ, by matching $E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_{i>1} p(\mathbf{x}_i)} \left[\Phi_0^\theta(\mathbf{x}_1, ..., \mathbf{x}_n)\right]$ and C:

$$\min_\theta \quad E_{\mathbf{x}_1}(E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_i p(\mathbf{x}_i)} \left[\Phi_0'(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_{i \neq 1} \gamma_\theta(\mathbf{x}_i)\right] - \frac{C}{\gamma_\theta(\mathbf{x}_1)})^2$$

However, achieving this objective necessitates costly Monte Carlo estimations for the expectation over $n - 1$ independent samples. Moreover, obtaining an unbiased estimator for the full-batch gradient in a mini-batch setup is challenging. To bypass this issue, we implement a greedy update rule that optimizes the value $\gamma_\theta(\mathbf{x}_1)$ on the single sample $\mathbf{x}_1$. The greedy

update can be viewed as continuous extension of the Iterative Proportional Fitting (IPF) for symmetry matrices. The essence of the IPF algorithm lies in determining scaling factors for each row and column of a given matrix, ensuring that the sum of each row and column (the marginals) aligns with a specified target value. IPF is known for its uniqueness and convergency guarantees [245]. Let's denote $\phi_i(\mathbf{x}_1, ..., \mathbf{x}_n) = \texttt{stop\_grad}(\Phi'_0(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_{j \neq i} \gamma_\theta(\mathbf{x}_j))$, the objective for the greedy update rule is as follows:

$$\min_{\gamma_\theta(\mathbf{x}_1)} \quad E_{\mathbf{x}_1}(E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_{i \neq 1} p(\mathbf{x}_i)} \phi_1(\mathbf{x}_1, ..., \mathbf{x}_n) - \frac{C}{\gamma_\theta(\mathbf{x}_1)})^2$$

The gradient w.r.t θ in the objective above is:

$$\theta' = \theta - \beta E_{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_i p(\mathbf{x}_i)} \left[\frac{2C}{\gamma_\theta^2(\mathbf{x}_1)} \left(\phi_1(\mathbf{x}_1, ..., \mathbf{x}_n) - \frac{C}{\gamma_\theta(\mathbf{x}_1)} \right) \right] \nabla_\theta \gamma_\theta(x_1) \tag{A.51}$$

where β is the learning rate. The corresponding update by stochastic gradient is:

$$\theta' = \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \left[\frac{2C}{\gamma_\theta^2(\mathbf{x}_i)} \left(\phi_i(\mathbf{x}_1, ..., \mathbf{x}_n) - \frac{C}{\gamma_\theta(\mathbf{x}_i)} \right) \right] \nabla_\theta \gamma_\theta(x_i) \tag{A.52}$$

$$= \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \left[\frac{2C}{\gamma_\theta^3(\mathbf{x}_i)} \left(\Phi_0^\theta(\mathbf{x}_1, ..., \mathbf{x}_n) - C \right) \right] \nabla_\theta \gamma_\theta(x_i) \tag{A.53}$$

for $\mathbf{x}_1, ..., \mathbf{x}_n \sim \prod_i p(\mathbf{x}_i)$. The stochastic gradient is an unbiased estimator of the gradient in Equation A.51.

Empirical Synthetic Experiments

We demonstrate this training paradigm in a synthetic experiment using a mixture of Gaussian distributions in 2D. In particular, we set p to be a mixture of 7 Gaussians with the same variance with placed as shown in Figure A.1.A. The middle Gaussian has a weight that is four times that of the others.

Figure A.1.B shows the marginal distribution when sampling 10 particles with joint

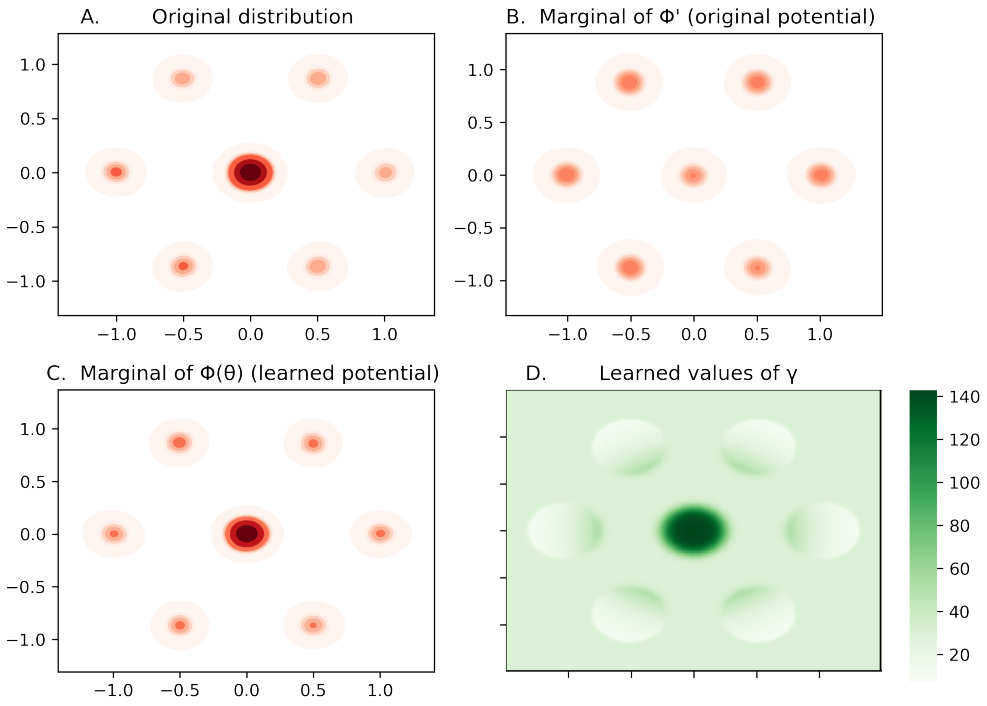


Figure A.1: Synthetic experiment on learning a potential that preserves marginal distributions. The description of each plot can be found in the text.

distribution:

$$\tilde{p}_0(\mathbf{x}_1, ..., \mathbf{x}_n) \propto \Phi'_0(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_{i=1}^{n} p(\mathbf{x}_i)$$

where Φ'_0 is a measure of diversity that in this case we take to be the exponential of the negative of the mean of pairwise Euclidean RBF similarity kernels. Table A.1 highlights how sampling from this modified joint distribution significantly increases the diversity of the samples, especially when it comes to the expected value of $\log \Phi'_0$ itself. However, as clear from Figure A.1.B, the marginal distribution is significantly changed with the middle mode being sampled only 13% of the times instead of 40% and even the shape of the outer modes being altered.

To fix this we learn a function γ_θ, here parameterized simply as the set of values on a grid representing the domain. We follow the training scheme presented in Equation A.52 obtaining the function presented in Figure A.1.D. As evident from the plot, this has the effect of overweighing samples in the central mode, while downsampling samples in the outside

modes, especially when coming from the outer parts. In Figure A.1.C we plot the marginal of the resulting distribution:

$$\hat{p}_0(\mathbf{x}_1, ..., \mathbf{x}_n) \propto \Phi'_0(\mathbf{x}_1, ..., \mathbf{x}_n) \prod_{i=1}^{n} p(\mathbf{x}_i)\gamma_\theta(\mathbf{x}_i).$$

Although the marginals closely match, $\hat{p}_0$ still has more diversity in its sets of samples compared to I.I.D. sampling of p, however, as seen in Table A.1 the level of diversity in $\hat{p}_0$ is lower than that of $\tilde{p}_0$, highlighting the "cost" of imposing the preservation of marginals.

Table A.1: Values of different observables under different joint probability distributions. For every method we take 5000 samples (of 10 particles), samples from $\tilde{p}_0$ and $\hat{p}_0$ were obtained reweighting 50000 samples of the independent I.I.D. distribution.

Observable	I.I.D. p	$\tilde{p}_0$	$\hat{p}_0$
Number of modes recovered at every sample	4.9	5.9	5.3
Expected value of $\log \Phi'_0$	-37.3	-31.8	-36.3

Then to match the marginals we can minimize:

$$\begin{aligned} D_{KL}(p \,||\, \hat{p}_1) &= E_{\mathbf{x}_1 \sim p}\big[-\log \frac{\hat{p}_1(\mathbf{x}_1)}{p(\mathbf{x}_1)} \big] \\ &= E_{\mathbf{x}_1 \sim p}\Big[-\log \frac{E_{\mathbf{x}_2,...,\mathbf{x}_n \sim \prod_{i>1} p(\mathbf{x}_i)}\big[\Phi^\theta_0(\mathbf{x}_1, ..., \mathbf{x}_n)\big]}{E_{\mathbf{x}'_1,...,\mathbf{x}'_n \sim \prod_i p(\mathbf{x}'_i)}\big[\Phi^\theta_0(\mathbf{x}'_1, ..., \mathbf{x}'_n)\big]} \Big] \\ &\leq - E_{\mathbf{x}_1,...,\mathbf{x}_n \sim p_0}\big[\log \Phi^\theta_0(\mathbf{x}_1, ..., \mathbf{x}_n)\big] + \log E_{\mathbf{x}_1,...,\mathbf{x}_n \sim p_0}\big[\Phi^\theta_0(\mathbf{x}_1, ..., \mathbf{x}_n)\big] \end{aligned}$$

Therefore, combining this with the loss presented in Section 6.5 we can train particle guidance with arbitrary Φ'_0 and to preserve marginals with the following objective:

$$\begin{aligned} \theta^* = \arg\min_\theta \mathbb{E}_{\mathbf{x}^0_1,...,\mathbf{x}^0_n \sim p_0}\, \mathbb{E}_{\mathbf{x}^t_i \sim p_{t|0}(\cdot|\mathbf{x}^0_i)}[\|\Phi_0(\mathbf{x}^0_1, ..., \mathbf{x}^0_n) - \Phi^\theta_t(\mathbf{x}^t_1, ..., \mathbf{x}^t_n)\|^2 - \beta \log \Phi^\theta_0(\mathbf{x}^0_1, ..., \mathbf{x}^0_n)] \\ + \beta \log E_{\mathbf{x}^0_1,...,\mathbf{x}^0_n \sim p_0}\big[\Phi^\theta_0(\mathbf{x}^0_1, ..., \mathbf{x}^0_n)\big] \end{aligned}$$

where Φ_0 indicates Φ^θ_0 with stop-gradients and β is an hyperparameter.

This is still not ready for being applied to training because of log outside the expectation

of the last term. Taking the gradient of the term w.r.t. θ we obtain:

$$\begin{aligned}\nabla_\theta \log E_{\mathbf{x}_1^0,\ldots,\mathbf{x}_n^0\sim p_0}\left[\Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)\right] =& \frac{E_{\mathbf{x}_1^0,\ldots,\mathbf{x}_n^0\sim p_0}\left[\Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)\nabla_\theta \log \Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)\right]}{E_{\mathbf{x}_1^0,\ldots,\mathbf{x}_n^0\sim p_0}\left[\Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)\right]} \\ =& \frac{1}{Z} E_{\mathbf{x}_1^0,\ldots,\mathbf{x}_n^0\sim p_0}\left[\Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)\nabla_\theta \log \Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)\right]\end{aligned}$$

where similarly to what has been done previously for example in [246], the denominator Z is estimated with a running average of the value of the kernel over the samples from the I.I.D. distribution. This produces unbiased gradients in the limit of a small learning rate. Therefore, an stochastic gradient descent procedure with batch size equal to 1 would correspond to, taking a sample from the independent distribution $\mathbf{x}_1^0,...,\mathbf{x}_n^0 \sim p_0$, sampling $\mathbf{x}_i^t \sim p_{t|0}(\cdot|\mathbf{x}_i^0)$ and then computing the updates:

$$\theta' = \theta - \nabla_\theta \left\|\Phi_0(\mathbf{x}_1^0,...,\mathbf{x}_n^0) - \Phi_t^\theta(\mathbf{x}_1^t,...,\mathbf{x}_n^t)\right\|^2 + \beta\left(1 - \frac{\Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)}{Z}\right)\nabla_\theta \log \Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0)$$

$$Z' = \epsilon\ \Phi_0^\theta(\mathbf{x}_1^0,...,\mathbf{x}_n^0) + (1-\epsilon)\ Z$$

A.3.4 Invariance of Particle Guidance

Proposition 3. *Let G be the group of rotations or permutations of a set of vectors. Assuming that $p_T(\mathbf{x})$ is a G-invariant distribution, the learned score $s(\mathbf{x},t)$ and $\mathbf{f}(\mathbf{x},t)$ are G-equivariant and the potential $\log \Phi_t(\mathbf{x}_1 \ldots \mathbf{x}_n)$ is G-invariant to a transformation of any of its inputs, then the resulting distribution we sample from will also be G-invariant to a transformation of any of the elements of the set.*

Note that in this section we will derive this specific formulation for the group of rotations or permutations and the Brownian motion in Euclidean space. For a more general statement on Lie groups G and Brownian motions associated with a given metric, one could generalize the result from [247] Proposition F.2.

Proof. For simplicity, we will consider Euler discretization steps going with time from T to 0

(as used in our experiments), however, the proposition applies in the continuous setting too:

$$p_\theta(\mathbf{x}_i^{(t-1)}|\mathbf{x}_{1:n}^{(t)}) = p_{\mathbf{z}}(\mathbf{x}_i^{(t-1)} - \mathbf{x}_i^{(t)} + \mathbf{f}(\mathbf{x}_i^{(t)}, t) - g^2(\mathbf{s}_\theta(\mathbf{x}_i^{(t)}, t) + \nabla_{\mathbf{x}_i^{(t)}} \log \Phi_t(\mathbf{x}_{1:n}^{(t)})))$$

where $\mathbf{z} \sim N(0, g^2 I)$. Without loss of generality since the whole method is invariant to permutations of the particles, consider $\mathbf{x}_n$ to be the particle to which we apply T_g the transformation of an arbitrary group element g.

Since by assumption $\log \Phi_t(\mathbf{x}_{1:n}^{(t)}) = \log \Phi_t(\mathbf{x}_{1:n-1}^{(t)}, T_g(\mathbf{x}_n^{(t)}))$ we have $p_\theta(\mathbf{x}_i^{(t-1)}|\mathbf{x}_{1:n}^{(t)}) = p_\theta(\mathbf{x}_i^{(t-1)}|\mathbf{x}_{1:n-1}^{(t)}, T_g(\mathbf{x}_n^{(t)}))$.

On the other hand, since $\log \Phi_t(\mathbf{x}_{1:n}^{(t)})$ is invariant to G transformations of $\mathbf{x}_n^{(t)}$, its gradient w.r.t. the same variable will be G-equivariant. Therefore:

$$\begin{aligned}
&p_\theta(T_g(\mathbf{x}_n^{(t-1)})|\mathbf{x}_{1:n-1}^{(t)}, T_g(\mathbf{x}_n^{(t)})) = \\
&= p_{\mathbf{z}}(T_g(\mathbf{x}_n^{(t-1)}) - T_g(\mathbf{x}_n^{(t)}) + \mathbf{f}(T_g(\mathbf{x}_n^{(t)}), t) - g^2(\mathbf{s}_\theta(T_g(\mathbf{x}_n^{(t)}), t) + \nabla_{\mathbf{x}_n^{(t)}} \log \Phi_t(\mathbf{x}_{1:n-1}^{(t)}, T_g(\mathbf{x}_n^{(t)})))) \\
&= p_{\mathbf{z}}(T_g(\mathbf{x}_n^{(t-1)}) - T_g(\mathbf{x}_n^{(t)}) + T_g(\mathbf{f}(\mathbf{x}_n^{(t)}, t)) - g^2(T_g(\mathbf{s}_\theta(\mathbf{x}_n^{(t)}, t)) + T_g(\nabla_{\mathbf{x}_n^{(t)}} \log \Phi_t(\mathbf{x}_{1:n}^{(t)})))) \\
&= p_{\mathbf{z}}(T_g(\mathbf{x}_n^{(t-1)} - \mathbf{x}_n^{(t)} + \mathbf{f}(\mathbf{x}_n^{(t)}, t) - g^2(\mathbf{s}_\theta(\mathbf{x}_n^{(t)}, t) + \nabla_{\mathbf{x}_n^{(t)}} \log \Phi_t(\mathbf{x}_{1:n}^{(t)})))) = p_\theta(\mathbf{x}_n^{(t-1)}|\mathbf{x}_{1:n}^{(t)})
\end{aligned}$$

where between lines 2 and 3 we have used the equivariance assumptions and in the latter two the properties of elements of G.

Putting these together, we follow a similar derivation of Proposition 1 from [184]:

$$
\begin{aligned}
&p_\theta(\mathbf{x}^{(0)}_{1:n-1}, T_g(\mathbf{x}^{(0)}_n)) = \\
&= \int p(\mathbf{x}^{(T)}_{1:n-1}, T_g(\mathbf{x}^{(T)}_n)) \prod_{t=1}^{T} p_\theta(\mathbf{x}^{(t-1)}_{1:n-1}, T_g(\mathbf{x}^{(t-1)}_n) | \mathbf{x}^{(t)}_{1:n-1}, T_g(\mathbf{x}^{(t)}_n)) = \\
&= \int \left(\prod_{i<n} p(\mathbf{x}^{(T)}_i) \right) \left(\prod_{t=1}^{T} \prod_{i<n} p_\theta(\mathbf{x}^{(t-1)}_i | \mathbf{x}^{(t)}_{1:n-1}, T_g(\mathbf{x}^{(t)}_n)) \right) \cdot \\
&\qquad \cdot \left(p(T_g(\mathbf{x}^{(T)}_n)) \prod_{t=1}^{T} p_\theta(T_g(\mathbf{x}^{(t-1)}_n) | \mathbf{x}^{(t)}_{1:n-1}, T_g(\mathbf{x}^{(t)}_n)) \right) = \\
&= \int \left(\prod_{i<n} p(\mathbf{x}^{(T)}_i) \right) \left(\prod_{t=1}^{T} \prod_{i<n} p_\theta(\mathbf{x}^{(t-1)}_i | \mathbf{x}^{(t)}_{1:n}) \right) \left(p(\mathbf{x}^{(T)}_n) \prod_{t=1}^{T} p_\theta(\mathbf{x}^{(t-1)}_n | \mathbf{x}^{(t)}_{1:n}) \right) = p_\theta(\mathbf{x}^{(0)}_{1:n})
\end{aligned}
$$

□

A.3.5 Connections with Existing Methods

Particle Guidance as SVGD

In this section, we derive the approximation of Eq. 6.3 starting from the probability flow ODE equivalent of Eq. 6.4 under the assumptions of no drift $\mathbf{f}(x,t) = 0$ and using the following form for $\Phi_t(x_1, ..., x_n) = (\sum_{i,j} k_t(x_i, x_j))^{-\frac{n-1}{2}}$ where k_t is a similarity kernel based on the Euclidean distance (e.g. RBF kernel).

$$
dx_i = \left[\mathbf{f}(x_i, t) - \frac{1}{2} g^2(t) \left(\nabla_{x_i} \log p_t(x_i) + \nabla_{x_i} \log \left(\sum_{ij} k_t(x_i, x_j) \right)^{-\frac{n-1}{2}} \right) \right] dt
$$

$$
= -\frac{1}{2} g^2(t) dt \left(\nabla_{x_i} \log p_t(x_i) - \frac{\frac{1}{2} \nabla_{x_i} \sum_{ij} k_t(x_i, x_j)}{\frac{1}{n-1} \sum_{ij} k_t(x_i, x_j)} \right)
$$

Now we can simplify the numerator using the fact that k_t is symmetric and approximate the denominator assuming that different particles will have similar average distances to other

particles:

$$\approx -\frac{1}{2}g^2(t)dt\left(\nabla_{x_i}\log p_t(x_i) - \frac{\nabla_{x_i}\sum_j k_t(x_i,x_j)}{\sum_j k_t(x_i,x_j)}\right)$$

$$= -\frac{g^2(t)dt}{2\ S(x_i)}\left(\sum_j k_t(x_i,x_j)\nabla_{x_i}\log p_t(x_i) - \nabla_{x_i}k_t(x_i,x_j)\right)$$

where $S(x_i) = \sum_j k_t(x_i,x_j)$. Now we can use the fact that $\nabla_{x_i}k_t(x_i,x_j) = -\nabla_{x_j}k_t(x_i,x_j)$ because the kernel only depends on the Euclidean distance between the two points:

$$= -\frac{n\ g^2(t)dt}{2\ S(x_i)}\left(\frac{1}{n-1}\sum_j k_t(x_i,x_j)\nabla_{x_i}\log p_t(x_i) + \nabla_{x_j}k_t(x_i,x_j)\right)$$

Letting $\epsilon_t(x_i) = \frac{n\ g^2(t)\Delta t}{2\ S(x_i)}$, we obtain Eq. 6.3:

$$x_i^{t-\Delta t} \approx x_i^t + \epsilon_t(x_i)\psi_t(x_i^t) \quad \text{where} \quad \psi(x) = \frac{1}{n-1}\sum_{j=1}^{n}[k_t(x_j^t,x)\nabla_x\log p_t(x) + \nabla_{x_j^t}k_t(x_j^t,x)]$$

Particle Guidance in Poisson Flow Generative Models

In this section, we consider the more general Poisson Flow Generative Models++ [31] framework in which the N-dimensional data distribution is embedded into $N+D$-dimensional space, where D is a positive integer ($D = 1/D \to \infty$ recover PFGM [50]/diffusion models). The data distribution is interpreted as a positive charge distribution. Each particle independently follows the electric field generated by the N-dimensional data distribution $p(x)$ embedded in a $N + D$-dimensional space. One can similarly do particle guidance in the PFGM++ scenarios, treating the group of particles as negative charges, not only attracted by the data distribution but also exerting the mutually repulsive force. Formally, for the augmented data the ODE in PFGM++ (Eq.4 in [31]) is

$$\frac{dx}{dr} = \frac{E(x,r)_x}{E(x,r)_r}$$

where E_x and E_r are the electric fields for different coordinates:

$$E(x,r)_x = \frac{1}{S_{N+D-1}(1)} \int \frac{x-y}{(\|x-y\|^2+r^2)^{\frac{N+D}{2}}} p(y)dy$$

$$E(x,r)_r = \frac{1}{S_{N+D-1}(1)} \int \frac{r}{(\|x-y\|^2+r^2)^{\frac{N+D}{2}}} p(y)dy$$

Note that when $r = \sigma\sqrt{D}, D \to \infty$, the ODE is $\frac{dx}{dr} = \frac{E(x,r)_x}{E(x,r)_r} = -\frac{\sigma}{\sqrt{D}} \nabla_x \log p_\sigma(x)$ and the framework degenerates to diffusion models.

Now if we consider the repulsive forces among a set of (uniformly weighted) particles with the same anchor variables r, $\{(x_i, r)\}_{i=1}^n$, only the electric field in the x coordinate changes (the component in the r coordinate is zero). Denote the new electric field in x component as $\hat{E}_x$:

$$\hat{E}(x_i,r)_x = \underbrace{E(x_i,r)_x}_{\text{attractive force by data}} + \underbrace{\frac{1}{S_{N+D-1}(1)} \frac{1}{n-1} \sum_{j \neq i} \frac{x_j - x_i}{(\|x_j - x_i\|^2)^{\frac{N+D}{2}}}}_{\text{repulsive force between particles}}$$

The corresponding new ODE for the i-th particle is

$$\begin{aligned}
\frac{dx_i}{dr} &= \frac{\hat{E}(x_i,r)_x}{E(x_i,r)_r} \\
&= \frac{E(x_i,r)_x}{E(x_i,r)_r} + \frac{\frac{1}{S_{N+D-1}(1)} \frac{1}{n-1} \sum_{j \neq i} \frac{x_j - x_i}{(\|x_j - x_i\|^2)^{\frac{N+D}{2}}}}{E(x_i,r)_r} \\
&= \underbrace{\frac{E(x_i,r)_x}{E(x_i,r)_r}}_{\text{predicted by pre-trained models}} + \underbrace{\frac{\frac{1}{n-1} \sum_{j \neq i} \frac{x_j - x_i}{(\|x_j - x_i\|^2)^{\frac{N+D}{2}}}}{\int \frac{r}{(\|x-y\|^2+r^2)^{\frac{N+D}{2}}} p(y)dy}}_{\text{particle guidance}} \\
&= \frac{E(x_i,r)_x}{E(x_i,r)_r} + \frac{\frac{1}{n-1} \sum_{j \neq i} \frac{x_j - x_i}{\|x_j - x_i\|^{N+D}}}{\frac{S_{N+D-1}}{r^{D-1} S_{D-1}} p_r(x_i)}
\end{aligned}$$

where p_r is the intermidate distribution, and S_n is the surface area of n-sphere. Clearly, the

direction of the guidance term can be regarded as the sum of the gradient of $N+D$-dimensional Green's function $G(x,y) \propto 1/||x-y||^{N+D-2}$, up to some scaling factors:

$$\nabla_{x_i} G(x_i, x_j) = \frac{x_i - x_j}{\|x_j - x_i\|^{N+D}}$$

A.3.6 Combinatorial Analysis of Synthetic Experiments

Proposition 4. *Let us have a random variable taking a value equiprobably between N distinct bins. The expectation of the proportion of bins discovered (i.e. sampled at least once) after N samples is* $1 - (\frac{N-1}{N})^N$ *which tends to* $1 - 1/e$ *as* N *tends to infinity.*

Proof. Let n_i be the number of samples in the i$^{\text{it}}$ bin. The proportion of discovered bins is equal to:

$$\frac{1}{N}\mathbb{E}\Big[\sum_{i=1}^{N} I_{n_i>0}\Big] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[I_{n_i>0}] = P(n_i > 0) = 1 - P(n_i = 0) = 1 - \left(\frac{N-1}{N}\right)^N$$

In the limit of $N \to \infty$:

$$\lim_{N\to\infty} 1 - \left(\frac{N-1}{N}\right)^N = 1 - y = 1 - \frac{1}{e}$$

since (using L'Hôpital's rule):

$$\log y = \lim_{N\to\infty} N \log\left(\frac{N-1}{N}\right) = \lim_{N\to\infty} \frac{\log(\frac{N-1}{N})}{1/N} = \lim_{N\to\infty} \frac{1/N^2}{-1/N^2} = -1$$

□

Therefore for $N = 10$ we would expect $10 * (1 - 0.9^{10}) \approx 6.51$, which corresponds to what is observed in the empirical results of Section B.4.2.

Proposition 5. ***(Coupon collector's problem)*** *Let us have a random variable taking a value equiprobably between N distinct bins. The expectation of the number of samples required to discover all the bins is* $N\,H_N$*, where* H_N *is the* N^{th} *harmonic number, which is* $\Theta(N \log N)$ *as* N *tends to infinity.*

Proof. Len $L_{i|j}$ be the number of samples it takes to go from j to i bins discovered. We are therefore interested in $\mathbb{E}[L_{N|0}]$.

$$\mathbb{E}[L_{j|j-1}] = \frac{N-(j-1)}{N} * 1 + \frac{j-1}{N}\left[\mathbb{E}[L_{j|j-1}] + 1\right] \implies \mathbb{E}[L_{j|j-1}] = \frac{N}{N-(j-1)}$$

Therefore:

$$\mathbb{E}[L_{N|0}] = \mathbb{E}\left[\sum_{j=1}^{N} L_{j|j-1}\right] = \sum_{j=1}^{N} \mathbb{E}[L_{j|j-1}] = N \sum_{j=1}^{N} \frac{1}{N-(j-1)} = N \sum_{j=1}^{N} \frac{1}{j} = N\, H_N$$

Since H_N is $\Theta(\log N)$, then $\mathbb{E}[L_{N|0}]$ is $\Theta(N \log N)$. □

For $N = 10$, $\mathbb{E}[L_{10|0}] \approx 29.29$.

A.4 Chapter 7

A.4.1 Proof for Theorem 6

Before proceeding to the proof for Theorem 6, we show a technical lemma that guarantees the existence-uniqueness of the solution to the Poisson equation, under some mild conditions.

Lemma 8. *Given $\Omega = \mathbb{R}^N, N \geq 3$, assume that the source function $\rho \in \mathcal{C}^0(\Omega)$, and ρ has a compact support. Then the the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -\rho(\mathbf{x})$ on Ω with zero boundary condition at infinity ($\lim_{\|\mathbf{x}\|_2\to\infty}\varphi(\mathbf{x}) = 0$) has a unique solution $\varphi(\mathbf{x}) \in \mathcal{C}^2(\Omega)$ up to a constant.*

Proof. For the existence of the solution, one can verify that the analytical construction using the extension of Green's function in $N \geq 3$ dimensional space (Lemma 11), *i.e.*, $\varphi(\mathbf{x}) = \int G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})d\mathbf{y}, G(\mathbf{x},\mathbf{y}) = \frac{1}{(N-2)S_{N-1}(1)}\frac{1}{||\mathbf{x}-\mathbf{y}||^{N-2}}$, is one possible solution to the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -\rho(\mathbf{x})$. Since $\rho \in \mathcal{C}^0(\Omega)$ and $\nabla^2\varphi(\mathbf{x}) = -\rho(\mathbf{x})$, we conclude that $\varphi(\mathbf{x}) \in \mathcal{C}^2(\Omega)$.

The proof idea of the uniqueness is similar to the uniqueness theorems in electrostatics. Suppose we have two different solutions $\varphi_1, \varphi_2 \in \mathcal{C}^2$ which satisfy

$$\nabla^2\varphi_1(\mathbf{x}) = -\rho(\mathbf{x}), \nabla^2\varphi_2(\mathbf{x}) = -\rho(\mathbf{x}). \tag{A.54}$$

We define $\tilde{\varphi}(\mathbf{x}) \equiv \varphi_2(\mathbf{x}) - \varphi_1(\mathbf{x})$. Subtracting the above two equations gives

$$\nabla^2\tilde{\varphi}(\mathbf{x}) = 0, \forall\mathbf{x} \in \Omega. \tag{A.55}$$

By the vector differential identity we have

$$\tilde{\varphi}(\mathbf{x})\nabla^2\tilde{\varphi}(\mathbf{x}) = \nabla\cdot(\tilde{\varphi}(\mathbf{x})\nabla\tilde{\varphi}(\mathbf{x})) - \nabla\tilde{\varphi}(\mathbf{x})\cdot\nabla\tilde{\varphi}(\mathbf{x}), \tag{A.56}$$

By the divergence theorem we have

$$\int_\Omega \nabla\cdot(\tilde{\varphi}(\mathbf{x})\nabla\tilde{\varphi}(\mathbf{x}))d^N\mathbf{x} = \oiint_{\partial\Omega}\tilde{\varphi}(\mathbf{x})\nabla\tilde{\varphi}(\mathbf{x})\cdot d^{N-1}\mathbf{S} = 0, \tag{A.57}$$

where $d^{N-1}\mathbf{S}$ denotes an $N-1$ dimensional surface element at infinity, and the second equation holds due to zero boundary condition at infinity. Combining Eq. (A.55)(A.56)(A.57), we have

$$\int_{\Omega} \nabla \cdot (\tilde{\varphi}(\mathbf{x}) \nabla \tilde{\varphi}(\mathbf{x})) d^N\mathbf{x} = \int_{\Omega} ||\nabla \tilde{\varphi}(\mathbf{x})||^2 d^N\mathbf{x} = 0, \tag{A.58}$$

since this is an integral of a positive quantity, we must have $\nabla\tilde{\varphi}(\mathbf{x}) = \mathbf{0}$, or $\tilde{\varphi}(\mathbf{x}) = c$, $\forall \mathbf{x} \in \Omega$. This means φ_1 and φ_2 differ at most by a constant, but a constant does not affect gradients, so $\nabla\varphi_1(\mathbf{x}) = \nabla\varphi_2(\mathbf{x})$. □

In our method section (Section 7.3.1), we augmented the original N-dimensional data with an extra dimension. The new data distribution in the augmented space is $\tilde{p}(\tilde{\mathbf{x}}) = p(\mathbf{x})\delta(z)$, where δ is the Dirac delta function. The support of the data distribution is in the $z = 0$ hyperplane. In the following lemma, we show the existence and uniqueness of the solution to $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$ outside the data support.

Lemma 9. *Assume the support of the data distribution in the augmented space (supp$(\tilde{p}(\tilde{\mathbf{x}}))$) is a compact set on the $z = 0$ hyperplane, $p(\mathbf{x}) \in \mathcal{C}^0$ and $N \geq 3$. The Poisson equation $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$ with zero boundary condition at infinity ($\lim_{\|\mathbf{x}\|_2 \to \infty} \varphi(\tilde{\mathbf{x}}) = 0$) has a unique solution $\varphi(\tilde{\mathbf{x}}) \in \mathcal{C}^2$ for $\tilde{x} \in \mathbb{R}^{N+1} \setminus$ supp$(\tilde{p}(\tilde{\mathbf{x}}))$, up to a constant.*

Proof. Similar to the proof in Lemma 8, one can easily verify that the analytical construction using Green's method, *i.e.*, $\varphi(\tilde{\mathbf{x}}) = \int G(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})\tilde{p}(\tilde{\mathbf{x}})d\tilde{\mathbf{y}}, G(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \frac{1}{(N-1)S_N(1)} \frac{1}{||\tilde{\mathbf{x}}-\tilde{\mathbf{y}}||^{N-1}}$, is one possible solution to the Poisson equation $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$. Since $\tilde{p}(\tilde{\mathbf{x}}) = 0$ for $\tilde{\mathbf{x}} \in \mathbb{R}^{N+1} \setminus$ supp$(\tilde{p}(\tilde{\mathbf{x}}))$ and $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$, we conclude that $\varphi(\tilde{\mathbf{x}}) \in \mathcal{C}^2(\mathbb{R}^{N+1} \setminus \text{supp}(\tilde{p}(\tilde{\mathbf{x}})))$.

For the uniqueness, suppose we have two different solutions $\varphi_1, \varphi_2 \in \mathcal{C}^2(\mathbb{R}^{N+1} \setminus \text{supp}(\tilde{p}(\tilde{\mathbf{x}})))$ which satisfy

$$\nabla^2\varphi_1(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}}), \nabla^2\varphi_2(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}}). \tag{A.59}$$

We define $\tilde{\varphi}(\tilde{\mathbf{x}}) \equiv \varphi_2(\tilde{\mathbf{x}}) - \varphi_1(\tilde{\mathbf{x}})$. Subtracting the above two equations gives

$$\nabla^2\tilde{\varphi}(\tilde{\mathbf{x}}) = 0, \forall \tilde{\mathbf{x}} \in \mathbb{R}^{N+1} \setminus \text{supp}(\tilde{p}(\tilde{\mathbf{x}})). \tag{A.60}$$

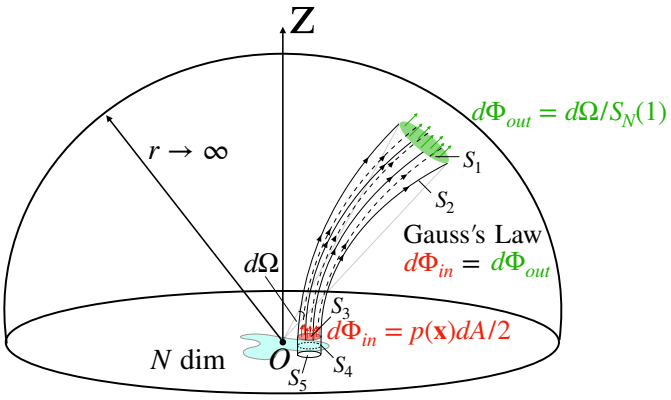


Figure A.2: Proof idea of Theorem 11. By Gauss's Law, the outflow flux $d\Phi_{out}$ equals the inflow flux $d\Phi_{in}$. The factor of two in $p(\mathbf{x})dA/2$ is due to the symmetry of Poisson fields in $z < 0$ and $z > 0$.

By the vector differential identity we have

$$\tilde{\varphi}(\tilde{\mathbf{x}})\nabla^2\tilde{\varphi}(\tilde{\mathbf{x}}) = \nabla \cdot (\tilde{\varphi}(\tilde{\mathbf{x}})\nabla\tilde{\varphi}(\tilde{\mathbf{x}})) - \nabla\tilde{\varphi}(\tilde{\mathbf{x}}) \cdot \nabla\tilde{\varphi}(\tilde{\mathbf{x}}), \tag{A.61}$$

By the divergence theorem we have

$$\int_{\mathbb{R}^{N+1}} \nabla \cdot (\tilde{\varphi}(\tilde{\mathbf{x}})\nabla\tilde{\varphi}(\tilde{\mathbf{x}}))d^{N+1}\tilde{\mathbf{x}} = \oiint_{\partial\mathbb{R}^{N+1}} \tilde{\varphi}(\tilde{\mathbf{x}})\nabla\tilde{\varphi}(\tilde{\mathbf{x}}) \cdot d^N\mathbf{S} = 0, \tag{A.62}$$

where $d^N\mathbf{S}$ denotes an N dimensional surface element at infinity, and the second equation holds due to zero boundary condition at infinity. Combining Eq. (A.60)(A.61)(A.62), we have

$$\begin{aligned}\int_{\mathbb{R}^{N+1}} \nabla \cdot (\tilde{\varphi}(\tilde{\mathbf{x}})\nabla\tilde{\varphi}(\tilde{\mathbf{x}}))d^{N+1}\tilde{\mathbf{x}} &= \int_{\mathbb{R}^{N+1}\setminus\mathrm{supp}(\tilde{p}(\tilde{\mathbf{x}}))} \nabla \cdot (\tilde{\varphi}(\tilde{\mathbf{x}})\nabla\tilde{\varphi}(\tilde{\mathbf{x}}))d^{N+1}\tilde{\mathbf{x}} \\ &= \int_{\mathbb{R}^{N+1}\setminus\mathrm{supp}(\tilde{p}(\tilde{\mathbf{x}}))} ||\nabla\tilde{\varphi}(\tilde{\mathbf{x}})||^2 d^{N+1}\tilde{\mathbf{x}} = 0,\end{aligned}$$

The first equation holds because Lebesgue measure of $\mathrm{supp}(\tilde{p}(\tilde{\mathbf{x}}))$ is zero. Since $||\nabla\tilde{\varphi}(\tilde{\mathbf{x}})||^2$ is an integral of a positive quantity, we must have $\nabla\tilde{\varphi}(\tilde{\mathbf{x}}) = \mathbf{0}$, or $\tilde{\varphi}(\tilde{\mathbf{x}}) = c$, $\forall\tilde{\mathbf{x}} \in \mathbb{R}^{N+1}\setminus\mathrm{supp}(\tilde{p}(\tilde{\mathbf{x}}))$. This means φ_1 and φ_2 differ at most by a constant function, but a constant does not affect gradients, so $\nabla\varphi_1(\tilde{\mathbf{x}}) = \nabla\varphi_2(\tilde{\mathbf{x}})$. □

As illustrated in Figure A.2, there is a bijective mapping between the upper hemisphere

of radius r and the $z = 0$ plane, where each pair of corresponding points is connected by an electric field line. We will now formally prove that, in the $r \to \infty$ limit, this mapping transforms the arbitrary charge distribution in the source plane (that generated the electric field) into a uniform distribution on the hemisphere.

Theorem 11. *Suppose particles are sampled from a uniform distribution on the upper ($z > 0$) half of the sphere of radius r and evolved by the backward ODE $\frac{d\tilde{\mathbf{x}}}{dt} = -\mathbf{E}(\tilde{\mathbf{x}})$ until they reach the $z = 0$ hyperplane, where the Poisson field $\mathbf{E}(\tilde{\mathbf{x}})$ is generated by the source $\tilde{p}(\tilde{\mathbf{x}})$. In the $r \to \infty$ limit, under the conditions in Lemma 9, this process generates a particle distribution $\tilde{p}(\tilde{\mathbf{x}})$, i.e., a distribution $p(\mathbf{x})$ in the $z = 0$ hyperplane.*

Proof. By Lemma 9, we know that with zero boundary at infinity, the Poisson equation $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$ has a unique solution $\varphi(\tilde{\mathbf{x}}) \in \mathcal{C}^2$ for $\tilde{\mathbf{x}} \in \mathbb{R}^{N+1} \setminus \text{supp}(\tilde{p}(\tilde{\mathbf{x}}))$. Hence $\mathbf{E}(\tilde{\mathbf{x}}) = -\nabla\varphi(\tilde{\mathbf{x}}) \in \mathcal{C}^1$, guaranteeing the existence-uniqueness of the solution to the ODE $\frac{d\tilde{\mathbf{x}}}{dt} = -\mathbf{E}(\tilde{\mathbf{x}})$ according to Theorem 2.8.1 in [248].

Consider the tube in Figure A.2 connecting an area on dA in the $z = \epsilon \to 0^+$ hyperplane (S_3) to a solid angle $d\Omega$ on the hemisphere (S_1), with S_2 as its side. The tube is the space swept by dA following electric field $\mathbf{E}$, so by definition the electric field is parallel to the tangent space of the tube sides S_2. The bottom of the tube S_3 is located at $z = \epsilon \to 0^+$, a bit above the $z = 0$ plane, so the tube does not enclose any charges. We note that the divergence of Poisson field is zero in $\mathbb{R}^{N+1} \setminus \text{supp}(\tilde{p}(\tilde{\mathbf{x}}))$:

$$\nabla \cdot \mathbf{E}(\tilde{\mathbf{x}}) = -\nabla^2\varphi(\tilde{\mathbf{x}}) = \tilde{p}(\tilde{\mathbf{x}}) = 0, \forall \tilde{\mathbf{x}} \in \mathbb{R}^{N+1} \setminus \text{supp}(\tilde{p}(\tilde{\mathbf{x}}))$$

Denote the volume and surface of the tube as V and $\mathbf{B}$. According to divergence theorem, $\oiint \mathbf{E}(\tilde{\mathbf{x}}) \cdot d\mathbf{B} = \int_V \nabla \cdot \mathbf{E}(\tilde{\mathbf{x}}) dV = 0$. Hence the net flux leaving the tube is zero:

$$\Phi_{S_1} + \Phi_{S_2} + \Phi_{S_3} = 0, \quad \Phi_{S_i} \equiv \oiint_{S_i} \mathbf{E}(\tilde{\mathbf{x}}) \cdot d\mathbf{B} \quad (i = 1, 2, 3) \tag{A.63}$$

There is no flux through the sides, i.e., $\Phi_{S_2} = 0$, since $\mathbf{E}(\tilde{\mathbf{x}})$ is orthogonal to the surface element $d\mathbf{B}$ on the tube sides by definition. As a result, the flux Φ_{S_3} entering from below must equal the flux Φ_{S_1} leaving the other end. Denote the l_2 norm of the vector $\mathbf{r}$ as r. We

first calculate the influx Φ_{S_3}. To do so, we study a Gaussian pillbox whose top, side and bottom are S_3, S_4 and S_5. S_3 and S_5 are located at $z = \epsilon$ and $z = -\epsilon$ ($\epsilon \to 0^+$). Denote the volume and surface of the pillbox as V' and $\mathbf{B}'$. The pillbox contains charge $p(\mathbf{x})dA$, so according to Gauss's law $\oiint \mathbf{E}(\tilde{\mathbf{x}}) \cdot d\mathbf{B}' = \int_{V'} \nabla \cdot \mathbf{E}(\tilde{\mathbf{x}}) dV' = \int_{V'} \tilde{p}(\tilde{\mathbf{x}}) dV' = p(\mathbf{x})dA$, i.e.,

$$\Phi'_{S_3} + \Phi'_{S_4} + \Phi'_{S_5} = p(\mathbf{x})dA, \quad \Phi'_{S_i} \equiv \oiint_{S_i} \mathbf{E}(\tilde{\mathbf{x}}) \cdot d\mathbf{B}' \quad (i = 3, 4, 5) \tag{A.64}$$

The flux on the sides $\Phi'_{S_4} \propto \epsilon \to 0$, and $\Phi'_{S_3} = \Phi'_{S_5}$ due to mirror symmetry of $z = 0$. So $\Phi'_{S_3} = \Phi'_{S_5} = p(\mathbf{x})dA/2$. Note on the S_3 surface, the outflux of the pillbox is exactly the influx of the tube, so we have:

$$\Phi_{S_3} = -\Phi'_{S_3} = -p(\mathbf{x})dA/2, \tag{A.65}$$

inserting which and $\Phi_{S_2} = 0$ to Eq. (A.63) gives

$$\Phi_{S_1} = -\Phi_{S_3} = p(\mathbf{x})dA/2. \tag{A.66}$$

On the other hand, in the far-field limit $r \to \infty$, since $\text{supp}(p(\mathbf{x}))$ is bounded, the data distribution can be effectively seen as a point charge (see Appendix A.4.3). By Lemma 10, we have $\lim_{r\to\infty} \mathbf{E}(\mathbf{r}) = -\lim_{r\to\infty} \nabla\varphi(\mathbf{r}) = \frac{\mathbf{r}}{S_N(1)r^{N+1}}$. The resulting outflux on the hemisphere is

$$\Phi_{S_1} = E_r r^N d\Omega = d\Omega / S_N(1) \tag{A.67}$$

where $E_r \equiv \mathbf{E}(\mathbf{r}) \cdot \mathbf{r}/r$ is the radial component of $\mathbf{E}$. Comparing Equation A.66 and Equation A.67 yields $d\Omega/dA = p(\mathbf{x})S_N(1)/2 \propto p(\mathbf{x})$. In other words, the mapping from the $z = 0$ hyperplane to the hemisphere dilutes the charge density $p(\mathbf{x})$ up to a constant factor. Thus by change-of-varible, we conclude that the mapping transforms the data distribution into a uniform distribution on the infinite hemisphere. Since the ODE is reversible, the backward ODE transforms the uniform distributoin on the infinite hemisphere to the distribution $\tilde{p}(\tilde{\mathbf{x}})$. □

A.4.2 Proof for the Prior Distribution on $z = z_{\max}$ Hyperplane

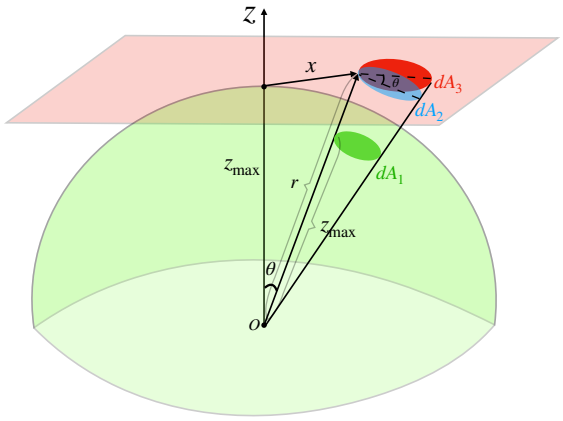


Figure A.3: Diagram of the deviation in Proposition 6

We obtain the prior distribution p_{prior} by projecting the uniform distribution $\mathcal{U}(S_N^+(z_{\max}))$ on the hemisphere $S_N^+(z_{\max})$ to the $z = z_{\max}$ hyperplane. In the following proposition, we show that the projected distribution is $p_{\text{prior}}(\mathbf{x}) = \frac{2z_{\max}}{S_N(1)r^{N+1}}$.

Proposition 6. *The radial projection of $\mathcal{U}(S_N^+(z_{max}))$ on the hemisphere $S_N^+(z_{max})$ to the $z = z_{max}$ hyperplane has the probability density $p_{prior}(\mathbf{x}) = \frac{2z_{max}}{S_N(1)r^{N+1}}$.*

Proof. We calculate the change-of-variable ratio by comparing two associate areas. As illustrated in Figure A.3, an area dA_1 on $S_N^+(z_{\max})$ is projected to an area dA_3 on the hyperplane in the $(\mathbf{x}, z_{\max})$ direction, and we have

$$\mathcal{U}(S_N^+(z_{\max}))dA_1 = p_{\text{prior}}(\mathbf{x})dA_3$$

We aim to calculate the ratio dA_1/dA_3 below. We define the angle between $(\mathbf{0}, z_{\max})$ and $\tilde{\mathbf{x}} = (\mathbf{x}, z_{\max})$ to be θ. We project dA_3 to the hyperplane orthogonal to $\tilde{\mathbf{x}}$ to get $dA_2 = dA_3\cos\theta = dA_3 z_{\max}/r$ where $r \equiv ||\tilde{\mathbf{x}}||_2 = \sqrt{||\mathbf{x}||_2^2 + z_{\max}^2}$. Since dA_1 is parallel to dA_2 and they lie in the same cone from the origin O, we have $dA_2/dA_1 = (r/z_{\max})^N$. Combining all

the results gives

$$\begin{aligned} p_{\text{prior}}(\mathbf{x}) &= \mathcal{U}(S_N^+(z_{\max}))\frac{dA_1}{dA_3} \\ &= \mathcal{U}(S_N^+(z_{\max}))\frac{dA_1}{dA_2}\frac{dA_2}{dA_3} = \frac{2}{S_N(1)z_{\max}^N}(\frac{z_{\max}}{r})^N\frac{z_{\max}}{r} = \frac{2z_{\max}}{S_N(1)r^{N+1}} \end{aligned}$$

□

In order to sample from $p_{\text{prior}}(\mathbf{x})$, we first sample the norm (radius) $R = ||\mathbf{x}||_2$ from the distribution:

$$\begin{aligned} p_{\text{radius}}(R) &\propto R^{N-1}p_{\text{prior}}(\mathbf{x}) \qquad (p_{\text{prior}} \text{ is isotropic}) \\ &\propto R^{N-1}/(||\mathbf{x}||_2^2 + z_{\max}^2)^{\frac{N+1}{2}} \\ &= R^{N-1}/(R^2 + z_{\max}^2)^{\frac{N+1}{2}} \end{aligned} \tag{A.68}$$

and then uniformly sample its angle. Sampling from p_{prior} encompasses three steps. We first sample a real number r_1 with parameters $\alpha = \frac{N}{2}, \beta = \frac{1}{2}$, *i.e.*,

$$R_1 \sim \text{Beta}(\alpha, \beta)$$

Next, we set $R_2 = \frac{R_1}{1-R_1}$ such that R_2 is effectively sampled from the inverse beta distribution a(also known as beta prime distribution) with parameters $\alpha = \frac{N}{2}, \beta = \frac{1}{2}$. Finally, we set $R_3 = \sqrt{z_{\max}^2 R_2}$. To verify the pdf of R_3 is p_{radius}, note that the pdf of inverse beta distribution is

$$p(R_2) \propto R_2^{\frac{N}{2}-1}(1+R_2)^{-\frac{N}{2}-\frac{1}{2}}$$

Next, by change-of-variable, the pdf of $R_3 = \sqrt{z_{\max}^2 R_2}$ is

$$\begin{aligned}
p(R_3) &\propto R_2^{\frac{N}{2}-1}(1+R_2)^{-\frac{N}{2}-\frac{1}{2}} * \frac{2R_3}{z_{\max}^2} \\
&\propto \frac{R_3 R_2^{\frac{N}{2}-1}}{(1+R_2)^{\frac{N+1}{2}}} \\
&= \frac{(R_3/z_{\max})^{N-1}}{(1+(R_3^2/z_{\max}^2))^{\frac{N+1}{2}}} \\
&\propto \frac{R_3^{N-1}}{(1+(R_3^2/z_{\max}^2))^{\frac{N+1}{2}}} \\
&\propto \frac{R_3^{N-1}}{(z_{\max}^2+R_3^2)^{\frac{N+1}{2}}} \propto p_{\text{radius}}(R_3) \qquad \text{(By Equation B.6)}
\end{aligned}$$

Hence we conclude that $p(R_3) = p_{\text{radius}}(R_3)$.

A.4.3 Multipole Expansion

We discuss the behaviors of the potential function in Poisson equation (Equation 7.1) under different scenarios, utilizing the multipole expansion. Suppose we have a unit point charge $q = 1$ located at $\mathbf{x} \in \mathbb{R}^N$. We know that the potential function at another point $\mathbf{y} \in \mathbb{R}^N$ is $\varphi(\mathbf{y}-\mathbf{x}) = 1/||\mathbf{y}-\mathbf{x}||^{N-2}$ (ignoring a constant factor). Now we assume that $\mathbf{x}$ is close to the origin such that we can Taylor expand around $\mathbf{x} = 0$:

$$\varphi(\mathbf{y}-\mathbf{x}) = \varphi(\mathbf{y}) - \sum_{\alpha=1}^{N} \mathbf{x}_\alpha \varphi_\alpha(\mathbf{y}) + \frac{1}{2}\sum_{\alpha=1}^{N}\sum_{\beta=1}^{N} \mathbf{x}_\alpha \mathbf{x}_\beta \varphi_{\alpha\beta}(\mathbf{y}) - ... \tag{A.69}$$

where

$$\begin{aligned}
\varphi_\alpha(\mathbf{y}) &= \left(\frac{\partial \varphi(\mathbf{y}-\mathbf{x})}{\partial \mathbf{x}_\alpha}\right)_{\mathbf{x}=0} = (N-2)\frac{\mathbf{y}_\alpha}{||\mathbf{y}||^N} \\
\varphi_{\alpha\beta}(\mathbf{y}) &= \left(\frac{\partial^2 \varphi(\mathbf{y}-\mathbf{x})}{\partial \mathbf{x}_\alpha \partial \mathbf{x}_\beta}\right)_{\mathbf{x}=0} = (N-2)\frac{N\mathbf{y}_\alpha\mathbf{y}_\beta - ||\mathbf{y}||^2\delta_{\alpha\beta}}{||\mathbf{y}||^{N+2}}
\end{aligned} \tag{A.70}$$

In the case where the source is a distribution $p(\mathbf{x})$, the potential $\varphi(\mathbf{y})$ can again be Taylor expanded:

$$\varphi(\mathbf{y}) = q\varphi(\mathbf{y}) + \sum_{\alpha=1}^{N} q_\alpha \varphi_\alpha(\mathbf{y}) + \sum_{\alpha=1}^{N}\sum_{\beta=1}^{N} q_{\alpha\beta}\varphi_{\alpha\beta}(\mathbf{y}) - ... \tag{A.71}$$

where

$$q = \int p(\mathbf{x})d\mathbf{x}, q_\alpha = \int p(\mathbf{x})\mathbf{x}_\alpha d\mathbf{x}, q_{\alpha\beta} = \int p(\mathbf{x})\mathbf{x}_\alpha\mathbf{x}_\beta d\mathbf{x}, \tag{A.72}$$

which are called monopole, dipole and quadrupole in physics, respectively. The gradient field $\mathbf{E}(y) = \nabla\Phi(\mathbf{y})$ can be expanded in the same such that

$$\mathbf{E}(\mathbf{y}) = \mathbf{E}^{(0)}(\mathbf{y}) + \mathbf{E}^{(1)}(\mathbf{y}) + \mathbf{E}^{(2)}(\mathbf{y}) + ... \tag{A.73}$$

It is easy to check that $||\mathbf{E}^{(i)}(\mathbf{y})||$ decays as $1/||\mathbf{y}||^{N-2+i}$, which means higher-order corrections decay faster than leading terms. So when $||\mathbf{y}|| \to \infty$, only the monopole term $||\mathbf{E}^{(0)}(\mathbf{y})||$ matters, which behaves like a point source.

In a more realistic setup, we only have a large but finite $||\mathbf{y}||$, so the question is: under what condition is the point source approximation valid? We examine $\varphi^{(0)}$, $\varphi^{(1)}$ and $\varphi^{(2)}$ more carefully:

$$\begin{aligned}
\varphi^{(0)} &= \frac{1}{||\mathbf{y}||^{N-2}} \\
\varphi^{(1)} &= \sum_{\alpha=1}^{N}(N-2)\frac{\mathbf{y}_\alpha\mathbf{x}_\alpha}{||\mathbf{y}||^N} = (N-2)\frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{y}||^N} \\
\varphi^{(2)} &= \frac{1}{2}\sum_{\alpha=1}^{N}\sum_{\beta=1}^{N}(N-2)\frac{N\mathbf{y}_\alpha\mathbf{y}_\beta - ||\mathbf{y}||^2\delta_{\alpha\beta}}{||\mathbf{y}||^{N+1}}\mathbf{x}_\alpha\mathbf{x}_\beta = \frac{N-2}{2}\frac{N(\mathbf{x}^T\mathbf{y})^2 - ||\mathbf{x}||^2||\mathbf{y}||^2}{||\mathbf{y}||^{N+2}}
\end{aligned} \tag{A.74}$$

Since $\varphi^{(1)}$ is an odd function of $\mathbf{x}$, integrating $\varphi^{(1)}$ over $\mathbf{x}$ leads to zero (samples are normalized to zero mean). In machine learning applications, N is usually a large number (although in physics N is merely 3). If $\mathbf{y}$ is a random vector of length $||\mathbf{y}||$, then $\mathbf{x}^T\mathbf{y} \sim (\frac{1}{\sqrt{N}} \pm \frac{1}{N})||\mathbf{x}||||\mathbf{y}||$. So Eq. (A.74) can be approximated as

$$\varphi^{(0)} \sim \frac{1}{||\mathbf{y}||^{N-2}}, \varphi^{(2)} \sim \frac{\sqrt{N}}{2}\frac{||\mathbf{x}||^2}{||\mathbf{y}||^N} \tag{A.75}$$

Requiring $\int \varphi^{(0)}p(\mathbf{x})d\mathbf{x} \gg \int \varphi^{(2)}p(\mathbf{x})d\mathbf{x}$ gives $||\mathbf{y}||^2 \gg \sqrt{N}\,\mathbb{E}_{p(x)}\,||\mathbf{x}||^2$. So the condition for

the point source approximation to be valid is:

$$\kappa = \frac{2||\mathbf{y}||^2}{\sqrt{N}\,\mathbb{E}_{p(x)}\,||\mathbf{x}||^2} \gg 1 \tag{A.76}$$

Based on this condition, we can partition space into three zones: (1) the far zone $\kappa \gg 1$, where the point source approximation is valid; (2) the intermediate zone $\kappa \sim O(1)$, where the gradient field has moderate curvature; (3) the near zone $\kappa \ll 1$, where the gradient field has high curvature. In practice, the initial value $||\mathbf{y}||$ is greater than 1000 (hence $\kappa \gg 1$) with high probability on CIFAR-10 and CelebA datasets, incidating that the initial samples lie in the far zone and gradually move toward the near zone where $||\mathbf{y}|| \approx ||\mathbf{x}||$ ($\kappa \ll 1$).

We summarize above observations in the following lemma in the $||\mathbf{y}|| \to \infty$ limit:

Lemma 10. *Assume the data distribution $p(\mathbf{x}) \in \mathcal{C}^0$ has a compact support in $\mathbb{R}^N$, then the solution φ to the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -p(\mathbf{x})$ with zero boundary condition at infinity satisfies $\lim_{\|\mathbf{x}\|_2\to\infty} \nabla\varphi(\mathbf{x}) = -\frac{1}{S_{N-1}(1)}\frac{\mathbf{x}}{\|\mathbf{x}\|_2^N}$.*

Proof. By Lemma 8, the gradient of the solution has the following form:

$$\nabla\varphi(\mathbf{x}) = \int \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) p(\mathbf{y}) d\mathbf{y}, \quad \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) = -\frac{1}{S_{N-1}(1)}\frac{\mathbf{x}-\mathbf{y}}{||\mathbf{x}-\mathbf{y}||^N}.$$

Since $p(\mathbf{x})$ has a bounded support, we assume $\max\{\| \mathbf{x} \|_2 \colon p(\mathbf{x}) \neq 0\} < B$. On the other hand, we have

$$\lim_{\|\mathbf{x}\|_2\to\infty} \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) = \lim_{\|\mathbf{x}\|_2\to\infty} -\frac{1}{S_{N-1}(1)}\frac{\mathbf{x}-\mathbf{y}}{||\mathbf{x}-\mathbf{y}||^N} = \lim_{\|\mathbf{x}\|_2\to\infty} -\frac{1}{S_{N-1}(1)}\frac{\mathbf{x}}{||\mathbf{x}||^N}$$

for $\forall y$ such that $\| y \|_2 < B$. Hence,

$$\begin{aligned}\lim_{\|\mathbf{x}\|_2\to\infty} \nabla\varphi(\mathbf{x}) = \lim_{\|\mathbf{x}\|_2\to\infty} \int \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) p(\mathbf{y}) d\mathbf{y} &= \int \lim_{\|\mathbf{x}\|_2\to\infty} \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) p(\mathbf{y}) d\mathbf{y} \\ &= -\frac{1}{S_{N-1}(1)}\frac{\mathbf{x}}{\| \mathbf{x} \|_2^N}\end{aligned}$$

□

A.4.4 Extension of Green's Function in N-dimensional Space

In this section, we show that the function $G(\mathbf{x}, \mathbf{y})$ defined in Equation 7.2 is the N-dimensional extension of the Green's function, $\varphi(\mathbf{x}) = \int G(\mathbf{x}, \mathbf{y})\rho(\mathbf{y})d\mathbf{y}$ solves the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -\rho(\mathbf{x})$.

Lemma 11. *Assume the dimension $N \geq 3$, and the source term satisfies $\rho \in \mathcal{C}^0(\Omega)$, $\int_{\mathbb{R}^N} \rho^2(\mathbf{x})d\mathbf{x} < +\infty, \lim_{\|\mathbf{x}\|_2\to\infty} \rho(\mathbf{x}) = 0$. The extension of Green's function $G(\mathbf{x}, \mathbf{y}) = \frac{1}{(N-2)S_{N-1}(1)} \frac{1}{||\mathbf{x}-\mathbf{y}||^{N-2}}$ solves the Poisson equation $\nabla^2_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) = -\delta(\mathbf{x} - \mathbf{y})$. In addition, with zero boundary condition at infinity ($\lim_{\|\mathbf{x}\|_2\to\infty} \varphi(\mathbf{x}) = 0$), $\varphi(\mathbf{x}) = \int G(\mathbf{x}, \mathbf{y})\rho(\mathbf{y})d\mathbf{y}$ solves the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -\rho(\mathbf{x})$.*

Proof. It is convenient to denote $\mathbf{r} = \mathbf{x} - \mathbf{y}$, $r = ||\mathbf{r}||$ and notice $\partial r/\partial \mathbf{x} = \mathbf{r}/r$. Firstly, we calculate $\nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})$:

$$\begin{aligned} \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) &= \frac{1}{(N-2)S_{N-1}(1)} \nabla_{\mathbf{x}} (\frac{1}{r^{N-2}}) \\ &= \frac{1}{(N-2)S_{N-1}(1)} \frac{\partial}{\partial r} (\frac{1}{r^{N-2}}) \nabla_{\mathbf{x}} r \\ &= -\frac{1}{S_{N-1}(1)} \frac{\mathbf{r}}{r^N} \end{aligned} \tag{A.77}$$

Then we calculate $\nabla^2_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})$:

$$\begin{aligned} \nabla^2_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) &\equiv \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) \\ &= -\frac{1}{S_{N-1}(1)} \nabla_{\mathbf{x}} \cdot \frac{\mathbf{r}}{r^N} \\ &= -\frac{1}{S_{N-1}(1)} (\nabla_{\mathbf{x}} (\frac{1}{r^N}) \cdot \mathbf{r} + \frac{1}{r^N} \nabla_{\mathbf{r}} \cdot \mathbf{r}) \\ &= -\frac{1}{S_{N-1}(1)} (-\frac{N}{r^N} + \frac{N}{r^N}) \\ &= -\frac{0}{S_{N-1}(1) r^N} \end{aligned} \tag{A.78}$$

which is 0 for $r > 0$, but undermined for $r = 0$. So we are left with proving

$$\int_{S_\epsilon(\mathbf{y})} \nabla^2_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) d^N\mathbf{x} = -1, \tag{A.79}$$

where $S_\epsilon(\mathbf{y})$ denotes a ball centered at $\mathbf{y}$ with a radius $\epsilon \to 0^+$. With the divergence theorem, we have

$$\int_{S_\epsilon(\mathbf{y})} \nabla^2_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) d^N\mathbf{x} = \oiint_{\partial S_\epsilon(\mathbf{y})} \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) \cdot d^{N-1}\mathbf{B} \tag{A.80}$$

where the surface integral can be computed

$$\oiint_{\partial S_\epsilon(\mathbf{y})} \nabla_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) \cdot d^{N-1}\mathbf{B} = \oiint_{\partial S_\epsilon(\mathbf{y})} (-\frac{1}{S_{N-1}(1)}\frac{\mathbf{r}}{r^N}) \cdot d^{N-1}\mathbf{B} = -\frac{1}{S_{N-1}(1)}\frac{S_{N-1}(\epsilon)}{\epsilon^{N-1}} = -1 \tag{A.81}$$

in which we used $\oiint_{\partial S_\epsilon(\mathbf{y})} \mathbf{r} \cdot d^{N-1}\mathbf{B} = \epsilon S_{N-1}(\epsilon)$. Together, we conclude that

$$\nabla^2_{\mathbf{x}} G(\mathbf{x},\mathbf{y}) = -\delta(\mathbf{x}-\mathbf{y}) \tag{A.82}$$

Next we show that $\varphi(\mathbf{x}) = \int G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})d\mathbf{y}$ solves $\nabla^2\varphi(\mathbf{x}) = -\rho(\mathbf{x})$. Taking the Laplacian operator of both sides gives:

$$\begin{aligned}
\nabla^2_{\mathbf{x}}\varphi(\mathbf{x}) &= \nabla^2_{\mathbf{x}} \int G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})d\mathbf{y} \\
&= \int \nabla^2_{\mathbf{x}} G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})d\mathbf{y} \\
&= \int -\delta(\mathbf{x}-\mathbf{y})\rho(\mathbf{y})d\mathbf{y} \quad \text{(By Equation A.82)} \\
&= -\rho(\mathbf{x})
\end{aligned}$$

In addition, we show that $\varphi(\mathbf{x})$ is zero at infinity. Since $\rho(\mathbf{x}) \in \mathcal{C}^0$ and has compact support, we know that $\rho(\mathbf{x})$ is bounded, and let $|\rho(\mathbf{x})| < B$.

$$\begin{aligned}
\lim_{\|\mathbf{x}\|_2\to\infty} \varphi(\mathbf{x}) &= \lim_{\|\mathbf{x}\|_2\to\infty} \int G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})d\mathbf{y} \\
&\leq B \lim_{\|\mathbf{x}\|_2\to\infty} \int_{\text{supp}(\rho)} \frac{1}{(N-2)S_{N-1}(1)} \frac{1}{||\mathbf{x}-\mathbf{y}||^{N-2}} d\mathbf{y} \\
&= 0
\end{aligned}$$

The last equality holds since $\text{supp}(\rho)$ is a compact set. □

A.4.5 Physical Interpretation of the ODEs in PFGM

In Section 7.2, in order to move the particles along the electric lines, we set the time derivative of x to the Poisson field $\mathbf{E}(x)$:

$$[q = 1, \text{forward ODE}] \quad \frac{d\mathbf{x}}{dt} = \mathbf{E}(\mathbf{x}), \quad [q = -1, \text{backward ODE}] \quad \frac{d\mathbf{x}}{dt} = -\mathbf{E}(\mathbf{x}) \tag{A.83}$$

We give the interpretation of the ODEs from a physical perspective. Newton's law implies that the external force is proportional to the acceleration of the particle. In the overdamped limit, e.g., when the particle is moving in honey, the external force is instead proportional to the velocity of the particle, making the equation of motion a first-order ODE. Denoting the viscosity of the fluid as γ, the dynamics of the particle under the influence of the electric field of the source $\rho(\mathbf{x})$ is

$$m\frac{d^2\mathbf{x}}{dt^2} = -\gamma\frac{d\mathbf{x}}{dt} + q\mathbf{E}(\mathbf{x}),$$

which has an overdamped limit $\frac{d\mathbf{x}}{dt} = q\mathbf{E}(\mathbf{x})$ when we set $t \to \gamma t$ and $\gamma \to \infty$. In this case, a particle with mass $m = 1$ and charge $q = 1$ would follow the electric field with velocity equal to $\mathbf{E}$, justifying Eq. (A.83).

A.5 Chapter 8

A.5.1 Proof for Theorem 7

Theorem 7. *Assume the data distribution $p \in \mathcal{C}^1$ and p has compact support. As $r_{max} \to \infty$, for $D \in \mathbb{R}^+$, the ODE $\mathrm{d}\mathbf{x}/\mathrm{d}r = \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r$ defines a bijection between $\lim_{r_{max}\to\infty} p_{r_{max}}(\mathbf{x}) \propto \lim_{r_{max}\to\infty} r_{max}^D/(\|\mathbf{x}\|_2^2 + r_{max}^2)^{\frac{N+D}{2}}$ when $r = r_{max}$ and the data distribution p when $r = 0$.*

Proof. Let $q_r(\mathbf{x}) = \frac{S_{D-1}}{S_{N+D-1}} \int r^D/\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} p(\mathbf{y}) d\mathbf{y}$, where S_n is the surface area of the n-sphere. We will show that $q_r \propto \int r^D/\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} p(\mathbf{y}) d\mathbf{y}$ is equal to the r-dependent marginal distribution p_r by verifying (1) the starting distribution is correct when r=0; (2) the continuity equation holds, *i.e.*, $\partial_r q_r + \nabla \cdot (q_r \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r) = 0$. The starting distribution is $\lim_{r\to 0} q_r(\mathbf{x}) \propto \lim_{r\to 0} \int r^D/\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} p(\mathbf{y}) d\mathbf{y} \propto p(\mathbf{x})$, which confirms that q_r=p. The

continuity equation can be expressed as:

$$
\begin{aligned}
&\partial_r q_r + \nabla \cdot (q_r \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}} / E(\tilde{\mathbf{x}})_r) \\
&= \partial_r \left(\int \frac{r^D}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y} \right) + \nabla \cdot \left(\int \frac{r^D}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y} \frac{\int \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{y}}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y}}{\int \frac{r}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y}} \right) \\
&= \int \left(\frac{D r^{D-1}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} - \frac{(N+D) r}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2}} \right) p(\mathbf{y}) d\mathbf{y} + \nabla \cdot \left(r^{D-1} \int \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{y}}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y} \right) \\
&= \int \left(\frac{D r^{D-1}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} - \frac{(N+D) r}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2}} \right) p(\mathbf{y}) d\mathbf{y} + \nabla \cdot \left(r^{D-1} \int \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{y}}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y} \right) \\
&= \int \left(\frac{D r^{D-1}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} - \frac{(N+D) r}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2}} \right) p(\mathbf{y}) d\mathbf{y} \\
&\quad + r^{D-1} \sum_{i=1}^{N} \int \frac{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} - \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2} (\mathbf{x}_i - \mathbf{y}_i)^2 (N+D)}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{2(N+D)}} p(\mathbf{y}) d\mathbf{y} \\
&= \int \left(\frac{D r^{D-1}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D}} - \frac{(N+D) r^{D+1}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2}} \right) p(\mathbf{y}) d\mathbf{y} \\
&\quad + r^{D-1} \int \frac{N \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} - \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2} \|\mathbf{x} - \mathbf{y}\|^2 (N+D)}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{2(N+D)}} p(\mathbf{y}) d\mathbf{y} \\
&= r^{D-1} \int \frac{(N+D)(\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} - \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2} \|\mathbf{x} - \mathbf{y}\|^2) - (N+D) r^2 \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{2(N+D)}} \\
&\quad p(\mathbf{y}) d\mathbf{y} \\
&= r^{D-1} \int \frac{(N+D) r^2 \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2} - (N+D) r^2 \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D-2}}{\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{2(N+D)}} p(\mathbf{y}) d\mathbf{y} \\
&= 0
\end{aligned}
$$

It means that q_r satisfies the continuity equation for any $r \in \mathbb{R}_{\geq 0}$. Together, we conclude

that $q_r = p_r$. Lastly, note that the terminal distribution is

$$\begin{aligned}
\lim_{r_{\max}\to\infty} p_{r_{\max}}(\mathbf{x}) &\propto \lim_{r_{\max}\to\infty} \int \frac{r_{\max}^D}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y} \\
&= \lim_{r_{\max}\to\infty} \int \frac{r_{\max}^D}{(\|\mathbf{x}-\mathbf{y}\|^2 + r_{\max}^2)^{\frac{N+D}{2}}} p(\mathbf{y}) d\mathbf{y} \\
&= \lim_{r_{\max}\to\infty} \frac{r_{\max}^D}{(\|\mathbf{x}\|^2 + r_{\max}^2)^{\frac{N+D}{2}}} \\
&\quad + \lim_{r_{\max}\to\infty} \int \left(\frac{r_{\max}^D}{(\|\mathbf{x}-\mathbf{y}\|^2 + r_{\max}^2)^{\frac{N+D}{2}}} - \frac{r_{\max}^D}{(\|\mathbf{x}\|^2 + r_{\max}^2)^{\frac{N+D}{2}}} \right) p(\mathbf{y}) d\mathbf{y} \\
&= \lim_{r_{\max}\to\infty} \frac{r_{\max}^D}{(\|\mathbf{x}\|^2 + r_{\max}^2)^{\frac{N+D}{2}}} \qquad (p \text{ has a compact support})
\end{aligned}$$

□

A.5.2 Proof for Proposition 1

Proposition 1. *With perturbation kernel* $p_r(\mathbf{x}|\mathbf{y}) \propto 1/(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}$*, for* $\forall \mathbf{x} \in \mathbb{R}^N, r > 0$*, the minimizer* $f_\theta^*(\tilde{\mathbf{x}})$ *in the PFGM++ objective (Equation 8.3) matches the direction of electric field* $\mathbf{E}(\tilde{\mathbf{x}})$ *in Equation 8.1. Specifically,* $f_\theta^*(\tilde{\mathbf{x}}) \propto (S_{N+D-1}(1)/p_r(\mathbf{x}))\mathbf{E}(\tilde{\mathbf{x}})$.

Proof. The minimizer at $\tilde{\mathbf{x}}$ in Equation 8.3 is

$$\begin{aligned}
f_\theta^*(\tilde{\mathbf{x}}) &= \int p_r(\mathbf{y}|\mathbf{x})(\tilde{\mathbf{x}}-\tilde{\mathbf{y}}) d\tilde{\mathbf{y}} \\
&= \frac{\int p_r(\mathbf{x}|\mathbf{y})(\tilde{\mathbf{x}}-\tilde{\mathbf{y}}) p(\mathbf{y}) d\mathbf{y}}{p_r(\mathbf{x})}
\end{aligned} \tag{A.84}$$

The choice of perturbation kernel is

$$p_r(\mathbf{x}|\mathbf{y}) \propto \frac{1}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}\|^{N+D}} = \frac{1}{(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}}$$

By substituting the perturbation kernel in Equation A.84, we have:

$$
\begin{aligned}
f_\theta^*(\tilde{\mathbf{x}}) &\propto \frac{\int \frac{\tilde{\mathbf{x}}-\tilde{\mathbf{y}}}{(\|\mathbf{x}-\mathbf{y}\|_2^2+r^2)^{\frac{N+D}{2}}} p(\mathbf{y}) d\mathbf{y}}{p_r(\mathbf{x})} \\
&= \frac{\int \frac{\tilde{\mathbf{x}}-\tilde{\mathbf{y}}}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}\|_2{}^{N+D}} p(\mathbf{y}) d\mathbf{y}}{p_r(\mathbf{x})} \\
&= (S_{N+D-1}(1)/p_r(\mathbf{x}))\mathbf{E}(\tilde{\mathbf{x}})
\end{aligned}
$$

□

A.5.3 Proof for Theorem 8

Theorem 8. *Assume the data distribution $p \in \mathcal{C}^1$. Consider taking the limit $D \to \infty$ while holding $\sigma = r/\sqrt{D}$ fixed. Then, for all $\mathbf{x}$,*

$$
\lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \left\| -\frac{\sqrt{D}}{E(\tilde{\mathbf{x}})_r}\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}} - \sigma\nabla_{\mathbf{x}} \log p_{\sigma=r/\sqrt{D}}(\mathbf{x}) \right\|_2 = 0
$$

where $\mathbf{E}(\tilde{\mathbf{x}} = (\mathbf{x}, r))_{\mathbf{x}}$ is given in Equation 8.1. Further, given the same initial point, the trajectory of the PFGM++ ODE ($\mathrm{d}\mathbf{x}/\mathrm{d}r = \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r$) matches the diffusion ODE [27] ($\mathrm{d}\mathbf{x}/\mathrm{d}t = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p_{\sigma(t)}(\mathbf{x})$) in the same limit.

Proof. The $\mathbf{x}$ component in the Poisson field can be re-expressed as

$$
\begin{aligned}
\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}} &= \frac{1}{S_{N+D-1}(1)} \int \frac{\mathbf{x}-\mathbf{y}}{\|\tilde{\mathbf{x}}-\tilde{\mathbf{y}}\|^{N+D}} p(\mathbf{y}) d\mathbf{y} \\
&\propto \int p_r(\mathbf{x}|\mathbf{y})(\mathbf{x}-\mathbf{y}) p(\mathbf{y}) d\mathbf{y}
\end{aligned}
$$

where the perturbation kernel $p_r(\mathbf{x}|\mathbf{y}) \propto 1/(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}$. The direction of the score can also be written down in a similar form:

$$
\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}) = \frac{\int p_\sigma(\mathbf{x}|\mathbf{y}) \frac{\mathbf{y}-\mathbf{x}}{\sigma^2} p(\mathbf{y}) d\mathbf{y}}{p_\sigma(\mathbf{x})} \propto \int p_\sigma(\mathbf{x}|\mathbf{y})(\mathbf{x}-\mathbf{y}) p(\mathbf{y}) d\mathbf{y}
$$

where $p_\sigma(\mathbf{x}|\mathbf{y}) \propto \exp -\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}$. Since $p \in \mathcal{C}^1$, and obviously $p_r(\mathbf{x}|\mathbf{y}) \in C^1$, then

$\lim_{D\to\infty}\int p_r(\mathbf{x}|\mathbf{y})(\mathbf{x}-\mathbf{y})p(\mathbf{y})d\mathbf{y} = \int \lim_{D\to\infty} p_r(\mathbf{x}|\mathbf{y})(\mathbf{x}-\mathbf{y})p(\mathbf{y})d\mathbf{y}$. It suffices to prove that the perturbation kernel $p_r(\mathbf{x}|\mathbf{y})$ point-wisely converge to the Gaussian kernel $p_\sigma(\mathbf{x}|\mathbf{y})$, *i.e.*, $\lim_{D\to\infty} p_r(\mathbf{x}|\mathbf{y}) = p_\sigma(\mathbf{x}|\mathbf{y})$, to ensure $\mathbf{E}(\mathbf{x})_\mathbf{x} \propto \nabla_\mathbf{x} \log p_\sigma(\mathbf{x})$. Given $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^N$,

$$
\begin{aligned}
\lim_{D\to\infty} p_r(\mathbf{x}|\mathbf{y}) &\propto \lim_{D\to\infty} \frac{1}{(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}} \\
&= \lim_{D\to\infty} (\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{-\frac{N+D}{2}} \\
&\propto \lim_{D\to\infty} (1 + \frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{r^2})^{-\frac{N+D}{2}} \\
&= \lim_{D\to\infty} (1 + \frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{D\sigma^2})^{-\frac{N+D}{2}} && (r = \sigma\sqrt{D}) \\
&= \lim_{D\to\infty} \exp\left(-\frac{N+D}{2}\ln(1 + \frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{D\sigma^2})\right) \\
&= \lim_{D\to\infty} \exp\left(-\frac{N+D}{2}\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{D\sigma^2}\right) && (\lim_{D\to\infty} \tfrac{\|\mathbf{x}-\mathbf{y}\|_2^2}{D\sigma^2} = 0) \\
&= \exp -\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2} \\
&\propto p_\sigma(\mathbf{x}|\mathbf{y})
\end{aligned}
$$

Hence $\lim_{D\to\infty} p_r(\mathbf{x}|\mathbf{y}) = p_\sigma(\mathbf{x}|\mathbf{y})$, and we establish that $\mathbf{E}(\tilde{\mathbf{x}})_\mathbf{x} \propto \nabla_\mathbf{x} \log p_\sigma(\mathbf{x})$. We can rewrite the drift term in the PFGM++ ODE as

$$
\begin{aligned}
\lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \sqrt{D}\mathbf{E}(\tilde{\mathbf{x}})_\mathbf{x}/E(\tilde{\mathbf{x}})_r &= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\sqrt{D}\int p_r(\mathbf{x}|\mathbf{y})(\mathbf{x}-\mathbf{y})p(\mathbf{y})d\mathbf{y}}{\int p_r(\mathbf{x}|\mathbf{y})(-r)p(\mathbf{y})d\mathbf{y}} \\
&= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\sqrt{D}\int p_r(\mathbf{x}|\mathbf{y})(\mathbf{y}-\mathbf{x})p(\mathbf{y})d\mathbf{y}}{r p_r(\mathbf{x})} \\
&= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\sqrt{D}\int p_\sigma(\mathbf{x}|\mathbf{y})(\mathbf{y}-\mathbf{x})p(\mathbf{y})d\mathbf{y}}{r p_\sigma(\mathbf{x})} \\
&= \sigma\nabla_\mathbf{x} \log p_\sigma(\mathbf{x}) \qquad (\nabla_\mathbf{x} \log p_\sigma(\mathbf{x}) = \frac{\int p_\sigma(\mathbf{x}|\mathbf{y})\frac{\mathbf{y}-\mathbf{x}}{\sigma^2}p(\mathbf{y})d\mathbf{y}}{p_\sigma(\mathbf{x})})
\end{aligned}
\tag{A.85}
$$

which establishes the first part of the theorem. For the second part, by the change-of-variable

$d\sigma = dr/\sqrt{D}$, the PFGM++ ODE is

$$\begin{aligned}
\lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\sigma} &= \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}r} \cdot \frac{\mathrm{d}r}{\mathrm{d}\sigma} \\
&= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}} \cdot E(\tilde{\mathbf{x}})_r^{-1} \cdot \sqrt{D} \\
&= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\sigma \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})}{\sqrt{D}} \cdot \sqrt{D} \qquad \text{(by Equation A.85)} \\
&= \sigma \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})
\end{aligned}$$

which is equivalent to the diffusion ODE. □

A.5.4 Proof for Proposition 2

Proposition 2. *When $r = \sigma\sqrt{D}, D \to \infty$, the minimizer in the PFGM++ objective (Equation 8.4) is **equaivalent to** the minimizer in the weighted sum of denoising score matching objective (Equation 8.6)*

Proof. For $\forall \mathbf{x} \in \mathbb{R}^N$, the minimizer in PFGM++ objective (Equation 8.4) at point $\tilde{\mathbf{x}} = (\mathbf{x}, r)$ is

$$\begin{aligned}
f^*_{\theta,\text{PFGM++}}(\tilde{\mathbf{x}}) &= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\int p_r(\mathbf{x}|\mathbf{y}) \frac{\mathbf{x}-\mathbf{y}}{r/\sqrt{D}} p(\mathbf{y}) d\mathbf{y}}{p_r(\mathbf{x})} \\
&= \lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} \frac{\int p_\sigma(\mathbf{x}|\mathbf{y}) \frac{\mathbf{x}-\mathbf{y}}{r/\sqrt{D}} p(\mathbf{y}) d\mathbf{y}}{p_\sigma(\mathbf{x})} \\
&\qquad \text{(By Theorem 8, } \lim_{D\to\infty} p_r(\mathbf{x}|\mathbf{y}) = p_\sigma(\mathbf{x}|\mathbf{y})\text{)} \\
&= \frac{\int p_\sigma(\mathbf{x}|\mathbf{y}) \frac{\mathbf{x}-\mathbf{y}}{\sigma} p(\mathbf{y}) d\mathbf{y}}{p_\sigma(\mathbf{x})} \qquad \text{(A.86)}
\end{aligned}$$

On the other hand, the minimizer in denoising score matching at point $\mathbf{x}$ in noise level $\sigma = r/\sqrt{N+D}$ is

$$f^*_{\theta,\text{DSM}}(\mathbf{x}, \sigma) = \frac{\int p_\sigma(\mathbf{x}|\mathbf{y}) \frac{\mathbf{x}-\mathbf{y}}{\sigma} p(\mathbf{y}) d\mathbf{y}}{p_\sigma(\mathbf{x})} \qquad \text{(A.87)}$$

Combining Equation A.86 and Equation A.87, we have

$$\lim_{\substack{D\to\infty \\ r=\sigma\sqrt{D}}} f^*_{\theta,\text{PFGM++}}(\mathbf{x}, \sigma\sqrt{N+D}) = f^*_{\theta,\text{DSM}}(\mathbf{x}, \sigma)$$

□

A.6 Chapter 9

A.6.1 Green's Function Review

Math Basics

Fourier transformation The Green's functions of many PDEs can be found with Fourier transforms. For a scalar function $\phi(\mathbf{x}, t),\ \mathbf{x} \in \mathbb{R}^N$, we define the Fourier transformation

$$\tilde{\phi}(\mathbf{k}, t) = \mathcal{F}[\phi] \equiv \int \phi(\mathbf{x}, t) e^{-i\mathbf{k}\cdot\mathbf{x}} d^N\mathbf{x}, \tag{A.88}$$

and the inverse Fourier transformation

$$\phi(\mathbf{x}, t) = \mathcal{F}^{-1}[\tilde{\phi}] \equiv \frac{1}{(2\pi)^N} \int \tilde{\phi}(\mathbf{k}, t) e^{i\mathbf{k}\cdot\mathbf{x}} d^N\mathbf{k}. \tag{A.89}$$

One nice property of Fourier transformation is that derivatives in $\mathbf{x}$ space becomes multiplication in $\mathbf{k}$ space, i.e.,

$$\mathcal{F}[\nabla\phi] = -i\mathbf{k}\tilde{\phi},\ \mathcal{F}[\nabla^2\phi] = -|\mathbf{k}|^2\tilde{\phi} \tag{A.90}$$

So solving the PDE in the $\mathbf{k}$ space can be much simpler than in the $\mathbf{x}$ space. That said, transforming $\tilde{\phi}(\mathbf{k}, t)$ back to $\phi(\mathbf{x}, t)$ via inverse Fourier transformation may be hard.

Volume and Surface in high dimensions The n-dimensional unit ball $\mathcal{B}_N = \{\mathbf{x} : \sum_{i=1}^N x_i^2 \leq 1\}$ has volume $V_N = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n+1}{2})}$ and surface area $A_{N-1} = nV_n = \frac{2\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})}$. The N-dimensional volume element $d^N\mathbf{x}$ expressed in spherical coordinates is

$$d^N\mathbf{x} = r^{N-1}dr \prod_{i=1}^{N-2} \left[(\sin\theta_i)^{N-1-i}\, d\theta_i\right] d\varphi \tag{A.91}$$

where $r \equiv |\mathbf{x}|$, $\theta_i \in [0, \pi]$, for $i = 1, \cdots, N-2$, $\varphi \in [0, 2\pi)$. Usually one is interested in calculating the integral $\int f(\mathbf{x}) d^N\mathbf{x}$, so if $f(\mathbf{x})$ has axial symmetries except for θ_1, i.e., $f(\mathbf{x})$ is independent of $\theta_i (i \geq 2)$ and ϕ, the $\mathbb{R}^N$ space can be sliced into "rings" based on θ_1 and r,

and the ring at $(\theta_1 \to \theta_1 + d\theta_1, r \to r + dr)$ has the volume

$$d^N\mathbf{x} = 2\pi A_{N-2}(r\sin\theta_1)^{N-2} r d\theta_1 dr. \tag{A.92}$$

Diffusion Equation

The Green's function $G(\mathbf{x}, t)$ of the diffusion equation for $(\mathbf{x} \in \mathbb{R}^N)$ satisfies:

$$G_t - \nabla^2 G = \delta(\mathbf{x})\delta(t), \tag{A.93}$$

where for simplicity, we have assumed the source point $\mathbf{x}' = \mathbf{0}$. Using Fourier transformation Eq. (A.88), the transformed PDE becomes:

$$\tilde{G}_t + |\mathbf{k}|^2 \tilde{G} = \delta(t), \tag{A.94}$$

which is equivalent to

$$\tilde{G}_t + |\mathbf{k}|^2 \tilde{G} = 0 \ (t > 0), \ \tilde{G}(\mathbf{k}, 0) = 1. \tag{A.95}$$

This initial value problem has the solution

$$\tilde{G}(\mathbf{k}, t) = \exp(-|\mathbf{k}|^2 t) \tag{A.96}$$

which is a Gaussian distribution in $\mathbf{k}$ space. Now transforming back to $\mathbf{x}$ space:

$$\begin{aligned}
G(\mathbf{x}, t) &= \mathcal{F}^{-1}[\tilde{G}] \\
&= \frac{1}{(2\pi)^N} \int_{\mathbf{k}} \exp(-|\mathbf{k}|^2 t) \exp(i\mathbf{k}\cdot\mathbf{x}) d^N\mathbf{k} \\
&= \frac{1}{(2\pi)^N} \int_{\mathbf{k}} \exp(-k^2 t) \exp(ikx\cos\theta) A_{N-2}(k\sin\theta)^{N-2} k dk d\theta \quad (k \equiv |\mathbf{k}|, x \equiv |\mathbf{x}|) \quad \text{(invoke Eq. A.92)} \\
&= \frac{A_{N-2}}{(2\pi)^N} \int_{k=0}^{\infty} dk k^{N-1} \exp(-k^2 t) \int_{\theta=0}^{\pi} \exp(ikx\cos\theta)(\sin\theta)^{N-2} d\theta,
\end{aligned} \tag{A.97}$$

and the integral of θ is:

$$\int_{\theta=0}^{\pi} \exp(ikx\cos\theta)(\sin\theta)^{N-2}d\theta = \sqrt{\pi}\Gamma(\frac{N-1}{2})_0\tilde{\Gamma}_1(\frac{N}{2}, -\frac{(kx)^2}{4}), \tag{A.98}$$

where ${}_0\tilde{\Gamma}_1(b,z)$ is a regularized hypergeometric function. So

$$\begin{aligned} G(\mathbf{x},t) &= \frac{A_{N-2}\sqrt{\pi}\Gamma(\frac{N-1}{2})}{(2\pi)^N}(\int_{k=0}^{\infty} k^{N-1}\exp(-k^2t)_0\tilde{\Gamma}_1(\frac{N}{2}, -\frac{(kx)^2}{4})) \\ &= \frac{A_{N-2}\sqrt{\pi}\Gamma(\frac{N-1}{2})}{(2\pi)^N}(\frac{1}{2}\exp(-\frac{x^2}{4t})t^{\frac{N}{2}}) \\ &= \frac{1}{(4\pi t)^{\frac{N}{2}}}\exp(-\frac{r^2}{4t}) \\ &= \frac{1}{(4\pi t)^{\frac{N}{2}}}\exp(-\frac{|\mathbf{x}|^2}{4t}) \end{aligned} \tag{A.99}$$

which is a Gaussian distribution with zero mean and variance $2t$. In fact, there is a much simpler way to compute the inverse Fourier transform, by noticing that the components of k are separable:

$$\begin{aligned} G(\mathbf{x},t) &= \mathcal{F}^{-1}[\tilde{G}] \\ &= \frac{1}{(2\pi)^N}\int_{\mathbf{k}} \exp(-|\mathbf{k}|^2t)\exp(i\mathbf{k}\cdot\mathbf{x})d^N\mathbf{k} \\ &= \prod_{i=1}^{N}(\frac{1}{2\pi}\int_{-\infty}^{\infty}\exp(-k_i^2t)\exp(ik_ix_i)dk_i) \\ &= \prod_{i=1}^{N}(\frac{\exp(-\frac{x_i^2}{4t})}{2\pi}\int_{-\infty}^{\infty}\exp(-t(k_i-\frac{ix_i}{2t})^2)dk_i) \\ &= \prod_{i=1}^{N}(\frac{\exp(-\frac{x_i^2}{4t})}{2\pi}\sqrt{\frac{\pi}{t}}) \\ &= \frac{1}{(4\pi t)^{\frac{N}{2}}}\exp(-\frac{|\mathbf{x}|^2}{4t}) \end{aligned} \tag{A.100}$$

However, the more difficult derivation is more general, and is useful when we attempt to interpolate between DMs and PFGMs (see Appendix A.6.2). For general $\mathbf{x}'$, we thus have

$$G(\mathbf{x},t;\mathbf{x}') = \frac{1}{(4\pi t)^{\frac{N}{2}}}\exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{4t}) \tag{A.101}$$

For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x}, t) = \int p_{\text{data}}(\mathbf{x}')G(\mathbf{x}, t; \mathbf{x}')d^N\mathbf{x}'$. We now check the three conditions:

[density flow condition] holds. $p(\mathbf{x}, t; \mathbf{x}') = \phi(\mathbf{x}, t; \mathbf{x}') \geq 0$.

[smooth condition] holds. We have

$$F(\mathbf{x}_1', \mathbf{x}_2', T) \equiv \int \sqrt{p(\mathbf{x}, T; \mathbf{x}_1')}\sqrt{p(\mathbf{x}, T; \mathbf{x}_2')}d^N\mathbf{x} = \exp(-\frac{|\mathbf{x}_1 - \mathbf{x}_2|^2}{8T}), \tag{A.102}$$

so $\lim_{T\to\infty} F(\mathbf{x}_1', \mathbf{x}_2', T) \to 1$, implying data-independent priors.

Poisson Equation

The Green's function $\phi(\mathbf{x})$ of the Poisson equation for $(\mathbf{x} \in \mathbb{R}^N)$ satisfies:

$$\nabla^2 G = \delta(\mathbf{x}) \tag{A.103}$$

where for simplicity we have assumed the source point $\mathbf{x}' = \mathbf{0}$. Note that (so far) $G(\mathbf{x})$ does not depend on time t, since Poisson equation physically describes steady states (which are time independent). To bake in the notion of time t, PFGM [50] augments $\mathbf{x} \in \mathbb{R}$ to be $\tilde{\mathbf{x}} \equiv (\mathbf{x}, t) \in \mathbb{R}^{N+1}$ [1], where $\mathbf{x}$ is the data point, and t is the augmented dimension. By splitting $\mathbf{x}$ and t explicitly, the Poisson equation becomes:

$$G_{tt} + \nabla^2 G = 0, \quad G(\mathbf{x}, 0) = \delta(\mathbf{x}). \tag{A.104}$$

Using Fourier transformation Eq. (A.88) , the transformed PDE becomes:

$$\tilde{G}_{tt} - |\mathbf{k}|^2\tilde{G} = 0, \quad \tilde{G}_t(\mathbf{k}, 0) = 1, \tag{A.105}$$

whose general solution is $\tilde{G} = \frac{1}{|\mathbf{k}|}(A\text{exp}(-|\mathbf{k}|t) + B\text{exp}(|\mathbf{k}|t))$ with $-A + B = 1$. Considering the free boundary condition, i.e., as $t \to \infty, \tilde{G} \to 0$, we have $A = -1, B = 0$. So $\tilde{G} = -\frac{1}{|\mathbf{k}|}\text{exp}(-|\mathbf{k}|t)$. Now transforming back to $\mathbf{x}$ space:

[1] The original PFGM [50] uses z instead of t.

$$
\begin{aligned}
G(\mathbf{x},t) &= \mathcal{F}^{-1}[\tilde{G}] \\
&= \frac{1}{(2\pi)^N}\int_{\mathbf{k}} \frac{1}{|\mathbf{k}|}\exp(-|\mathbf{k}|t)\exp(i\mathbf{k}\cdot\mathbf{x})d^N\mathbf{k} \\
&= \frac{1}{(2\pi)^N}\int_{\mathbf{k}} \frac{1}{k}\exp(-kt)\exp(ikx\cos\theta) \\
&\quad A_{N-2}(k\sin\theta)^{N-2}kdkd\theta \quad (k\equiv|\mathbf{k}|, x\equiv|\mathbf{x}|) \quad \text{(invoke Eq. A.92)} \\
&= \frac{A_{N-2}}{(2\pi)^N}\int_{k=0}^{\infty} dkk^{N-2}\exp(-kt)\int_{\theta=0}^{\pi}\exp(ikx\cos\theta)(\sin\theta)^{N-2}d\theta, \\
&= \frac{A_{N-2}\sqrt{\pi}\Gamma(\frac{N-1}{2})}{(2\pi)^N}(\int_{k=0}^{\infty} k^{N-2}\exp(-kt){}_0\tilde{\Gamma}_1(\frac{N}{2},-\frac{(kx)^2}{4})) \\
&= \frac{2}{(4\pi)^{\frac{N}{2}}}(\frac{2^{N-2}\Gamma(\frac{N-1}{2})}{\sqrt{\pi}}\frac{1}{(t^2+x^2)^{\frac{N-1}{2}}}) \\
&= \frac{\Gamma(\frac{N-1}{2})}{2\pi^{\frac{N+1}{2}}}\frac{1}{(t^2+x^2)^{\frac{N-1}{2}}} \\
&= \frac{\Gamma(\frac{N-1}{2})}{2\pi^{\frac{N+1}{2}}}\frac{1}{(t^2+|\mathbf{x}|^2)^{\frac{N-1}{2}}},
\end{aligned}
\tag{A.106}
$$

which is the Poisson kernel in PFGM [50]. For general $\mathbf{x}'$, we have

$$
G(\mathbf{x},t;\mathbf{x}) = \frac{\Gamma(\frac{N-1}{2})}{2\pi^{\frac{N+1}{2}}}\frac{1}{(t^2+|\mathbf{x}-\mathbf{x}'|^2)^{\frac{N-1}{2}}} \tag{A.107}
$$

For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x},t) = \int p_{\text{data}}(\mathbf{x}')G(\mathbf{x},t;\mathbf{x}')d^N\mathbf{x}'$. We now check the three conditions:

[density flow condition] holds.

[smooth condition] holds. Although $F(\mathbf{x}_1',\mathbf{x}_2',t)$ may not have a closed form for any $t>0$, we notice that for large T:

$$
p(\mathbf{x},T;\mathbf{x}') \sim \frac{1}{(1+\frac{|\mathbf{x}-\mathbf{x}'|^2}{T^2})^{\frac{N+1}{2}}} \approx \exp(-\frac{(N+1)|\mathbf{x}-\mathbf{x}'|^2}{2T^2}), \tag{A.108}
$$

so we can show $\lim_{T\to\infty} F(\mathbf{x}_1',\mathbf{x}_2',T) \equiv \int\sqrt{p(\mathbf{x},T;\mathbf{x}_1')}\sqrt{p(\mathbf{x},T;\mathbf{x}_2')}d^N\mathbf{x} \to 1$ similar to the diffusion equation case.

Wave Equation

The Green's function $G(\mathbf{x}, t)$ of the wave equation for ($\mathbf{x} \in \mathbb{R}^N$) satisfies:

$$G_{tt} - \nabla^2 G = \delta(\mathbf{x})\delta(t), \tag{A.109}$$

where for simplicity we have assumed the source point $\mathbf{x}' = \mathbf{0}$. Using Fourier transformation Eq. (A.88) , the transformed PDE becomes:

$$\tilde{G}_{tt} + |\mathbf{k}|^2 \tilde{G} = 0, \quad \tilde{G}_t(\mathbf{k}, 0) = 1, \quad \tilde{G}(\mathbf{k}, 0) = 0 \tag{A.110}$$

whose solution is $\tilde{G} = \frac{1}{|\mathbf{k}|}(\sin(|\mathbf{k}|t)$. Now transforming back to $\mathbf{x}$ space:

$$\begin{aligned} G(\mathbf{x}, t) &= \mathcal{F}^{-1}[\tilde{G}] \\ &= \frac{1}{(2\pi)^N} \int_{\mathbf{k}} \frac{1}{|\mathbf{k}|} \sin(|\mathbf{k}|t) \exp(i\mathbf{k} \cdot \mathbf{x}) d^N \mathbf{k} \\ &= \frac{A_{N-2}\sqrt{\pi}\Gamma(\frac{N-1}{2})}{(2\pi)^N} (\int_{k=0}^{\infty} k^{N-2} \sin(kt) {}_0\tilde{\Gamma}_1(\frac{N}{2}, -\frac{(kx)^2}{4})) \end{aligned} \tag{A.111}$$

Doing the integration is not easy, but we refer readers to [229] for the ideal wave solutions in arbitrary N dimensions, and [230] for dissipative waves. Here we summarize the main results in [229]. We denote the Green's function in N-dimensions by ϕ_n, then solutions differing by 2 dimensions are related ($r \equiv |\mathbf{x}|$):

$$G_{N+2}(r, t) = -\frac{1}{2\pi r} \frac{\partial G_N(r, t)}{\partial r} \tag{A.112}$$

The solutions for $N \leq 5$ are listed ($\tau \equiv t - r$):

$$
\begin{aligned}
G_1 &= \frac{1}{2}\Theta(\tau) \\
G_2 &= \frac{1}{2\pi}\frac{\Theta(\tau)}{\sqrt{t^2 - r^2}} \\
G_3 &= \frac{1}{4\pi}\frac{\delta(\tau)}{r} \\
G_4 &= \frac{1}{4\pi^2}\left(\frac{\delta(\tau)}{r(t^2 - r^2)^{\frac{1}{2}}} - \frac{\Theta(t-r)}{(t^2 - r^2)^{\frac{3}{2}}}\right) \\
G_5 &= \frac{1}{8\pi^2}\left(\frac{\delta(\tau)}{r^3} + \frac{\delta'(\tau)}{r^2}\right)
\end{aligned}
\tag{A.113}
$$

One interesting observation is that solutions in even and odd dimension have qualitative differences. When $N = 1, 3, 5, ...$ is odd, the solution only contains $\delta(\tau)$ and its derivatives (the only exception is $N = 1$), meaning only the wave front $t = r$ is excited, with everywhere else zero. By contrast, when $N = 2, 4, ...$ is even, the solution contains the step function $\Theta(\tau)$, which does not vanish for $t > r$, is referred to as "wake" [229]. For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x}, t) = \int p_{\text{data}}(\mathbf{x}')G(\mathbf{x}, t; \mathbf{x}')d^N\mathbf{x}'$.

[density flow condition] fails. In Table 9.1, we match $p \equiv -\phi_t$, but is p a valid probability density distribution? We check the 2D case

$$
p \equiv -\phi_t = \frac{1}{2\pi}\left(\frac{t\Theta(t-r)}{(t^2 - r^2)^{\frac{3}{2}}} - \frac{\delta(t-r)}{\sqrt{t^2 - r^2}}\right). \tag{A.114}
$$

The second term is negative. Beyond the wave front $r > t$, $p = 0$, so $\mathbf{v} = \frac{\nabla\phi}{p}$ is ill-defined.

[smooth condition] fails. For $\mathbf{x}'_1 \neq \mathbf{x}'_2$, even the support of $p(\mathbf{x}, t; \mathbf{x}'_1)$ and $p(\mathbf{x}, t; \mathbf{x}'_2)$ do not match.

Helmholtz Equation

The Helmholtz equation is $(\nabla^2 + k_0^2)\phi = 0$. It is easy to see that $k_0 = 0$ recovers the Poisson equation. From the physics perspective, the Helmholtz equation can be interpreted as single-frequency wave equation, or single-energy Schrödinger equation.

Relation to wave equation The wave equation is $\phi_{tt} - \nabla^2\phi = 0$. It is usually interesting to study periodic solutions (in time) $\phi(\mathbf{x}) = f(\mathbf{x}, t)e^{-ik_0 t}$ with the angular frequency k_0.

Inserting the ansatz to the wave equation gives $(\nabla^2 + k_0^2)f = 0$, which is the Helmholtz equation.

Relation to Schrödinger equation The Schrödinger equation of a free particle is $i\hbar\frac{\partial\phi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\phi$ where ϕ is the wave function. It is usually interesting to study steady states, such that $\phi(\mathbf{x}, t) = f(\mathbf{x})e^{-iEt/\hbar}$. Inserting the steady-state ansatz to the Schrödinger equation gives $(\nabla^2 + \frac{2mE}{\hbar^2})f = 0$, which is the Helmholtz equation with $k_0 = \frac{\sqrt{2mE}}{\hbar}$.

The Green's function is the solution to (for simplicity, we set $\mathbf{x}' = \mathbf{0}$):

$$\nabla^2 G(\mathbf{x}) + k_0^2 G(\mathbf{x}) = \delta(\mathbf{x}) \tag{A.115}$$

The solution can be found at [249]:

$$G(\mathbf{x}) = \frac{i}{4}(\frac{k_0}{2\pi r})^{\frac{N}{2}-1} H^{(1)}_{\frac{N}{2}-1}(k_0 r), \quad r \equiv |\mathbf{x}| \tag{A.116}$$

where $H^{(1)}$ is first kind Hankel function. $H^{(1)}_n \equiv J_n + iY_n$ where J_n and Y_n are first-kind and second-kind Bessel functions, respectively. When $k \to 0$, $kr \to 0$, $H^{(1)}_{\frac{n}{2}-1}(k_0 r) \approx -\frac{i\Gamma(\frac{N}{2}-1)}{\pi}(\frac{2}{k_0 r})^{\frac{N}{2}-1}$, so $\phi(\mathbf{x}) \sim \frac{1}{r^{N-2}}$, which is the Green's function of the Poisson equation. Similar to the Poisson equation, we identify one dimension in $\mathbf{x}$ as time t, and we change $N \to N+1$, $\mathbf{x} \to \mathbf{x}' = [\mathbf{x}, t]$. This means the equation

$$G_{tt} + \nabla^2 G + k_0^2 G = \delta(\mathbf{x} - \mathbf{x}'), \quad \mathbf{x} \in \mathbb{R}^N \tag{A.117}$$

has the solution

$$G(\mathbf{x}, t; \mathbf{x}') = \frac{i}{4}(\frac{k_0}{2\pi\sqrt{t^2 + r^2}})^{\frac{N-1}{2}} H^{(1)}_{\frac{N-1}{2}}(k_0\sqrt{t^2 + r^2}), \quad r \equiv |\mathbf{x} - \mathbf{x}'|. \tag{A.118}$$

For simplicity, we only consider the real part (which is still a solution to the Helmholtz equation):

$$G(\mathbf{x}, t; \mathbf{x}') = -\frac{1}{4}(\frac{k_0}{2\pi\sqrt{t^2 + r^2}})^{\frac{N-1}{2}} Y_{\frac{N-1}{2}}(k_0\sqrt{t^2 + r^2}), \quad r \equiv |\mathbf{x} - \mathbf{x}'|. \tag{A.119}$$

For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x}, t) = \int p_{\text{data}}(\mathbf{x}')G(\mathbf{x}, t; \mathbf{x}')d^N\mathbf{x}'$.

[density flow condition] Conditionally holds. Note G is a decreasing function of t for small (t, r), because $\frac{k_0}{2\pi\sqrt{t^2+r^2}}$ decreases with increasing t, $-Y_{\frac{N-1}{2}}(k_0\sqrt{t^2+r^2})$ is a decreasing (positive) function of t for some range $k_0\sqrt{t^2+r^2} \leq r_c$, where r_c is the first zero of $Y_{\frac{N-1}{2}}$. So $p(\mathbf{x}, t, \mathbf{x}') \equiv -\phi_t(\mathbf{x}, t; \mathbf{x}') \geq 0$ for $k_0\sqrt{t^2+r^2} \leq r_c$. When $k_0\sqrt{t^2+r^2} > r_c$, $Y_{\frac{N-1}{2}}(k_0\sqrt{t^2+r^2})$ oscillates and can change sign. For the (t, r) region one is interested in, it suffices to choose $k_0 \leq \frac{r_c}{(\sqrt{t^2+r^2})_{\max}}$ to make sure that p is a valid density distribution.

[smooth condition] Conditionally holds. As long as k_0 is small enough (equivalently, phase space is properly clipped), p defines a density distribution.

Screened Poisson Equation

The screened Poisson equation is $(\nabla^2 - m^2)\phi = 0$. The screened Poisson equation is very similar to the Helmholtz equation, the only difference being the sign within the brackets. $m = 0$ recovers the Poisson equation. m can be interpreted as the screening magnitude, as in Thomas-Fermi screening or Debye screening; or m can be interpreted as the mass of bosons, as in Yukawa potential.

Relation to Klein-Gordon equation The Klein-Gordon (KG) equation is $\phi_{tt} - \nabla^2\phi + m^2\phi = 0$. Time-independent solutions (hence $\phi_{tt} = 0$) of KG is the screened Poisson equation.

The Green's function is the solution to (for simplicity, we have set $\mathbf{x}' = \mathbf{0}$):

$$\nabla^2 G(\mathbf{x}) - m^2 G(\mathbf{x}) = \delta(\mathbf{x}) \tag{A.120}$$

The solution can be found at [250]:

$$G(\mathbf{x}) = -\frac{1}{(2\pi)^{\frac{N}{2}}}(\frac{m}{r})^{\frac{N}{2}-1}K_{\frac{N}{2}-1}(mr), \quad r \equiv |\mathbf{x}| \tag{A.121}$$

where K_n is the second kind modified Bessel functions. When $x \ll 1$, $K_n(x) \sim \frac{1}{x^n}$. When $m \to 0$, $K_{\frac{N}{2}-1}(mr) \sim \frac{1}{(mr)^{\frac{N}{2}-1}}$, so $G(\mathbf{x}) \sim \frac{1}{r^{N-1}}$ recovers the Poisson kernel.

Similar to the Poisson equation, we identify one dimension in $\mathbf{x}$ as time t, and we change

$N \to N+1$, $\mathbf{x} \to \mathbf{x}' = [\mathbf{x}, t]$. This means the equation

$$G_{tt} + \nabla^2 G - m^2 G = \delta(\mathbf{x} - \mathbf{x}'), \quad \mathbf{x} \in \mathbb{R}^N \tag{A.122}$$

has the solution

$$G(\mathbf{x}, t; \mathbf{x}') = \frac{1}{(2\pi)^{\frac{N+1}{2}}} \left(\frac{m}{\sqrt{t^2 + r^2}}\right)^{\frac{N-1}{2}} K_{\frac{N-1}{2}}(m\sqrt{t^2 + r^2}), \quad r \equiv |\mathbf{x} - \mathbf{x}'| \tag{A.123}$$

For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x}, t) = \int p_{\text{data}}(\mathbf{x}') G(\mathbf{x}, t; \mathbf{x}') d^N \mathbf{x}'$

[density flow condition] holds. $p(\mathbf{x}, t; \mathbf{x}') \equiv -\phi_t > 0$, since both $\left(\frac{m}{\sqrt{t^2+r^2}}\right)^{\frac{N+1}{2}}$ and $K_{\frac{N-1}{2}}(m\sqrt{t^2 + r^2})$ are positive and decreasing functions of t.

[smooth condition] holds, since the dispersion relation $\omega(k) = -i\sqrt{m^2 + k^2}$ satisfies the smoothing condition.

Schrödinger Equation

The Schrödinger equation is $i\phi_t = -\nabla^2 \phi + V(\mathbf{x})\phi$, where ϕ is the (complex) wave function. It describes the evolution of a quantum particle [232]. For $V(\mathbf{x}) = 0$, the free particle Schrödinger equation is $i\phi_t = -\nabla^2 \phi$. Note that since ϕ is a complex scalar function, it is not a valid probability distribution. In fact, quantum mechanics interpret $|\phi|^2$ as the probability distribution. Now we aim to calcuate $\frac{\partial |\phi|^2}{\partial t}$ from the Schrödinger equation:

$$i\phi_t + \nabla^2 \phi = 0 \tag{A.124}$$

Taking the complex conjugate gives

$$-i\phi_t^* + \nabla^2 \phi^* = 0 \tag{A.125}$$

Multiplying ϕ^* to Eq. (A.124) and multiplying ϕ to Eq. (A.125) and subtracting the two gives:

$$i(\phi^* \phi_t + \phi \phi_t^*) + \phi^* \nabla^2 \phi - \phi \nabla^2 \phi^* = 0. \tag{A.126}$$

Note that

$$\phi^*\phi_t + \phi\phi_t^* = \frac{\partial|\phi|^2}{\partial t}, \phi^*\nabla^2\phi - \phi\nabla^2\phi^* = \nabla\cdot(\phi^*\nabla\phi - \phi\nabla\phi^*), \tag{A.127}$$

Eq. (A.126) can simplify to

$$i\frac{\partial|\phi|^2}{\partial t} + \nabla\cdot(\phi^*\nabla\phi - \phi\nabla\phi^*) = i\frac{\partial|\phi|^2}{\partial t} + i\nabla\cdot(|\phi|^2(2\text{Im}\nabla\log\phi)) = 0. \tag{A.128}$$

By comparing to Eq. (9.2), we have $p = |\phi|^2$, $\mathbf{v} = 2\text{Im}\nabla\log\phi$ and $R = 0$. The Green's function $G(\mathbf{x}, t; \mathbf{x}') = \frac{1}{(4it)^{N/2}}\exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{4it})$, which can be obtained by simply $t \to it$ in the diffusion kernel. For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x}, t) = \int p_{\text{data}}(\mathbf{x}')G(\mathbf{x}, t; \mathbf{x}')d^N\mathbf{x}'$.

[density flow condition] fails. $p \equiv |\phi|^2$ is a probability distribution, but it may have zeros due to interference, resulting divergence of $\mathbf{v}$.

[smooth condition] fails. $p(\mathbf{x}, T)$ oscillates restlessly even for large T and dependent on the initial distribution.

A.6.2 Interpolating Between DMs and PFGMs

Let's study this PDE:

$$a\phi_{tt} - b\phi_t + \nabla^2\phi = \delta(\mathbf{x})\delta(t), \tag{A.129}$$

where $(a, b) = (1, 0)$ and $(a, b) = (0, 1)$ recovers the Poisson equation and the diffusion equation, respectively. The Fourier transformation Eq. (A.88) converts the PDE to

$$a\tilde{\phi}_{tt} - b\tilde{\phi}_t - |\mathbf{k}|^2\tilde{\phi} = \delta(t), \tag{A.130}$$

or equivalently,

$$a\tilde{G}_{tt} - b\tilde{G}_t - |\mathbf{k}|^2\tilde{G} = 0, \quad (a\tilde{G}_t - b\tilde{G})(\mathbf{k}, 0) = 1. \tag{A.131}$$

Inserting the trial equation $\tilde{G} = A\exp(-\omega t)$ to above gives

$$a\omega^2 + b\omega - |\mathbf{k}|^2 = 0, -\omega aA - bA = 1 \tag{A.132}$$

The quadratic equation has the solution $\omega_\pm = \frac{1}{2a}(-b \pm \sqrt{b^2+4|\mathbf{k}|^2})$ where only w_+ is consistent with the free boundary condition ($\tilde{\phi} \to 0$, when $t \to \infty$). So $\omega \equiv \omega_+ = \frac{1}{2a}(-b+\sqrt{b^2+4|\mathbf{k}|^2})$, and $A = -\frac{1}{\omega a + b} = -\frac{2}{b+\sqrt{b^2+4|\mathbf{k}|^2}}$,

$$\tilde{G}(\mathbf{k},t) = -\frac{2}{b+\sqrt{b^2+4|\mathbf{k}|^2}}\exp(-\frac{1}{2a}(\sqrt{b^2+4|\mathbf{k}|^2}-b)t) \tag{A.133}$$

Now we try to transform back to $\mathbf{x}$ space:

$$\begin{aligned}
G(\mathbf{x},t) &= \mathcal{F}^{-1}[\tilde{G}] \\
&= \frac{1}{(2\pi)^N}\int_{\mathbf{k}} -\frac{2}{b+\sqrt{b^2+4|\mathbf{k}|^2}}\exp(-\frac{1}{2a}(\sqrt{b^2+4|\mathbf{k}|^2}-b)t)\exp(i\mathbf{k}\cdot\mathbf{x})d^N\mathbf{k} \\
&= -\frac{A_{N-2}}{(2\pi)^N}\int_{k=0}^{\infty} dk \frac{k^{N-1}}{b+\sqrt{b^2+4k^2}}\exp(-\frac{1}{2a}(\sqrt{b^2+4k^2}-b)t) \\
&\qquad \int_{\theta=0}^{\pi}\exp(ikx\cos\theta)(\sin\theta)^{N-2}d\theta \\
&= -\frac{A_{N-2}\sqrt{\pi}\Gamma(\frac{N-1}{2})}{(2\pi)^N}(\int_{k=0}^{\infty}\frac{k^{N-1}}{b+\sqrt{b^2+4k^2}}\exp(-\frac{1}{2a}(\sqrt{b^2+4k^2}-b)t){}_0\tilde{\Gamma}_1(\frac{N}{2},-\frac{(kx)^2}{4}))
\end{aligned} \tag{A.134}$$

To the best of the our knowledge and Wolfram Mathematica's ability, the integral does not have a closed from. For general data distribution $p_{\text{data}}(\mathbf{x})$, we have $\phi(\mathbf{x},t) = \int p_{\text{data}}(\mathbf{x}')G(\mathbf{x},t;\mathbf{x}')d^N\mathbf{x}'$

[density flow condition] holds.

[smooth condition] holds. The dispersion relation $\omega = \frac{i}{2a}(b-\sqrt{b^2+4ak^2})$ satisfies Eq. (9.8).

A.6.3 Smooth Condition and Dispersion Relations

Define the overlap between $p(\mathbf{x},t;\mathbf{x}_i')$ $(i=1,2)$:

$$F(\mathbf{x}_1',\mathbf{x}_2',t) \equiv \frac{\int p(\mathbf{x},t;\mathbf{x}_1')p(\mathbf{x},t;\mathbf{x}_2')d^N\mathbf{x}}{\int p(\mathbf{x},t;\mathbf{x}_1')p(\mathbf{x},t;\mathbf{x}_1')d^N\mathbf{x}}, \tag{A.135}$$

then the density flow condition requires that $\lim_{t\to\infty} F(\mathbf{x}'_1, \mathbf{x}'_2, t) \to 1$. We now attempt to rewrite Eq. (A.135) in terms of Fourier bases, where dispersion relation should emerge. Define

$$\tilde{p}(\mathbf{k}, t; \mathbf{x}'_i) = \int p(\mathbf{x}, t; \mathbf{x}'_i) \exp(-i\mathbf{k}\cdot\mathbf{x}) d^N\mathbf{x}, \quad i = 1, 2 \tag{A.136}$$

Note that $p(\mathbf{x}, t; \mathbf{x}'_i)$ $(i = 1, 2)$ differ only by a translation, so their Fourier function differs only by a phase

$$p(\mathbf{k}, t; \mathbf{x}'_2) = p(\mathbf{k}, t; \mathbf{x}'_1) \exp(i\mathbf{k}\cdot(\mathbf{x}'_1 - \mathbf{x}'_2)) \tag{A.137}$$

Due to the unitarity of Fourier transformation, the integration in $\mathbf{x}$ can be converted to integration in $\mathbf{k}$, so

$$\begin{aligned} F(\mathbf{x}'_1, \mathbf{x}'_2, t) &= \frac{\int p^*(\mathbf{k}, t; \mathbf{x}'_1) p(\mathbf{k}, t; \mathbf{x}'_2) d^N\mathbf{k}}{\int p^*(\mathbf{k}, t; \mathbf{x}'_1) p(\mathbf{k}, t; \mathbf{x}'_1) d^N\mathbf{k}} \\ &= \frac{\int p^*(\mathbf{k}, t; \mathbf{x}'_1) p(\mathbf{k}, t; \mathbf{x}'_1) \exp(i\mathbf{k}\cdot(\mathbf{x}'_1 - \mathbf{x}'_2)) d^N\mathbf{k}}{\int p^*(\mathbf{k}, t; \mathbf{x}'_1) p(\mathbf{k}, t; \mathbf{x}'_1) d^N\mathbf{k}} \\ &= \frac{\int |p(\mathbf{k}, t; \mathbf{x}'_1)|^2 \cos(\mathbf{k}\cdot(\mathbf{x}'_1 - \mathbf{x}'_2)) d^N\mathbf{k}}{\int |p(\mathbf{k}, t; \mathbf{x}'_1)|^2 d^N\mathbf{k}} \\ &= \langle \cos(\mathbf{k}\cdot(\mathbf{x}'_1 - \mathbf{x}'_2)) \rangle_{|p(\mathbf{k}, t; \mathbf{x}'_1)|^2} \end{aligned} \tag{A.138}$$

where is the expected value of $\cos(\mathbf{k}\cdot(\mathbf{x}'_1 - \mathbf{x}'_2))$ over (unnormalized) distribution $|p(\mathbf{k}, t; \mathbf{x}'_1)|^2$. Note that

$$p(\mathbf{k}, t; \mathbf{x}'_1) = \exp(-i\omega(k)t) p(\mathbf{k}, 0; \mathbf{x}'_1) = \exp(-i\mathrm{Re}\ \omega(k)t) \exp(\mathrm{Im}\ \omega(k)t) p(\mathbf{k}, t; \mathbf{x}'_1), \tag{A.139}$$

we have

$$|p(\mathbf{k}, t; \mathbf{x}'_1)|^2 = \exp(2\mathrm{Im}\ \omega(k)t) \tag{A.140}$$

where we used $|p(\mathbf{k}, 0; \mathbf{x}'_1)|^2 = 1$. Define

$$k^* \equiv \operatorname*{argmax}_k \mathrm{Im}\ \omega(k), \tag{A.141}$$

as t increases, $|p(\mathbf{k}, t; \mathbf{x}_1')|^2$ has increasingly more probability concentrated around $k = k^*$. As a result,

$$\lim_{t\to\infty} F(\mathbf{x}_1', \mathbf{x}_2', t) = \langle \cos(\mathbf{k} \cdot (\mathbf{x}_1' - \mathbf{x}_2')) \rangle_{|\mathbf{k}|=k^*}, \tag{A.142}$$

where the averaging is over the sphere $|\mathbf{k}| = k^*$. The limit is 1 if and only if $k^* = 0$ for $\mathbf{x}_1' \neq \mathbf{x}_2'$. $k^* = 0$ is equivalent to

$$\boxed{\text{Im } \omega(k) < \text{Im } \omega(0), \quad \text{for all } k > 0.} \tag{A.143}$$

The physical interpretation is that waves of $k > 0$ should decay faster than $k = 0$.

Appendix B

Additional Details and Results

B.1 Chapter 3

B.1.1 Experimental Details

In this section, we include more details about the training and sampling of score-based models by the STF and DSM objectives.

Details for the Experiments in Section 3.2

In Section 3.2, we demonstrate the behavior of $V_{\mathrm{DSM}}(t)$ in the three phases on Two Gaussians and a subset of CIFAR-10. Here we provide more details about the two datasets.

The distribution of the two Gaussian is $\frac{1}{2}\mathcal{N}(\boldsymbol{\mu}, \hat{\sigma}^2\boldsymbol{I}_{64\times 64}) + \frac{1}{2}\mathcal{N}(-\boldsymbol{\mu}, \hat{\sigma}^2\boldsymbol{I}_{64\times 64})$, where $\boldsymbol{\mu} = 0.1 \cdot \mathbf{1} \in \mathbb{R}^{64}$, and $\hat{\sigma} = 1e-4$. We estimate all the integrals in Equation 3.3 by sampling 1k points from the corresponding distributions. For the subset of CIFAR-10, we uniformly sample 4096 images from CIFAR-10 dataset, and assign uniform distribution on the discrete set. We also approximate $V_{\mathrm{DSM}}(t)$ by Monte Carlo estimation and sample 200 perturbations for each t. We use VE SDE for all simulations, and set $\sigma_m = 1e-2, \sigma_M = 50$.

Interestingly, $V_{\mathrm{DSM}}(t)$ is relatively large for the Two Gaussians distribution compared to CIFAR-10 (see Figure 3.2b) when $t \to 0$. This can be explained by their continuous and

discrete natures. For the two Gaussian distribution, we can rewrite $V_{\mathrm{DSM}}(t)$ as

$$V_{\mathrm{DSM}}(t) = \mathbb{E}_{\tilde{\mathbf{x}}\sim p_t(\tilde{\mathbf{x}})}\left[\frac{\hat{\sigma}^2}{\sigma_t^2(\sigma_t^2+\hat{\sigma}^2)} + 4\frac{\alpha(\tilde{\mathbf{x}})(1-\alpha(\tilde{\mathbf{x}}))\|\boldsymbol{\mu}\|\sigma_t^4}{\sigma_t^4+\hat{\sigma}^2}\right]$$

where $\alpha(\mathbf{x}) = \frac{1}{1+\exp^{-4\mathbf{x}^T\boldsymbol{\mu}}}$ can be regarded as the probability that $\mathbf{x}$ comes from the Gaussian component $\mathcal{N}(\boldsymbol{\mu}, \hat{\sigma}^2\boldsymbol{I}_{64\times 64})$. When $t \to 0$, obviously $\alpha(\tilde{\mathbf{x}})(1-\alpha(\tilde{\mathbf{x}})) \to 0$, and the term $\frac{\alpha(\tilde{\mathbf{x}})(1-\alpha(\tilde{\mathbf{x}}))\|\boldsymbol{\mu}\|\sigma_t^4}{\sigma_t^4+\hat{\sigma}^2}$ vanishes. Hence $\lim_{t\to 0} V_{\mathrm{DSM}}(t) \approx \lim_{t\to 0}\frac{\hat{\sigma}^2}{\sigma_t^2(\sigma_t^2+\hat{\sigma}^2)} = \frac{\hat{\sigma}^2}{\sigma_m^2(\sigma_m^2+\hat{\sigma}^2)}$, which can not be neglected when σ_m is small. On the other hand, we can effectively view the 4069 discrete samples as a mixture of 4096 0-variance Gaussians, *i.e.*, $\hat{\sigma} = 0$. Thus by similar reasoning we could see $\lim_{t\to 0} V_{\mathrm{DSM}}(t) \approx \lim_{t\to 0}\frac{\hat{\sigma}^2}{\sigma_t^2(\sigma_t^2+\hat{\sigma}^2)} = 0$.

Training

We consider the CIFAR-10 and CelebA 64^2 in image generation tasks. Following [49], we first center-crop the CelebA images and then resize them to 64×64. For VE/VP, we use the same set of hyper-parameters and the NCSN++/DDPM++ backbones and the continuous-time training objectives for forward SDEs in [35]. For EDM, we adopt the improved hyper-parameters and architectures for NCSN++ in [27]. We set the reference batch size n to 4096 on CIFAR-10, 1024 on CelebA 64^2. The training iteration is 1.3M on CIFAR-10 and 1M on CelebA 64^2 for VE/VP, and 200M images for EDM [27]. The small batch size $\mathcal{B}$ in Algorithm 1 is the same as the batch size in the baseline score-based methods. For model selection, we pick the checkpoint with the lowest FID per 50k iterations on 10k samples for computing all the scores, as in [35] for VE/VP, and per 2.5M images on 50k samples as in [27] for EDM.

To measure the stability of converged VE models, we repeat the experiment 3 times on CIFAR-10 for DSM and STF objectives, using different random seeds.

We quantitatively study the training overhead of STF. All the numbers are measured on two NVIDIA A100 GPUs. In Table B.1 and Table B.2, we report the wall-clock training time (s) per 50 iterations/50k images on VE/EDM. We can see that the STF introduces additional overhead after incorporating the large reference batch. Since the calculation of the mini-batch target does not involve neural networks, the STF does not take significantly

longer training time. Indeed, in Section 3.5.3 we show that STF achieves comparable or better performance within a shorter training time.

Table B.1: Wall-clock training time (s) per 50 iterations on VE with NCSN++ [35]

Dataset-Method	CIFAR-10-DSM	CIFAR-10-STF	CelebA-DSM	CelebA-STF
Wall-clock time	13	16	24	26

Table B.2: Wall-clock training time (s) per 50k images on EDM with improved NCSN++ [27]

Dataset-Method	CIFAR-10 - DSM	CIFAR-10 - STF
Wall-clock time	98.5	101.5
Memory per GPU (G)	36.05	40.64

Sampling

We adopt the RK45 method for the backward ODE sampling of VE, and the DDIM sampler [54] for VP. For RK45 sampler of VE, we use the function implemented in `scipy.integrate.solve_ivp` with the tolerances `atol`$=1e-5/1e-4$, `rtol`$=1e-5/1e-4$ for CIFAR-10/CelebA 64^2. As in [35], we set the terminal time to $1e-5/1e-3$ for VE/VP. For EDM, we adopt Heun's 2^{nd} order method and the discretization scheme in [27], with 35 NFE.

We use the predictor-corrector (PC) sampler for reverse-time SDE. We follow [35] to set the Euler-Maruyama method as the predictor and the Langevin dynamics (MCMC) as the corrector.

Evaluations

For the evaluation, we compute the Fréchet distance between 50000 samples and the pre-computed statistics of CIFAR-10. For CelebA 64^2, we adopt the setting in [49] where the distance is computed between 10000 samples and the test set.

B.1.2 Extra Experiments

Stability of converged models

In Table B.3, we report the sample quality measured by FID/Inception score, and their standard deviations across random seeds on CIFAR-10. We can see that models trained with STF objective have lower variations of their final performances, in most cases. In particular, the standard deviation decreases from 4.41 to 0.06 for RK45 sampler on VE. It suggests that the STF objective can stabilize the performance of converged models.

Table B.3: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE), with standard deviation.

Methods	Inception ↑	FID ↓	NFE ↓
RK45 method (ODE)			
VE (DSM)	9.27 ± 0.12	8.90 ± 4.41	264 ± 52
VE (STF)	$\mathbf{9.52 \pm 0.06}$	$\mathbf{5.51 \pm 0.06}$	200 ± 8
PC sampler (SDE)			
VE (DSM)	9.68 ± 0.12	2.75 ± 0.13	2000
VE (STF)	$\mathbf{9.86 \pm 0.05}$	$\mathbf{2.66 \pm 0.13}$	2000

Effects of step size

In Figure B.1, we show the FID scores with the number of function evaluations of ODE samplers on CIFAR-10 and CelebA 64^2 . To vary the NFE, we adjust the error tolerance in the RK45 method. The sample quality of the STF objective degrades gracefully when decreasing the NFE. The STF objective consistently outperforms the DSM one for all NFEs on CIFAR-10, and largely improves over the baseline when setting the tolerance to $5e-3$ on CelebA 64^2 . It suggests that the STF has greater robustness to different step sizes.

B.1.3 Samples

We provide extended samples from score-based models trained by DSM/STF objective on CIFAR-10 and CelebA 64^2 by ODE samplers. For systematic comparison, we visualize

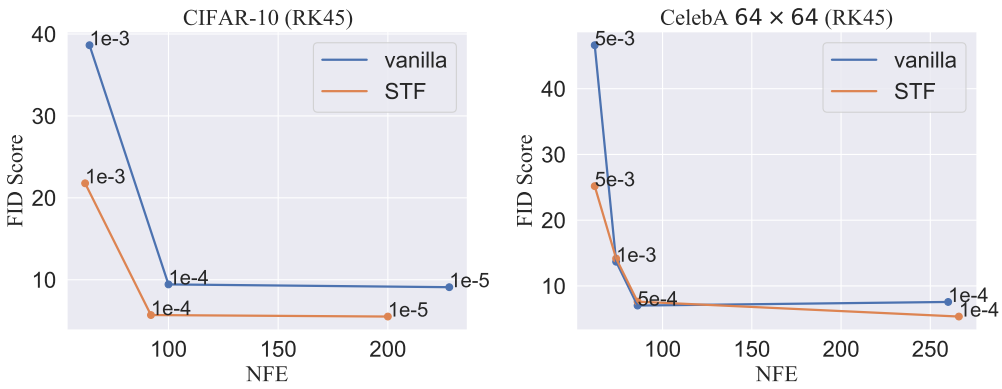


Figure B.1: FID versus NFE using ODE samplers on CIFAR-10 (left) and CelebA 64 × 64 (right)

samples from models trained on different seeds. We also provide samples generated by the state-of-the-art model — STF with EDM framework.

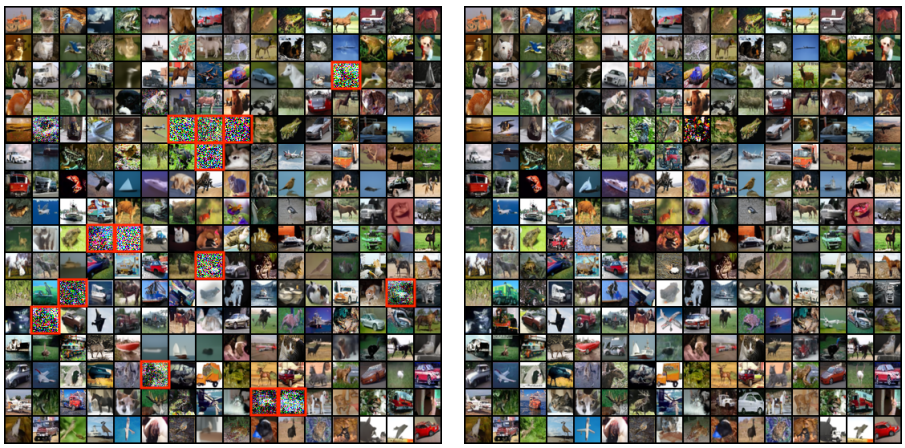
CIFAR-10

In Figure B.2, we visualize the samples produced by different methods across random seeds for VE. We use the RK45 sampler for sampling. We observe that the model trained by the DSM objective can produce noisy images (in red boxes), and the image quality has great variability across different random seeds. In contrast, models trained by STF objective generate clean and consistent samples with varying random seeds.

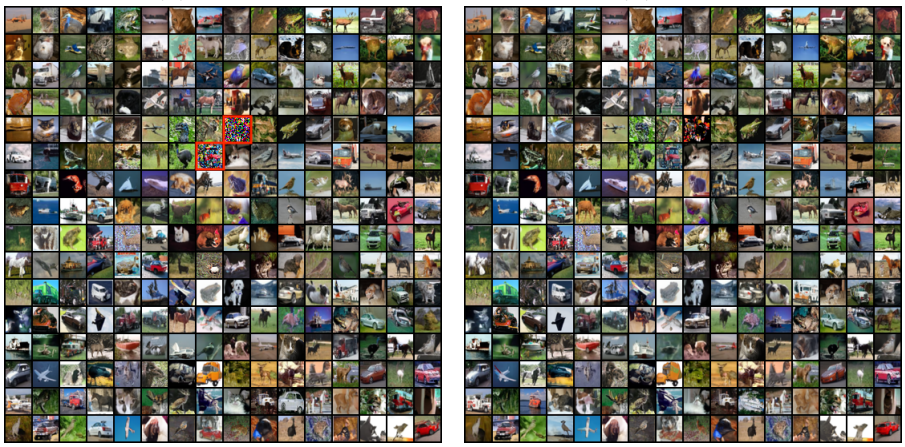
In Figure B.3, we further provide samples from a model trained by STF under the EDM framework [27]. The model is the current state-of-the-art on the unconditional CIFAR-10 generation task.

CelebA 64 × 64

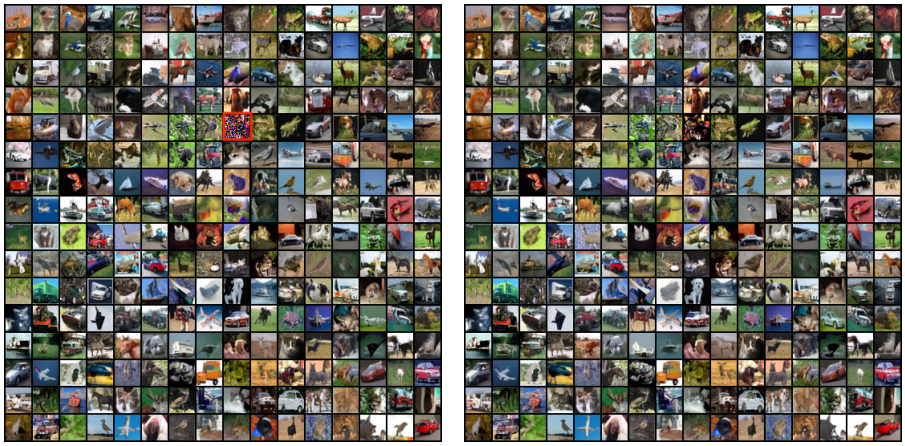
In Figure B.4, we provide samples from models trained on DSM and STF objectives with VE.



(a) DSM-1 (b) STF-1



(c) DSM-2 (d) STF-2



(e) DSM-3 (f) STF-3

Figure B.2: Samples generated by three different final models of DSM (left column) and STF (right column) on CIFAR-10. Red boxes indicate noisy images.

Figure B.3: CIFAR-10 samples from STF using EDM model. The FID is 1.90 and NFE is 35.

(a) DSM

(b) STF

Figure B.4: Samples generated by models trained on DSM (top) and STF (bottom) on CelebA 64^2 with VE.

B.2 Chapter 4

B.2.1 Algorithm Pseudocode

We provide algorithm pseudocode for the training in the first stage for denoiser and encoder (Alg 5) and the second stage for auto-regressive model (Alg 6) for clarity. We also include the pseudocode for sampling in Alg 7. Note that we generalize the equations in the main text by considering the conditional generation with condition c.

Algorithm 5 Mini-batch training of denoiser and encoder in DisCo-Diff

Input: Denoiser $\boldsymbol{D}_\theta$, encoder $\boldsymbol{E}_\phi$, training dataset D, batch size $\mathcal{B}$, Gumbel-Softmax temperature τ, training iteration T
for $i = 0, \ldots, T-1$ **do**
  Sample mini-batch data $\{(\mathbf{y}_i, c_i)\}_{i=1}^{\mathcal{B}}$ from D
  Sample time variables $\{t_i\}_{i=1}^{\mathcal{B}}$ from $p(t)$ and noise vectors $\{\mathbf{n}_i \sim \mathcal{N}(0, \sigma_i^2 \boldsymbol{I})\}_{i=1}^{\mathcal{B}}$
  Get perturbed data $\{\hat{\mathbf{y}}_i = \mathbf{y}_i + \mathbf{n}_i\}_{i=1}^{\mathcal{B}}$
  Sample the discrete latent from the encoder $\{\mathbf{z}_i \sim \boldsymbol{E}_\phi(\mathbf{y}_i)\}_{i=1}^{\mathcal{B}}$ using Gumbel-Softmax relaxation with temperature τ
  Calculate loss $\ell(\theta, \phi) = \sum_{i=1}^{\mathcal{B}} \lambda(t) \|\boldsymbol{D}_\theta(\hat{\mathbf{y}}_i, t_i, \mathbf{z}_i, c_i) - \mathbf{y}_i\|_2^2$
  Update the network parameter θ and ϕ via Adam optimizer
end for

Algorithm 6 Mini-batch training of auto-regressive model in DisCo-Diff

Input: Auto-regressive model $\boldsymbol{A}_\psi$, encoder $\boldsymbol{E}_\phi$, training dataset D, batch size $\mathcal{B}$, Gumbel-Softmax temperature τ, training iteration T
for $i = 0, \ldots, T-1$ **do**
  Sample mini-batch data $\{(\mathbf{y}_i, c_i)\}_{i=1}^{\mathcal{B}}$ from D
  Sample the discrete latent from the encoder $\{\mathbf{z}_i \sim \boldsymbol{E}_\phi(\mathbf{y}_i)\}_{i=1}^{\mathcal{B}}$ using Gumbel-Softmax relaxation with temperature τ
  Calculate loss $\ell(\psi) = \sum_{i=1}^{\mathcal{B}} \sum_j^m \log p_\psi((\mathbf{z}_i)_j | (\mathbf{z}_i)_{<j}, c_i)$
  Update the network parameter ψ via Adam optimizer
end for

B.2.2 Experimental Details

ImageNet Experiments

Architecture

Algorithm 7 Sampling procedure of DisCo-Diff

Input: Auto-regressive model $\boldsymbol{A}_\psi$, denoiser $\boldsymbol{D}_\phi$, time discretization $\{t_i\}_{i=0}^n$, condition c
`/* Sample discrete latent from auto-regressive model */`
for $i = 1, \ldots, m$ **do**
  Sample $\mathbf{z}_i \sim p_\psi(\mathbf{z}_i|\mathbf{z}_{<i}, c)$
end for
`/* Diffusion ODE (Heun's 2nd order method) */`
Sample $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, t_0^2\boldsymbol{I})$
for $i = 0, \ldots, n-1$ **do**
  $\mathbf{d}_i = (\mathbf{x}_i - \boldsymbol{D}_\theta(\mathbf{x}_i, t_i, \mathbf{z}, c))/t_i$
  $\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i)\mathbf{d}_i$
  if $t_{i+1} \neq 0$ **then**
    $\mathbf{d}'_i = (\mathbf{x}_{i+1} - \boldsymbol{D}_\theta(\mathbf{x}_{i+1}, t_{i+1}, \mathbf{z}, c))/t_{i+1}$
    $\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i)(\frac{1}{2}\mathbf{d}_i + \frac{1}{2}\mathbf{d}'_i)$
  end if
end for

	Feature maps	Attention resolution	Encoder patch size
ImageNet-64	1-2-3-4 ($\times$192)	(8,16,32)	8×8
ImageNet-128	1-2-4-16 ($\times$128)	(16,32,64)	16×16

Table B.4: Specific network configurations on ImageNet

For all the ImageNet experiments, we fix the latent dimension $m = 10$ in DisCo-Diff, and the codebook size for each discrete latent to 1000. Below we provide architecture details for the denoiser network, encoder, and auto-regressive model. Table B.4 also lists some key network configurations. Please see the source code in the Supplementary Material for all low-level details.

Denoiser Neural Network. (1) **ImageNet-64**: We use the same UNet architecture in EDM [27] for ImageNet-64, with newly injected cross-attention layers after each self-attention layers in each residual block. We feed the discrete vector into a six-layer Transformer encoder (with latent dimension 192) to obtain the corresponding embeddings for discrete variables. These embeddings are then input into the cross-attention layers. (2) **ImageNet-128**: We employ the UViT design in simple diffusion [106] and VDM++ [91]. UViT uses convolutional layers for down-/up-sampling, and a 36-layer ViT to process the lowest-resolution feature maps in the bottleneck, to strike a better balance between expressiveness and computation. Since the authors didn't release the code and model, we reimplemented the architecture by ourselves. We empirically observe that the convolutional blocks in EDM work better than the ones described in the simple diffusion paper, so we combine the up-/down-sampling blocks in EDM with the 36-layer ViT at the bottleneck layer. We further introduce a cross-attention layer for discrete latent in each up-/down-sampling block, and every three Transformer blocks in the ViT (*e.g.,* 12 new cross-attention layers in the ViT) to save computation.

Encoder. We utilize a 12-layer standard ViT [69] as the backbone for encoder. Its latent dimension is 768 and the number of attention heads is 12. The patch size for ImageNet-64 is 8×8 and for ImageNet-128 is 16×16. We treat each of the m discrete latents as a classification token, and concatenate their embeddings with the path embeddings.

Auto-regressive model. We use a standard Transformer decoder [109], with a depth of 12, a number of heads of 8, and a latent dimension of 512. The inference time of the auto-regressive model is much smaller than the iterative denoising process, given that the discrete latent only has 10 dimensions. To generate 32 images on ImageNet-128, the auto-regressive model takes 0.44 seconds, while the diffusion model takes 78 seconds for 114 NFE, with an

average of 0.68 s/NFE on a single NVIDIA A100 GPU.

Training and Sampling

We borrow the preconditioning techniques, training noisy schedule, optimizers, exponential moving average (EMA) schedule, and hyper-parameters from previous state-of-the-art diffusion model EDM [27] on ImageNet-64. We employ the shifted EDM-monotonic noisy schedule proposed in VDM++ [91] on ImageNet-128, and keep other training details the same in ImageNet-64. We use the Gumbel-Softmax [83] as the continuous relaxation for the discrete latents. The temperature τ in Gumbel-Softmax controls the smoothness of the categorical distribution. When $\tau \to 0$, the expected value of the Gumbel-Softmax is the same as the one of the underlying predicted distribution. As we increase t, the Gumbel-Softmax would gradually converge to a uniform distribution. Hence, a relatively large τ effectively provides regualization effects. During training, we set the τ to a constant 1. However, for the extraction of latents from training images, which aids in constructing the dataset for the second stage of the auto-regressive model, we adjust τ to a lower value of 0.01.

We use Heun's second-order ODE solver as the default ODE sampler, which is proven effective in previous works [27], [82]. We directly use the hyper-parameters for the 623 NFE setting in Restart sampler [29] on ImageNet-64, for DisCo-Diff.

Evaluation

For the evaluation, we follow the standard protocol and compute the Fréchet distance between 50000 generated samples and the training images.

Molecular Docking

In this appendix, we will introduce the task of molecular docking and some of the existing approaches to tackle it for readers who are not familiar with this field. Experienced readers may skip to Section B.2.2 where we start describing the details of our docking approach.

Task Overview

Molecular docking consists of finding the 3D structure that a protein (also referred to as receptor) and a small molecule (or ligand) take when binding. This is an important task in drug design because most drugs are small molecules that operate by binding to a specific protein of interest in our body and inhibiting or enhancing its function. The common

particular instantiation of the docking problem that we consider is also referred to as rigid blind docking i.e. where we are given as input the correct protein structure (*rigid*) but are not provided any information about where the ligand will bind on the protein nor the conformations it will take (*blind*).

Ground truth data for training and testing is obtained through experimental methods, like X-ray crystallography, that, for each protein-ligand complex, allow to observe a particular pose that protein and ligand took when binding together inside of the crystal. Although there may be other poses that this particular protein and ligand may take when binding in a natural environment, methods are evaluated based on their capacity to retrieve the crystal pose. This accuracy is typically computed as the percentage of test complexes where the predicted structure of the ligand (also referred to as ligand pose) is within a root mean square distance (RMSD) of 2**r**A from the ground truth when aligning the protein structures.

Related work

Traditional approaches tackled the task via a search-based paradigm where, given a scoring function, they would search over possible ligand poses with an optimization algorithm to find a global minimum [121], [251]. Recently, deep learning methods have been trying to speed up this search process by generating poses directly through a neural network. Initial approaches [117], [122] used regression-based frameworks to predict the pose, but, although significantly faster, they did not outperform traditional methods.

[107] argued that the issue with these regression-based approaches is their treatment of the uncertainty in the multimodal model posterior pose distribution. They also proposed DiffDock, a diffusion-based generative model to generate docked poses that was able to outperform previous methods, which we use as a starting point for the integration of our DisCo-Diff approach to diffusion.

Most deep learning approaches to docking model the data as a geometric graph or point cloud in 3D. The nodes of this graph are the (heavy) atoms of the ligand and, typically, the C-alpha carbon atoms of the protein backbone (sometimes full-atom representations are also used for the protein but these are less common for computational complexity reasons). These nodes are connected by edges in case of chemical bonds or pairwise distances below a certain cutoff. Neural architectures learn features over the nodes of this graph through a number of

message passing layers, the geometric structure is encoded via invariant (e.g. relying only on distance embeddings, see [252]) or equivariant operations [108].

Latent variables

We design each latent variable to take values indicating one of the nodes in the protein-ligand joint graph. Therefore the codebook size for the latent variable of any given protein-ligand complex is equal to the total number of nodes in the graph i.e. the sum of the number of atoms in the ligand and the number of residues in the protein. With this choice, intuitively each latent variable will indicate one particular atom or residue involved in some key component of the protein-ligand interaction. For example, using two latents the model can learn to indicate a geometric contact between a pair of nodes in the final representation. Further note, each "codebook", when considered as a set of one-hot vectors indicating notes, has a permutation equivariance property with the nodes of the graph (because they are associated with node properties): if the nodes of the input graph are permuted each latent variable coming out of the encoder or autoregressive model will also have its codebook representation permuted.

Architecture details

Denoiser. This design choice for the discrete latents codebooks fits very well with the preexisting DiffDock's denoiser architecture composed of equivariant message passing layers [108]. Each latent variable is encoded in a binary label for each node which is set to zero for all nodes except the one indicated by the latent. These binary labels are concatenated to the initial node features while the rest of the denoiser is kept unchanged. With probability 0.1 during training we drop the latents, in this case, the binary labels are set to zero for all latents, and a learnable null-embedding is fed to all initial node features.

Encoder. The encoder and autoregressive models adopt very similar architectures to the denoiser with a few key distinctions. The encoder takes as input the ground truth pose of the ligand, learns features for each node through message passing, and finally m separate feedforward MLPs (where m is the number of latents) with a one-dimensional output are applied to each node representation. The concatenated outputs of each of these MLPs form the logit vectors for each of the latent variables which are passed through the Gumbel-Softmax discretization step.

Autoregressive model. Unlike the image synthesis experiments setting where the images are often generated with relatively vague conditioning information, for docking, we are interested in generating ligand poses conditioned on a particular protein (structure) and ligand. This conditioning information significantly influences the posterior pose distribution and consequently the learned latent variables. Therefore, we need to condition the autoregressive model on the protein structure and the ligand. We achieve this, once again, through an equivariant message passing network, operating on an input composed of the protein structure and the ligand. The latter is centered at the protein's center, given an arbitrary conformer (i.e. molecular conformation) from RDKit [253] and a uniformly random orientation. Like the denoiser, the autoregressive model takes as input the additional binary node labels for the existing latents (masked out appropriately during training), and, like the encoder, it uses its final node embeddings to predict the logits for the next latent variable.

Experimental details

For the docking experiments, we follow the datasets and procedures established by [117] and [107]. Data for training and evaluation comes from the PDBBind dataset [118] with time-based splits (complexes before 2019 for training and validation, selected complexes from 2019 for testing).

Denoiser. We use a denoiser architecture analogous to the one proposed by [116], which is a smaller version of DiffDock's original architecture where the main changes are: (1) 5 convolutional layers (vs 6 of the original DiffDock's architecture) (2) node representations with 24 scalars and pseudoscalars and 6 vectors and pseudovectors (vs respectively 48 and 10) (3) spherical harmonics order limited to 1 (vs 2). These changes, although somewhat affecting the inference quality, make training and testing of the models significantly more affordable (from 18 days on 4 GPUs to 9 days on 2 GPUs for training).

We keep the same denoiser architecture for both the baseline without discrete latents (DiffDock-S) and our model and apply similar hyperparameter searches when applicable to both models. At inference time, similarly to [107], we take 40 independent samples and use the original DiffDock's confidence model[1] to select the top one. For DisCo-DiffDock each of

[1]The confidence model is an additional model, [107] trained to select the most likely correct poses out of the diffusion models samples. The reader can think of this as trying to select the maximum likelihood pose.

the samples is taken by independently sampling from the autoregressive model and then the (conditioned) denoiser.

Encoder. For the encoder, we use a similar but slightly smaller architecture with 3 convolutional layers, 24 scalars, and 4 vectors. We set the number of discrete latent variables (each taking values over the whole set of possible nodes in the joint graph) to two, as we found this to equilibrate the complexity of the generative task between the score and autoregressive models.

Autoregressive model. One challenge with the autoregressive model in this domain is its tendency to overfit the latent variables in the training set given the limited training data, the complexity of the conditioning information, and the low training signal that discrete labels provide. Therefore we found it beneficial to design the autoregressive model to use the pretrained layers of the denoiser itself. In particular, we simply add independent MLPs for each latent variable that are applied to the final scalar representations of the nodes. During the autoregressive training, for the first five epochs, the weights of the convolutional layers are frozen.

Inference hyperparameters. For inference, we maintain the number of inference steps from DiffDock (20) and, for both DisCo-DiffDock and the baseline, we tune on the validation set the sampling temperature for the different components of the diffusion similarly to how it was done by [254]. For DisCo-DiffDock we also tune the temperature used to sample the autoregressive model. We find, with 40 samples, to be beneficial to set this temperature > 1 while the diffusion sampling temperature < 1, this corresponds to encouraging exploration of different binding modes while trying to obtain the maximum likelihood pose for each mode. This further highlights the advantage provided by enabling the decomposition of different degrees of uncertainty. Please see the source code in the Supplementary Material for all low-level details and hyperparameters used.

Gaussian Mixture Experiments

Data generation For the toy example in section 4.2, we set the true data distribution to a mixture of eight Gaussian components:

$$p_{data}(\mathbf{x}) = \frac{1}{8}\sum_{i=1}^{8}\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i \boldsymbol{I}_{2\times 2})$$

where $\forall i, \sigma_i = 0.2$, and

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} -3 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \quad \boldsymbol{\mu}_4 = \begin{pmatrix} 0 \\ -3 \end{pmatrix},$$
$$\boldsymbol{\mu}_5 = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ \frac{3}{\sqrt{2}} \end{pmatrix}, \quad \boldsymbol{\mu}_6 = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ \frac{-3}{\sqrt{2}} \end{pmatrix}, \quad \boldsymbol{\mu}_7 = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ \frac{-3}{\sqrt{2}} \end{pmatrix}, \quad \boldsymbol{\mu}_8 = \begin{pmatrix} \frac{-3}{\sqrt{2}} \\ \frac{-3}{\sqrt{2}} \end{pmatrix}.$$

To construct the toy dataset, we randomly sampled 1000 data points from each component, totaling 8000 data points. We visualize the KDE plot of the generated data in Fig. 4.3(a).

Training and Sampling

We employ a four-layer MLP as the diffusion decoder (Denoiser Neural Network) $\mathcal{G}$, for both Disco-Diff and diffusion models. We use a three-layer MLP as the encoder $\mathcal{E}$ in Disco-Diff. We set the latent dimension of discrete latent to 1 and the vocabulary size to 8. Ideally, each discrete latent should correspond to a Gaussian component, and the time-dependent scores for a single Gaussian component have a simple analytical expression. We leverage this simplicity and reparameterize the output of diffusion decoder as $\mathcal{G}(\mathbf{x}, t, z) = \frac{\mathcal{F}(z)-\mathbf{x}}{t^2+\sigma_1^2} + \mathcal{H}(\mathbf{x}, t)$, where $\mathcal{F}$ is the embedding for each discrete latent z and $\mathcal{H}$ is a four-layer MLP. The model optimization uses the Adam optimizer with a learning rate of 1e-3.

For sampling, we use the Heun's second-order sampler. We followed the time discretization scheme in EDM [27] with 50 sampling steps.

Metric

We detail the metrics used in Fig. 4.4. The curvature for points $\mathbf{x}(t)$ on ODE trajectory $d\mathbf{x}/dt = \mathcal{G}(\mathbf{x}, t, z)$ (z is null in diffusion models) is defined as $\kappa(\mathbf{x}(t)) = \frac{||\partial_t \boldsymbol{T}(\mathbf{x}(t),t)||}{||\mathbf{x}'(t)||}$ where $\boldsymbol{T}(\mathbf{x}, t) = \frac{\mathcal{G}(\mathbf{x}(t),t,z)}{||\mathcal{G}(\mathbf{x}(t),t,z)||}$ is the unit tangent vector. We can approximate the curvature by finite

difference: $\kappa(\mathbf{x}(t)) = \frac{||\partial_t \boldsymbol{T}(t)||}{||\mathbf{x}'(t)||} \approx \frac{||\boldsymbol{T}(t) - \boldsymbol{T}(t-\Delta t)||}{||\mathbf{x}(t) - \mathbf{x}(t-\Delta t)||}$. We approximate $\mathbf{x}(t - \Delta t)$ by a single Euler step, *i.e.,* $\mathbf{x}(t - \Delta t) = \mathbf{x}(t) - \mathcal{G}(\mathbf{x}, t, z)\Delta t$. In Fig. 4.4(a), we report the expected curvature given the backward time when simulating the ODE, *i.e.,* $\mathbb{E}_{\mathbf{x}(t)}\left[\frac{||\boldsymbol{T}(t) - \boldsymbol{T}(t-\Delta t)||}{||\mathbf{x}(t) - \mathbf{x}(t-\Delta t)||}\right]$. We set the time elapsed to $\Delta t = 0.001$.

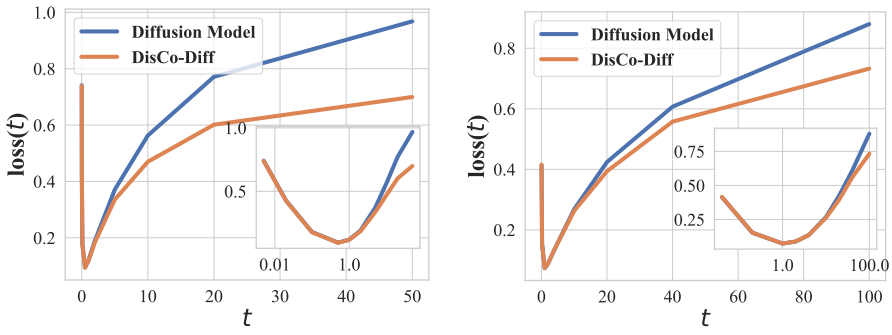
In Fig. 4.4(b), we measure the complexity of the trained neural networks using the expected squared Frobenius norm of the network's Jacobians, *i.e.,* $\mathbb{E}_{\mathbf{x}(t)}\left[||\nabla_{\mathbf{x}} \mathcal{G}(\mathbf{x}, t, z)||_F^2\right]$.

Additionally, to quantitatively evaluate the generation quality, we report the Wasserstein-2 (W-2) distance between the generated distribution and the ground truth distribution. In the DisCo-Diff model, the W-2 distance is at 0.118, compared to 0.27 in the standard diffusion model. It suggests that DisCo-Diff better captures the multimodal distribution, even in 2-dim space.

B.2.3 Extra Experiments

Loss Analysis

In Fig. B.5, we provide the loss versus time curve on both ImageNet-64 and ImageNet-128 datasets. We have also included a log-scale version of the x-axis in the inset plot.



(a) ImageNet-64 (b) ImageNet-128

Figure B.5: Averaged training loss versus noise level t.

B.2.4 Samples

Class-conditoned ImageNet-64

We provide extended samples generated by DisCo-Diff in Fig. B.6.

Class-conditoned ImageNet-128

We provide extended samples generated by DisCo-Diff in Fig. B.7. We also visualize samples with shared discrete latent in Fig. B.8.

Group Hierarchical DisCo-Diff

We further provide extended samples from the Group hierarchical DisCo-Diff. Fig. B.9 showcases the generated images when composing two discrete latents together, *i.e.,* ($\mathbf{z}_{0:20}$, $\hat{\mathbf{z}}_{20:30}$). We can see that the generated images from composed latent generally inherit the shape from images generated by $\mathbf{z}$, and the color from images generated by $\hat{\mathbf{z}}$.

Fig. B.10 further shows the effect when progressively fixing more coordinates of the discrete latent, and sampling the remaining coordinates by the auto-regressive model. The images first converge in shape/layout, and subsequently converge in color/texture.

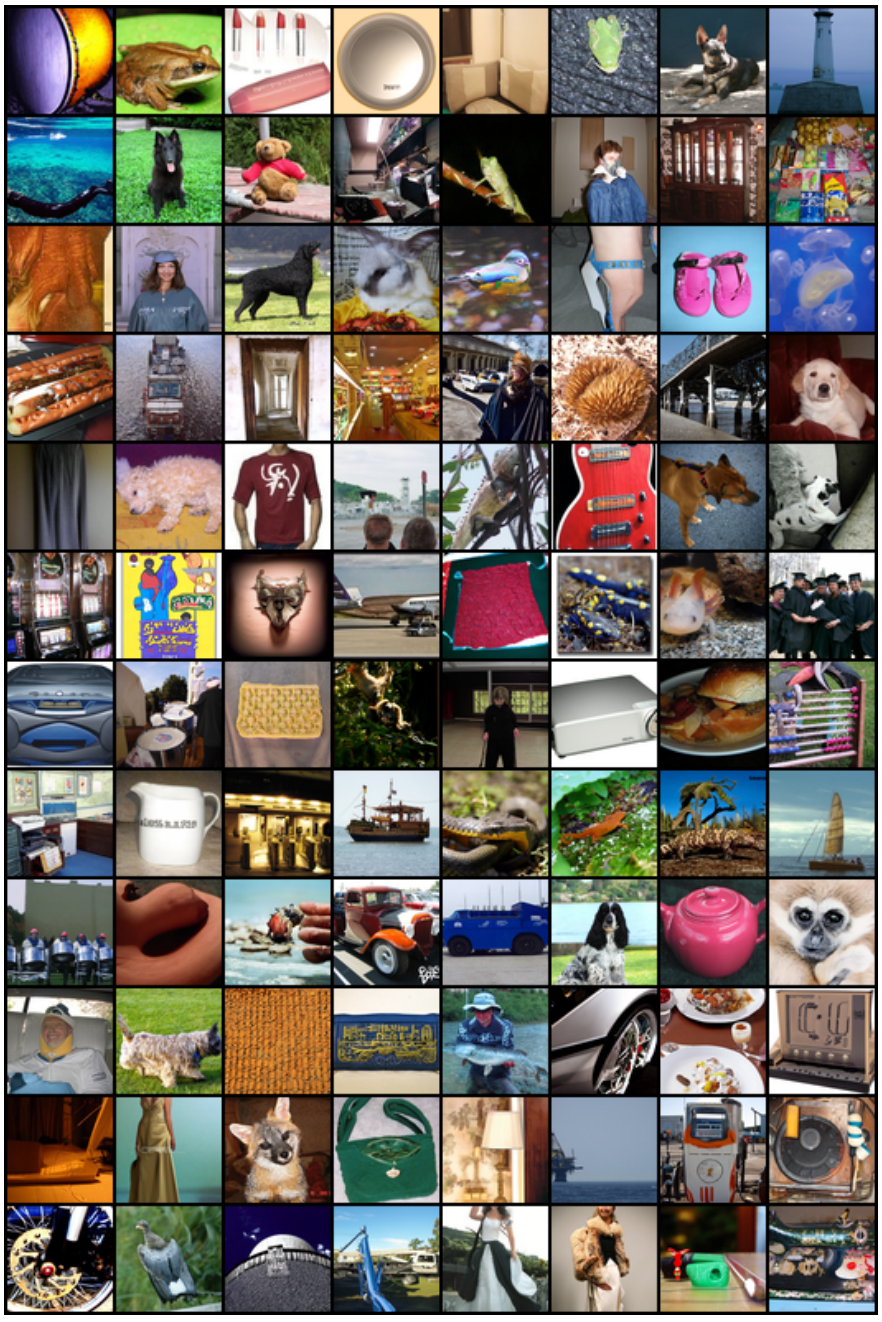


Figure B.6: Generated samples by DisCo-Diff on class-conditioned ImageNet-64, with ODE sampler (FID=1.65, NFE=78).

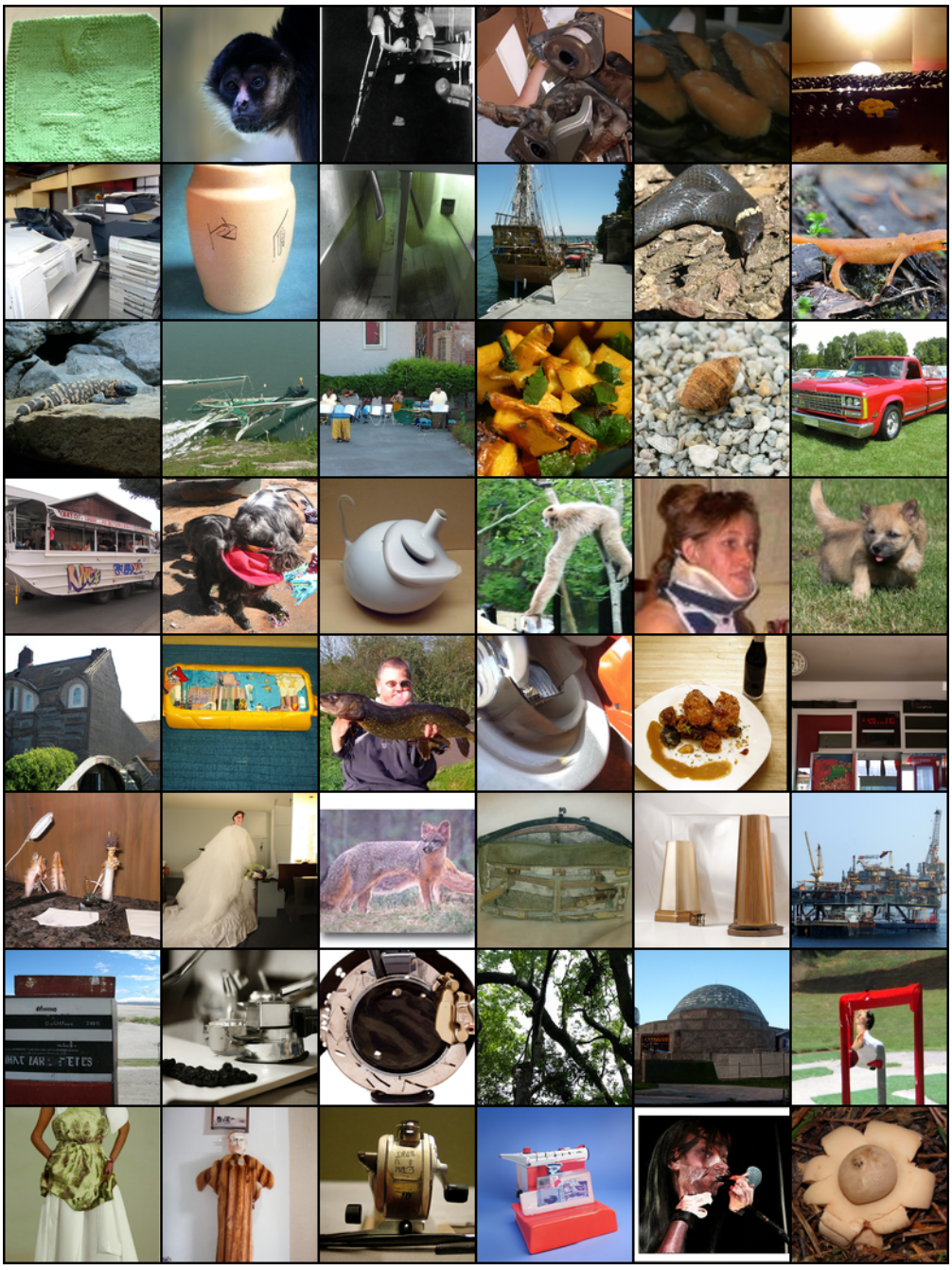


Figure B.7: Generated samples by DisCo-Diff on class-conditioned ImageNet-128, with ODE sampler (FID=2.08, NFE=114).

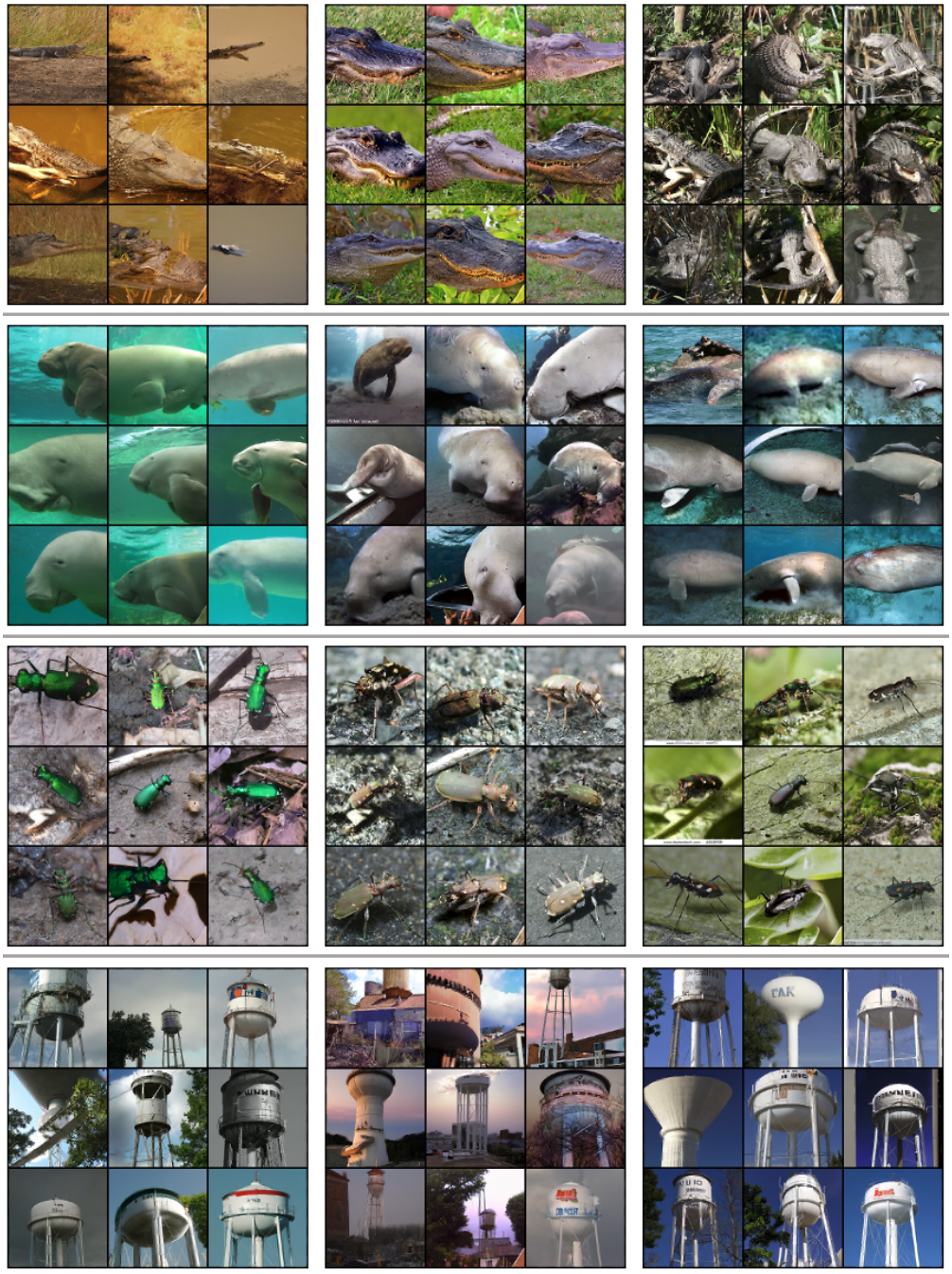


Figure B.8: Generated samples by DisCo-Diff on class-conditioned ImageNet-128, with ODE sampler. Samples in each grid share the same latent, and grids in each row share the same class labels. We can see that generally, images sharing the same discrete latents demonstrate similar global characteristics, such as shape, layout, and color, despite being under the same class. It suggests that discrete latents provide complementary information to the class labels.

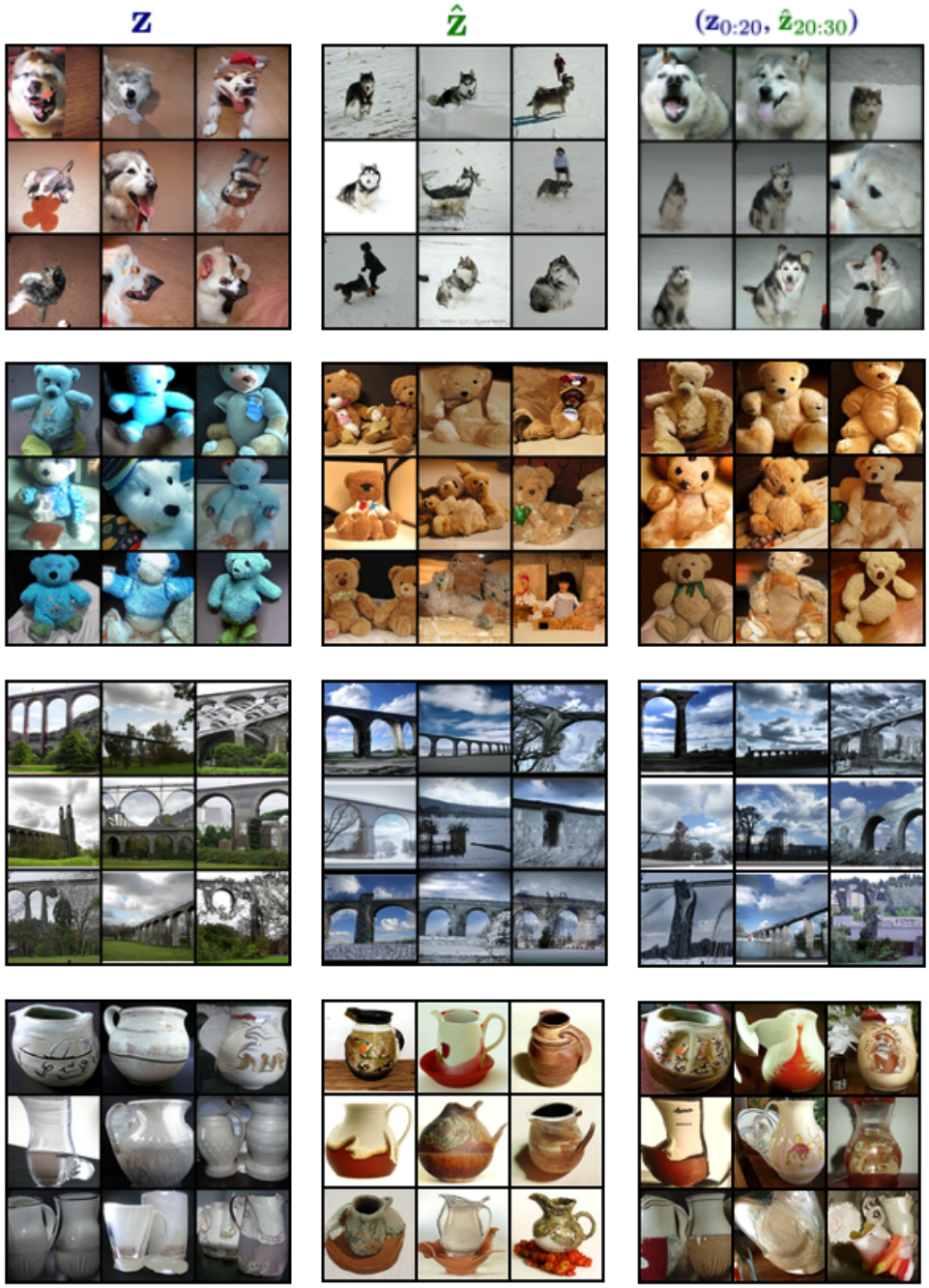


Figure B.9: Generated images with a shared latent, using group hierarchical DisCo-Diff trained on ImageNet-64. *Left:* Shared latent $\mathbf{z}$. *Middle:* Shared latent $\hat{\mathbf{z}}$. *Right:* Shared latent $(\mathbf{z}_{0:20}, \hat{\mathbf{z}}_{20:30})$, where the first 20 coordinates are from $\mathbf{z}$ and the last 10 coordinates are from $\hat{\mathbf{z}}$. We can see that the generated images from composed latents generally inherit the shape from images generated by $\mathbf{z}$, and the color from images generated by $\hat{\mathbf{z}}$.

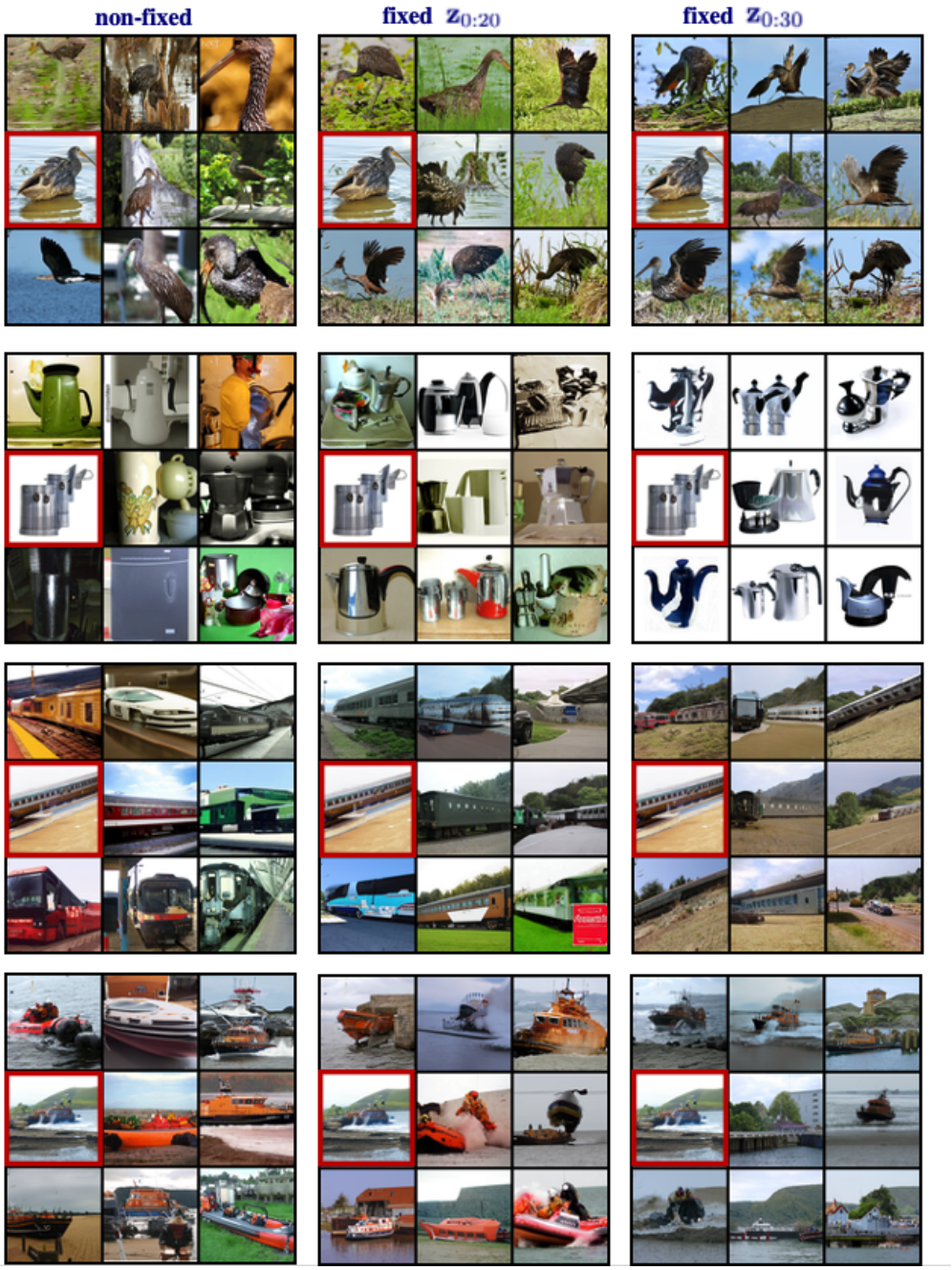


Figure B.10: Progressively fixing more subcoordinates of the discrete latents, using our group hierarchical DisCo-Diff on ImageNet-64. *Left:* Randomly sampled $\mathbf{z}$. *Middle:* Fixing the first 20 coordinates $\mathbf{z}_{:20}$ as the one derived from the red-boxed image, sampling the rest. *Right:* Fixing the whole 30-dim. $\mathbf{z}$ as the one derived from the red-boxed image. The figure shows the effect when progressively fixing more coordinates of the discrete latent, and sampling the remaining coordinates by the auto-regressive model. The images first converge in shape/layout, and subsequently converge in color/texture.

B.3 Chapter 5

B.3.1 Algorithm Pseudocode

EDM Discretization Scheme

[27] proposes a discretization scheme for ODE given the starting $t_{\max}$ and end time $t_{\min}$. Denote the number of steps as N, then the EDM discretization scheme is:

$$t_{i<N} = \left(t_{\max}^{\frac{1}{\rho}} + \frac{i}{N-1}(t_{\min}^{\frac{1}{\rho}} - t_{\max}^{\frac{1}{\rho}})\right)^{\rho}$$

with $t_0 = t_{\max}$ and $t_{N-1} = t_{\min}$. ρ is a hyperparameter that determines the extent to which steps near $t_{\min}$ are shortened. We adopt the value $\rho = 7$ suggested by [27] in all of our experiments. We apply the EDM scheme to creates a time discretization in each Restart interval $[t_{\max}, t_{\min}]$ in the Restart backward process, as well as the main backward process between $[0, T]$ (by additionally setting $t_{\min} = 0.002$ and $t_N = 0$ as in [27]). It is important to note that $t_{\min}$ should be included within the list of time steps in the main backward process to seamlessly incorporate the Restart interval into the main backward process. We summarize the scheme as a function in Algorithm 8.

Algorithm 8 EDM_Scheme($t_{\min}, t_{\max}, N, \rho = 7$)

1: **return** $\left\{(t_{\max}^{\frac{1}{\rho}} + \frac{i}{N-1}(t_{\min}^{\frac{1}{\rho}} - t_{\max}^{\frac{1}{\rho}}))^{\rho}\right\}_{i=0}^{N-1}$

Restart Algorithm

We present the pseudocode for the Restart algorithm in Algorithm 9. In this pseudocode, we describe a more general case that applies l-level Restarting strategy. For each Restart segment, the include the number of steps in the Restart backward process N_{Restart}, the Restart interval $[t_{\min}, t_{\max}]$ and the number of Restart iteration K. We further denote the number of steps in the main backward process as N_{main}. We use the EDM discretization scheme (Algorithm 8) to construct time steps for the main backward process ($t_0 = T, t_{N_{\text{main}}} = 0$) as well as the

Restart backward process, when given the starting/end time and the number of steps.

Although Heun's 2nd order method [154] (Algorithm 10) is the default ODE solver in the pseudocode, it can be substituted with other ODE solvers, such as Euler's method or the DPM solver [26].

The provided pseudocode in Algorithm 9 is tailored specifically for diffusion models [27]. To adapt Restart for other generative models like PFGM++ [31], we only need to modify the Gaussian perturbation kernel in the Restart forward process (line 10 in Algorithm 9) to the one used in PFGM++.

Algorithm 9 Restart sampling

1: **Input:** Score network s_θ, time steps in main backward process $t_{i\in\{0,N_{\text{main}}\}}$, Restart parameters $\{(N_{\text{Restart},j}, K_j, t_{\text{min},j}, t_{\text{max},j})\}_{j=1}^{l}$
2: Round $t_{\text{min},j\in\{1,l\}}$ to its nearest neighbor in $t_{i\in\{0,N_{\text{main}}\}}$
3: Sample $x_0 \sim \mathcal{N}(\mathbf{0}, T^2\boldsymbol{I})$
4: **for** $i = 0 \dots N_{\text{main}} - 1$ **do**
5: $\quad x_{t_{i+1}} = \text{OneStep\_Heun}(s_\theta, t_i, t_{i+1})$
6: $\quad$ **if** $\exists j \in \{1, \dots, l\}, t_{i+1} = t_{\text{min},j}$ **then**
7: $\quad\quad t_{\text{min}} = t_{\text{min},j}, t_{\text{max}} = t_{\text{max},j}, K = K_j, N_{\text{Restart}} = N_{\text{Restart},j}$
8: $\quad\quad x^0_{t_{\text{min}}} = x_{t_{i+1}}$
9: $\quad\quad$ **for** $k = 0 \dots K - 1$ **do**
10: $\quad\quad\quad \varepsilon_{t_{\text{min}}\to t_{\text{max}}} \sim \mathcal{N}(\mathbf{0}, (t^2_{\text{max}} - t^2_{\text{min}})\boldsymbol{I})$
11: $\quad\quad\quad x^{k+1}_{t_{\text{max}}} = x^k_{t_{\text{min}}} + \varepsilon_{t_{\text{min}}\to t_{\text{max}}}$
12: $\quad\quad\quad \{\bar{t}_m\}_{m=0}^{N_{\text{Restart}}-1} = \text{EDM\_Scheme}(t_{\text{min}}, t_{\text{max}}, N_{\text{Restart}})$
13: $\quad\quad\quad$ **for** $m = 0 \dots N_{\text{Restart}} - 1$ **do**
14: $\quad\quad\quad\quad x^{k+1}_{\bar{t}_{m+1}} = \text{OneStep\_Heun}(s_\theta, \bar{t}_m, \bar{t}_{m+1})$
15: $\quad\quad\quad$ **end for**
16: $\quad\quad$ **end for**
17: $\quad$ **end if**
18: **end for**
19: **return** $x_{t_{N_{\text{main}}}}$

Algorithm 10 OneStep_Heun$(s_\theta, x_{t_i}, t_i, t_{i+1})$

1: $d_i = t_i s_\theta(x_{t_i}, t_i)$
2: $x_{t_{i+1}} = x_{t_i} - (t_{i+1} - t_i)d_i$
3: **if** $t_{i+1} \neq 0$ **then**
4: $\quad d'_i = t_{i+1} s_\theta(x_{t_{i+1}}, t_{i+1})$
5: $\quad x_{t_{i+1}} = x_{t_i} - (t_{i+1} - t_i)(\frac{1}{2}d_i + \frac{1}{2}d'_i)$
6: **end if**
7: **return** $x_{t_{i+1}}$

B.3.2 Experimental Details

In this section, we discuss the configurations for different samplers in details. All the experiments are conducted on eight NVIDIA A100 GPUs.

Configurations for Baselines

We select **Vanilla SDE** [37], **Improved SDE** [27], **Gonna Go Fast** [28] as SDE baselines and the **Heun**'s 2$^{\text{nd}}$ order method [154] (Alg 10) as ODE baseline on standard benchmarks CIFAR-10 and ImageNet 64×64. We choose **DDIM** [25], **Heun**'s 2$^{\text{nd}}$ order method, and **DDPM** [167] for comparison on Stable Diffusion model.

Vanilla SDE denotes the reverse-time SDE sampler in [37]. For Improved SDE, we use the recommended dataset-specific hyperparameters ($e.g. S_{\text{max}}, S_{\text{min}}, S_{\text{churn}}$) in Table 5 of the EDM paper [27]. They obtained these hyperparameters by grid search. Gonna Go Fast [28] applied an adaptive step size technique based on Vanilla SDE and we directly report the FID scores listed in [28] for Gonna Go Fast on CIFAR-10 (VP). For fair comparison, we use the EDM discretization scheme [27] for Vanilla SDE, Improved SDE, Heun as well as Restart.

We borrow the hyperparameters such as discretization scheme or initial noise scale on Stable Diffusion models in the diffuser [2] code repository. We directly use the DDIM and DDPM samplers implemented in the repo. We apply the same set of hyperparameters to Heun and Restart.

Configurations for Restart

We report the configurations for Restart for different models and NFE on standard benchmarks CIFAR-10 and ImageNet 64×64. The hyperparameters of Restart include the number of steps in the main backward process N_{main}, the number of steps in the Restart backward process N_{Restart}, the Restart interval $[t_{\text{min}}, t_{\text{max}}]$ and the number of Restart iteration K. In Table B.6 (CIFAR-10, VP) we provide the quintuplet $(N_{\text{main}}, N_{\text{Restart}}, t_{\text{min}}, t_{\text{max}}, K)$ for each experiment. Since we apply the multi-level Restart strategy for ImageNet 64×64, we provide N_{main} as well as a list of quadruple $\{(N_{\text{Restart},i}, K_i, t_{\text{min},i}, t_{\text{max},i})\}_{i=1}^{l}$ (l is the

[2] https://github.com/huggingface/diffusers

number of Restart interval depending on experiments) in Table B.8. In order to integrate the Restart time interval to the main backward process, we round $t_{\min,i}$ to its nearest neighbor in the time steps of main backward process, as shown in line 2 of Algorithm 9. We apply Heun method for both main/backward process. The formula for NFE calculation is NFE $= \underbrace{2 \cdot N_{\text{main}} - 1}_{\text{main backward process}} + \sum_{i=1}^{l} \underbrace{K_i}_{\text{number of repetitions}} \cdot \underbrace{(2 \cdot (N_{\text{Restart},i} - 1))}_{\text{per iteration in } i^{\text{th}} \text{ Restart interval}}$ in this case. Inspired by [27], we inflate the additive noise in the Restart forward process by multiplying $S_{\text{noise}} = 1.003$ on ImageNet 64×64, to counteract the over-denoising tendency of neural networks. We also observe that setting $\gamma = 0.05$ in Algorithm 2 of EDM [27] would sligtly boost the Restart performance on ImageNet 64×64 when $t \in [0.01, 1]$.

We further include the configurations for Restart on Stable Diffusion models in Table B.13, with a varying guidance weight w. Similar to ImageNet 64×64, we use multi-level Restart with a fixed number of steps $N_{\text{main}} = 30$ in the main backward process. We utilize the Euler method for the main backward process and the Heun method for the Restart backward process, as our empirical observations indicate that the Heun method doesn't yield significant improvements over the Euler method, yet necessitates double the steps. The number of steps equals to $N_{\text{main}} + \sum_{i=1}^{l} K_i \cdot (2 \cdot (N_{\text{Restart},i} - 1))$ in this case. We set the total number of steps to 66, including main backward process and Restart backward process.

Given the prohibitively large search space for each Restart quadruple, a comprehensive enumeration of all possibilities is impractical due to computational limitations. Instead, we adjust the configuration manually, guided by the heuristic that weaker/smaller models or more challenging tasks necessitate a stronger Restart strength (e.g., larger K, wider Restart interval, etc). On average, we select the best configuration from 5 sets for each experiment; these few trials have empirically outperformed previous SDE/ODE samplers. We believe that developing a systematic approach for determining Restart configurations could be of significant value in the future.

Pre-trained Models

For CIFAR-10 dataset, we use the pre-trained VP and EDM models from the EDM repository [3], and PFGM++ ($D = 2048$) model from the PFGM++ repository [4]. For ImageNet 64×64, we borrow the pre-trained EDM model from EDM repository as well.

Classifier-free Guidance

We follow the convention in [160], where each step in classifier-free guidance is as follows:

$$\tilde{s}_\theta(x, c, t) = w s_\theta(x, c, t) + (1 - w) s_\theta(x, t)$$

where c is the conditions, and $s_\theta(x, c, t)/s_\theta(x, t)$ is the conditional/unconditional models, sharing parameters. Increasing w would strength the effect of guidance, usually leading to a better text-image alignment [160].

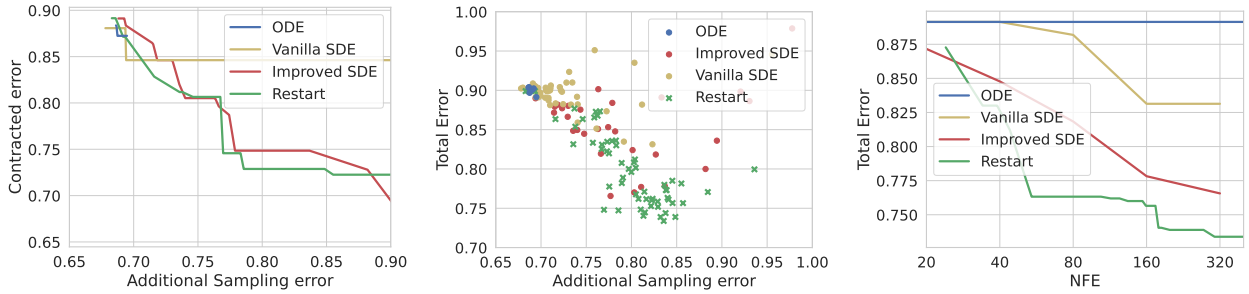
More on the Synthetic Experiment

Discrete Dataset

We generate the underlying discrete dataset S with $|S| = 2000$ as follows. Firstly, we sample 2000 points, denoted as S_1, from a mixture of two Gaussians in $\mathbb{R}^4$. Next, we project these points onto $\mathbb{R}^{20}$. To ensure a variance of 1 on each dimension, we scale the coordinates accordingly. This setup aims to simulate data points that primarily reside on a lower-dimensional manifold with multiple modes.

The specific details are as follows: $S_1 \sim 0.3N(a, s^2 I) + 0.7(-a, s^2 I)$, where $a = (3, 3, 3, 3) \subset \mathbb{R}^4$ and $s = 1$. Then, we randomly select a projection matrix $P \in \mathbb{R}^{20 \times 4}$, where each entry is drawn from $N(0, 1)$, and compute $S_2 = PS_1$. Finally, we scale each coordinate by a constant factor to ensure a variance of 1.

[3] https://github.com/NVlabs/edm
[4] https://github.com/Newbeeer/pfgmpp



(a) (b) (c)

Figure B.11: Comparison of additional sampling error versus **(a)** contracted error (plotting the Pareto frontier) and **(b)** total error (using a scatter plot). **(c)** Pareto frontier of NFE versus total error.

Model Architecture

We employ a common MLP architecture with a latent size of 64 to learn the score function. The training method is adapted from [27], which includes the preconditioning technique and denoising score-matching objective [38].

Varying Hyperparameters

To achieve the best trade-off between contracted error and additional sampling error, and optimize the NFE versus FID (Fréchet Inception Distance) performance, we explore various hyperparameters. [27] shows that the Vanilla SDE can be endowed with additional flexibility by varying the coefficient $\beta(t)$ (Eq.(6) in [27]). Hence, regarding SDE, we consider NFE values from $\{20, 40, 80, 160, 320\}$, and multiply the original $\beta(t) = \dot{\sigma}(t)/\sigma(t)$ [27] with values from $\{0, 0.25, 0.5, 1, 1.5, 2, 4, 8\}$. It is important to note that larger NFE values do not lead to further performance improvements. For restarts, we tried the following two settings: first we set the number of steps in Restart backward process to 40 and vary the number of Restart iterations K in the range $\{0, 5, 10, 15, 20, 25, 30, 35\}$. We also conduct a grid search with the number of Restart iterations K ranging from 5 to 25 and the number of steps in Restart backward process varying from 2 to 7. For ODE, we experiment with the number of steps set to $\{20, 40, 80, 160, 320, 640\}$.

Additionally, we conduct an experiment for Improved SDE in EDM. We try different values of S_{churn} in the range of $\{0, 1, 2, 4, 8, 16, 32, 48, 64\}$. We also perform a grid search where the number of steps ranged from 20 to 320 and S_{churn} takes values of $[0.2 \times \text{steps}, 0.5 \times \text{steps}, 20, 60]$.

The plot combines the results from SDE and is displayed in Figure B.11.

To mitigate the impact of randomness, we collect the data by averaging the results from five runs with the same hyperparameters. To compute the Wasserstein distance between two discrete distributions, we use minimum weight matching.

Plotting the Pareto frontier

We generate the Pareto frontier plots as follows. For the additional sampling error versus contracted error plot, we first sort all the data points based on their additional sampling error and then connect the data points that represent prefix minimums of the contracted error. Similarly, for the NFE versus FID plot, we sort the data points based on their NFE values and connect the points where the FID is a prefix minimum.

B.3.3 Extra Experiments

Numerical Results

In this section, we provide the corresponding numerical results of Figure 5.3a and Figure 5.3b, in Table B.5, B.6 (CIFAR-10 VP, EDM, PFGM++) and Table B.7, B.8 (ImageNet 64×64 EDM), respectively. We also include the performance of Vanilla SDE in those tables. For the evaluation, we compute the Fréchet distance between 50000 generated samples and the pre-computed statistics of CIFAR-10 and ImageNet 64×64. We follow the evaluation protocol in EDM [27] that calculates each FID scores three times with different seeds and reports the minimum.

We also provide the numerical results on the Stable Diffusion model [153], with a classifier guidance weight $w = 2, 3, 5, 8$ in Table B.9, B.10, B.11, B.12. As in [166], we report the zero-shot FID score on 5K random prompts sampled from the COCO validation set. We evaluate CLIP score [163] with the open-sourced ViT-g/14 [164], Aesthetic score by the more recent LAION-Aesthetics Predictor V2 [5]. We average the CLIP and Aesthetic scores over 5K generated samples. The number of function evaluations is two times the sampling steps in the Stable Diffusion model, since each sampling step involves the evaluation of the conditional

[5] https://github.com/christophschuhmann/improved-aesthetic-predictor

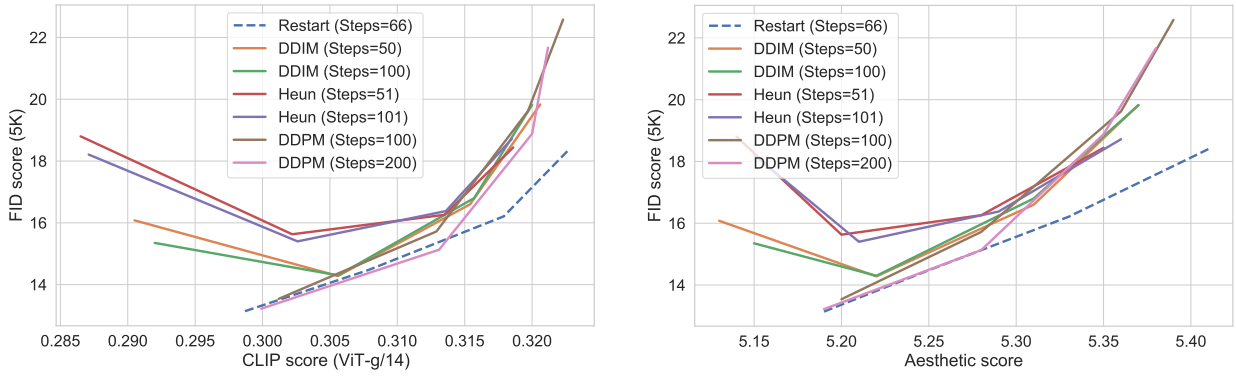
and unconditional model.

Table B.5: CIFAR-10 sample quality (FID score) and number of function evaluations (NFE) on VP [37] for baselines

	NFE	FID
ODE (Heun) [27]	1023	2.90
	511	2.90
	255	2.90
	127	2.90
	63	2.89
	35	2.97
Vanilla SDE [37]	1024	2.79
	512	4.01
	256	4.79
	128	12.57
Gonna Go Fast [28]	1000	2.55
	329	2.70
	274	2.74
	179	2.59
	147	2.95
	49	72.29
Improved SDE [27]	1023	2.35
	511	2.37
	255	2.40
	127	2.58
	63	2.88
	35	3.45

Table B.6: CIFAR-10 sample quality (FID score), number of function evaluations (NFE) and Restart configurations on VP [37], VP with DPM-Solver-3 [26], EDM [27] and PFGM++ [31]

Method	NFE	FID	Configuration $(N_{\text{main}}, N_{\text{Restart},i}, K_i, t_{\text{min},i}, t_{\text{max},i})$
VP			
	519	2.11	$(20, 9, 30, 0.06, 0.20)$
	115	2.21	$(18, 3, 20, 0.06, 0.30)$
	75	2.27	$(18, 3, 10, 0.06, 0.30)$
	55	2.45	$(18, 3, 5, 0.06, 0.30)$
	43	2.70	$(18, 3, 2, 0.06, 0.30)$
VP w/ DPM-Solver-3			
	27	2.11	$(8, 3, 1, 0.06, 0.3)$
	24	2.15	$(7, 3, 1, 0.06, 1)$
	21	2.28	$(6, 3, 1, 0.06, 1)$
	18	2.40	$(5, 3, 1, 0.06, 1)$
EDM			
	43	1.90	$(18, 3, 2, 0.14, 0.30)$
PFGM++			
	43	1.88	$(18, 3, 2, 0.14, 0.30)$



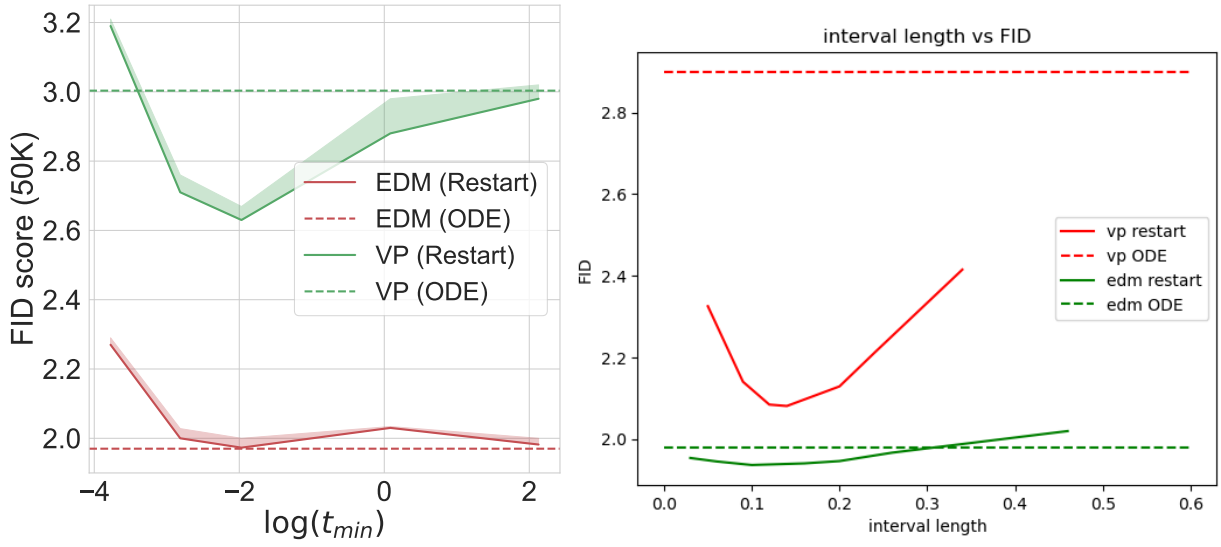
(a) FID versus CLIP score

(b) FID versus Aesthetic score

Figure B.12: FID score versus **(a)** CLIP ViT-g/14 score and **(b)** Aesthetic score for text-to-image generation at 512×512 resolution, using Stable Diffusion v1.5 with varying classifier-free guidance weight $w = 2, 3, 5, 8$.

Table B.7: ImageNet 64×64 sample quality (FID score) and number of function evaluations (NFE) on EDM [27] for baselines

	NFE	FID (50k)
ODE (Heun) [27]	1023	2.24
	511	2.24
	255	2.24
	127	2.25
	63	2.30
	35	2.46
Vanilla SDE [37]	1024	1.89
	512	3.38
	256	11.91
	128	59.71
Improved SDE [27]	1023	1.40
	511	1.45
	255	1.50
	127	1.75
	63	2.24
	35	2.97



(a) (b)

Figure B.13: (a): Adjusting $t_{\min}$ in Restart on VP/EDM; (b): Adjusting the Restart interval length when $t_{\min} = 0.06$.

Table B.8: ImageNet 64×64 sample quality (FID score), number of function evaluations (NFE) and Restart configurations on EDM [27]

NFE	FID (50k)	Configuration N_{main}, $\{(N_{\text{Restart},i}, K_i, t_{\min,i}, t_{\max,i})\}_{i=1}^{l}$
623	1.36	36, {(10, 3, 19.35, 40.79),(10, 3, 1.09, 1.92), (7, 6, 0.59, 1.09), (7, 6, 0.30, 0.59), (7, 25, 0.06, 0.30)}
535	1.39	36, {(6, 1, 19.35, 40.79),(6, 1, 1.09, 1.92), (7, 6, 0.59, 1.09), (7, 6, 0.30, 0.59), (7, 25, 0.06, 0.30)}
385	1.41	36, {(3, 1, 19.35, 40.79),(6, 1, 1.09, 1.92), (6, 5, 0.59, 1.09), (6, 5, 0.30, 0.59), (6, 20, 0.06, 0.30)}
203	1.46	36, {(4, 1, 19.35, 40.79),(4, 1, 1.09, 1.92), (4, 5, 0.59, 1.09), (4, 5, 0.30, 0.59), (6, 6, 0.06, 0.30)}
165	1.51	18, {(3, 1, 19.35, 40.79),(4, 1, 1.09, 1.92), (4, 5, 0.59, 1.09), (4, 5, 0.30, 0.59), (4, 10, 0.06, 0.30)}
99	1.71	18, {(3, 1, 19.35, 40.79),(4, 1, 1.09, 1.92), (4, 4, 0.59, 1.09), (4, 1, 0.30, 0.59), (4, 4, 0.06, 0.30)}
67	1.95	18, {(5, 1, 19.35, 40.79),(5, 1, 1.09, 1.92), (5, 1, 0.59, 1.09), (5, 1, 0.06, 0.30)}
39	2.38	14, {(3, 1, 19.35, 40.79), (3, 1, 1.09, 1.92), (3, 1, 0.06, 0.30)}

Table B.9: Numerical results on Stable Diffusion v1.5 with a classifier-free guidance weight $w = 2$

	Steps	FID (5k) ↓	CLIP score ↑	Aesthetic score ↑
DDIM [25]	50	16.08	0.2905	5.13
	100	15.35	0.2920	5.15
Heun	51	18.80	0.2865	5.14
	101	18.21	0.2871	5.15
DDPM [167]	100	13.53	0.3012	5.20
	200	13.22	0.2999	5.19
Restart	66	13.16	0.2987	5.19

Table B.10: Numerical results on Stable Diffusion v1.5 with a classifier-free guidance weight $w = 3$

	Steps	FID (5k) ↓	CLIP score ↑	Aesthetic score ↑
DDIM [25]	50	14.28	0.3056	5.22
	100	14.30	0.3056	5.22
Heun	51	15.63	0.3022	5.20
	101	15.40	0.3026	5.21
DDPM [167]	100	15.72	0.3129	5.28
	200	15.13	0.3131	5.28
Restart	66	14.48	0.3079	5.25

Table B.11: Numerical results on Stable Diffusion v1.5 with a classifier-free guidance weight $w = 5$

	Steps	FID (5k) ↓	CLIP score ↑	Aesthetic score ↑
DDIM [25]	50	16.60	0.3154	5.31
	100	16.80	0.3157	5.31
Heun	51	16.26	0.3135	5.28
	101	16.38	0.3136	5.29
DDPM [167]	100	19.62	0.3197	5.36
	200	18.88	0.3200	5.35
Restart	66	16.21	0.3179	5.33

Table B.12: Numerical results on Stable Diffusion v1.5 with a classifier-free guidance weight $w = 8$

	Steps	FID (5k) ↓	CLIP score ↑	Aesthetic score ↑
DDIM [25]	50	19.83	0.3206	5.37
	100	19.82	0.3200	5.37
Heun	51	18.44	0.3186	5.35
	101	18.72	0.3185	5.36
DDPM [167]	100	22.58	0.3223	5.39
	200	21.67	0.3212	5.38
Restart	47	18.40	0.3228	5.41

Table B.13: Restart (Steps=66) configurations on Stable Diffusion v1.5

w	Configuration N_{main}, $\{(N_{\text{Restart},i}, K_i, t_{\text{min},i}, t_{\text{max},i})\}_{i=1}^{l}$
2	30, {(5, 2, 1, 9), (5, 2, 5, 10)}
3	30, {(10, 2, 0.1, 3)}
5	30, {(10, 2 0.1, 2)}
8	30, {(10, 2, 0.1, 2)}

Sensitivity Analysis of Hyper-parameters

We also investigate the impact of varying t_{min} when $t_{\text{max}} = t_{\text{min}} + 0.3$, and the length the restart interval when $t_{\text{min}} = 0.06$. Figure B.13a reveals that FID scores achieve a minimum at a t_{min} close to 0 on VP, indicating higher accumulated errors at the end of sampling and poor neural estimations at small t. Note that the Restart interval 0.3 is about twice the length of the one in Table 5.1 and Restart does not outperform the ODE baseline on EDM. This suggests that, as a rule of thumb, we should apply greater Restart strength (*e.g.*larger K, $t_{\text{max}} - t_{\text{min}}$) for weaker or smaller architectures and vice versa.

In theory, a longer interval enhances contraction but may add more additional sampling errors. Again, the balance between these factors results in a V-shaped trend in our plots (Figure B.13b). In practice, selecting t_{max} close to the dataset's radius usually ensures effective mixing when t_{min} is small.

B.3.4 Samples

In this section, we provide extended generated images by Restart, DDIM, Heun and DDPM on text-to-image Stable Diffusion v1.5 model [153]. We showcase the samples of four sets of text prompts in Figure B.14, Figure B.15, Figure B.16, Figure B.17, with a classifier-guidance weight $w = 8$.

(a) Restart (Steps=66)

(b) DDIM (Steps=100)

(c) Heun (Steps=101)

(d) DDPM (Steps=100)

Figure B.14: Generated images with text prompt="A photo of an astronaut riding a horse on mars" and $w = 8$.

(a) Restart (Steps=66)

(b) DDIM (Steps=100)

(c) Heun (Steps=101)

(d) DDPM (Steps=100)

Figure B.15: Generated images with text prompt="A raccoon playing table tennis" and $w = 8$.

(a) Restart (Steps=66)

(b) DDIM (Steps=100)

(c) Heun (Steps=101)

(d) DDPM (Steps=100)

Figure B.16: Generated images with text prompt="Intricate origami of a fox in a snowy forest" and $w = 8$.

(a) Restart (Steps=66)

(b) DDIM (Steps=100)

(c) Heun (Steps=101)

(d) DDPM (Steps=100)

Figure B.17: Generated images with text prompt="A transparent sculpture of a duck made out of glass" and $w = 8$.

B.4 Chapter 6

B.4.1 Discussions

Suboptimality of I.I.D. sampling

Combinatorial analysis Let us consider again the setting of a random variable taking a value equiprobably between N distinct bins. In Fig. B.18, we plot the expected number of modes (or bins) captured as a function of the number of steps as derived in Appendix A.3.6. This suggests that the region where I.I.D. sampling is considerably suboptimal in these regards (capturing the modes of the distribution) is when the number of samples is comparable with the number of modes: if the number of modes is much larger than I.I.D. samples are still likely to capture separate modes, and if the number of samples is much larger the number of uncaptured modes is likely to be small.

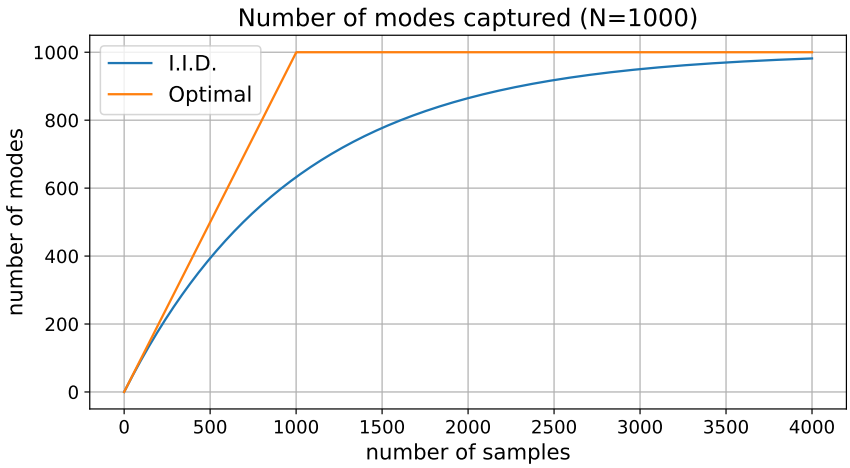


Figure B.18: Plot of the functions $y = N(1 - (\frac{N-1}{N})^x)$ and $y = \min(x, N)$ for $N = 1000$ representing, respectively, the expected number of modes captured by I.I.D. sampling distribution with N equiprobable modes and the optimal coverage.

Literature There is a vast literature that has studied the suboptimality of I.I.D. sampling from a statistical perspective and proposed different solutions. For example, in the field of Bayesian inference, antithetic sampling [255] has been proposed as a way to reduce the variance of Monte Carlo estimates. Determinantal Point Processes [256] have also been widely

studied as a technique to improve the diversity of samples.

Runtime and Memory Overhead

The runtime overhead due to the addition of particle guidance to the inference procedure largely depends on the potential that is used and on the size of the set n. In particular, while computing kernels tends to be significantly cheaper than running the score model, the number of kernel computations scales quadratically with n while the number of score model executions at every step is n.

In terms of memory, particle guidance does not create any significant overhead since the kernel computations can be aggregated per element. However, when running inference on GPU if n is larger than the batch size that fits the GPU memory when running score model inference, the data might have to be moved back and forth between RAM and GPU memory to enable synchronous steps for particle guidance, causing further overhead.

In the case of our experiments on Stable Diffusion, the number of samples extracted (4) does not create significant overhead. However, in the setting of conformer generation on DRUGS different molecules can have very different numbers of conformers with some even having thousands of them. For efficiency, we therefore cap the size of n in particle guidance to 128 and perform batches of 128 samples until the total number of conformers is satisfied.

Other Limitations A badly chosen potential or in general one with a guidance weight too high can overly change the marginal likelihood and negatively impact the sampling quality. As an example in Fig. B.19 the use of a particle guidance parameter four times larger than the best one caused various aliasing artifacts on the image.

Figure B.19: Example of a too large PG weight causing aliasing artifacts.

However, in fixed potential particle guidance, its parameters can be easily fit at inference time, therefore, it is typically relatively inexpensive to test the optimal value of the guidance for the application of interest and the chosen potential. This leads to the prevention of too high guidance

weights and the simple detection of bad potential when the optimal weight is close to 0.

B.4.2 Experimental Details

Synthetic Experiments

To show visually the properties of particle guidance and its effect on sample efficiency, we use a two-dimensional Gaussian mixture model. In particular, we consider a mixture of $N = 10$ identical Gaussian distributions whose centers are equally spaced over the unit circle and whose variance is 0.005. These Gaussians form a set of approximately disjoint equal bins. As we are interested in inference, no model is trained and the true score of the distribution is given as an oracle.

As expected if one runs normal I.I.D. diffusion, the sample falls in one bin at random. Taking ten samples, as shown in Fig. B.20, some of them will fall in the same bin and some bins will be left unfound. The empirical experiments confirm the combinatorial analysis (see Appendix A.3.6) which shows that the expected number of bins discovered with $N = 10$ samples is only 6.5 and it takes on average more than 29 samples to discover all the bins.

In many settings this behavior is suboptimal, and we would want our model to discover all the modes of the distribution with as few samples as possible. Using the straightforward application of particle guidance with a simple RBF kernel based on the squared Euclidean distance, we are able to encourage diversity obtaining, on average, the discovery of nearly 9 bins on average from 10 samples (see Fig. B.20).

Intrinsic diffusion models [257] have shown significant improvements when diffusion models operate on the submanifold where the data lies. Similarly, here building into the kernel the degrees of freedom over which the diversity lies helps the particle guidance to effectively distribute the samples over the distribution. We know that the different modes are distributed in a radial fashion, and thus we build an RBF kernel based on the angle difference w.r.t. the origin. Using this lower-dimensional kernel enables us to consistently discover all modes of the distribution. This submanifold observation aligns well with the practice of methods such as metadynamics where the kernels are defined over some lower-dimensional collective variables of interest.

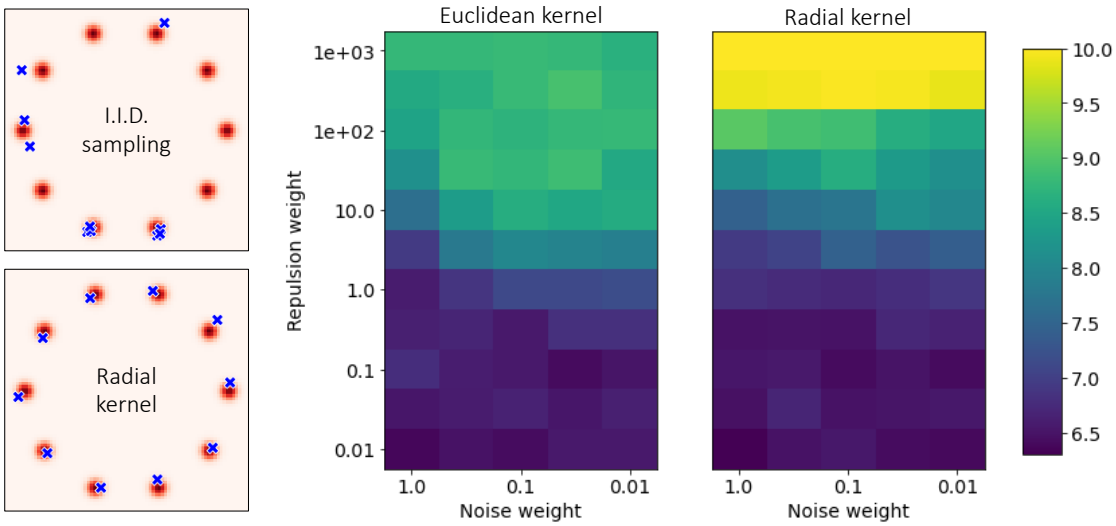


Figure B.20: Left: plot of random samples (in blue) of the two-dimensional Gaussian mixture distribution (density depicted in red). I.I.D. samples often recover the same modes, while particle guidance with a radial kernel captures all modes. Right: average number of modes recovered with 10 samples as a function of the weight given by the diffusion noising terms and the potential weight when using an RBF kernel with Euclidean and radial distances respectively. As expected with little weight to the potential terms we obtain approximately 6.5 modes recovered in line with the I.I.D. diffusion performance. Further increasing the potential weight on the Euclidean creates instability.

Molecular Conformer Generation

Dataset We evaluate the method for the task of molecular conformer generation using the data from GEOM [188], a collection of datasets that has become the standard benchmark for this task in the machine learning community. In particular, we focus on GEOM-DRUGS, the largest, most pharmaceutically relevant and widely used dataset in GEOM which consists of 304k drug-like molecules. For each of these molecules, an ensemble of conformers was generated with metadynamics in CREST [258], a procedure that gives accurate structures but is prohibitive in high-throughput applications, costing an average of 90 core-hours per molecule. To be able to use existing pre-trained models we rely on the experimental setup and splits introduced by [189] and used by several papers afterward. As we do not retrain the score model, we do not use the training set, instead, we finetune the inference parameters for *particle guidance* and the other ablation experiments on a random subset of 200 molecules out of 30433 from the validation set.

Evaluation metrics To evaluate conformer generation methods we want to test the ability of a method to generate a set of conformers that are both individually good poses (precision) and as a set cover the distribution of true conformers (recall). For this we employ the same evaluation setup and metrics used by several papers in the field starting from [189]. In this setup, methods are asked to generate twice as many conformers as in the original ensemble and then the so-called Average Minimum RMSD (AMR) and Coverage (COV) are measured for precision (P) and recall (R). For $K = 2L$ let $\{C_l^*\}_{l\in[1,L]}$ and $\{C_k\}_{k\in[1,K]}$ be respectively the sets of ground truth and generated conformers:

$$\begin{aligned} \text{COV-R} &:= \frac{1}{L}\left|\{l \in [1..L] : \exists k \in [1..K], \text{RMSD}(C_k, C_l^*) < \delta\right| \\ \text{AMR-R} &:= \frac{1}{L}\sum_{l\in[1..L]} \min_{k\in[1..K]} \text{RMSD}(C_k, C_l^*) \end{aligned} \tag{B.1}$$

where δ is the coverage threshold (set to $0.75\mathbf{r}$A for the GEOM-DRUGS experiments). Swapping ground truth and generated conformers in the equations above we obtain the precision metrics.

Baselines As baselines we report the performances of previous methods as measured by [189] and [168]. Cheminformatics conformer prediction methods rely on rules and heuristics derived from chemical structures to fix the local degrees of freedom and then use a combination of search and template techniques to set the more flexible degrees of freedom like torsion angles. The most accurate and widely used such methods include the open-source software RDKit ETKDG [253] and the commercial tool OMEGA [251], [259].

Before the already introduced Torsional Diffusion [168], a number of other machine learning approaches were proposed for this task, among these: GeoMol [189] uses a GNN to sample directly from a random seed local structures around each atom and then torsion angles, GeoDiff [184] defines a equivariant diffusion model over atom coordinates, and CGCF [260] learns an energy-based model over the space of pairwise distance matrices.

Reverse diffusion As discussed in Section 6.4.2, we applied particle guidance to torsional diffusion, as this is currently considered to be state-of-the-art and it uses, like most ML-based

methods before, I.I.D. sampling during inference. We define the particle guidance kernel to operate directly on the implicit hypertorus manifold where torsional diffusion defines the diffusion process, this, at the same time, makes the kernel lower dimensional and it involves a minor modification to the existing inference procedure. The reverse diffusion process that we apply is:

$$d\boldsymbol{\tau}_i = \underbrace{\frac{1}{2}\, g^2(T-t)\, \mathbf{s}(\boldsymbol{\tau}_i, L, T-t)\, dt}_{\text{diffusion ODE}} + \underbrace{\beta_{T-t}\left(\frac{1}{2}\, g^2(t)\, \mathbf{s}(\boldsymbol{\tau}_i, L, T-t)\, dt + g(T-t)\, d\mathbf{w}\right)}_{\text{Langevin diffusion SDE}}$$
$$+ \underbrace{\frac{\gamma_{T-t}}{2}\, g^2(T-t)\, \nabla_{\boldsymbol{\tau}_i} \log \Phi_{T-t}(\boldsymbol{\tau}_1, ..., \boldsymbol{\tau}_n) dt}_{\text{particle guidance}}$$

where we follow the idea from [27] of dividing the different components of the reverse diffusion and tuning their individual parameters. The potential was chosen to be:

$$\log \Phi_t(\boldsymbol{\tau}_1, ...\boldsymbol{\tau}_n) = -\frac{\alpha_t}{2n} \sum_{i,j} k_t(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j) \quad \text{where} \quad k_t(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j) = \exp(-\frac{||\boldsymbol{\tau}_i - \boldsymbol{\tau}_j||^2}{h_t}) \tag{B.2}$$

where the difference of each torsion angle is computed to be in $(-\pi, \pi]$. α_t, β_t, γ_t and h_t are inference hyperparameters that are 'logarithmically interpolated' between two end values chosen with hyperparameter tuning (using $T = 1$), e.g. $\alpha_t = \exp(t \log(\alpha_1) + (1-t) \log(\alpha_0))$.

Permutation invariant kernel Since the kernel operates on the torsion angle differences it is naturally invariant to SE(3) transformations, i.e. translations or rotations, of the conformers in space. Moreover, as illustrated in Fig. 2 of [168], while exact torsion angle values depend on arbitrary choices of neighbors or orientation (to compute the dihedral angle) differences in torsion angles are invariant to these choices. However, one transformation that the kernel in Equation B.2 is not invariant to are permutations of the atoms in the molecule. Many of these permutations lead to isomorphic molecular graphs where however each of the torsion angles may now refer to a different dihedral. To maximize the sample efficiency we make the kernel invariant to these by taking the minimum over the values of the kernel under all such

permutations:

$$k'_t(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j) = \min_{\pi \in \Pi} k_t(\boldsymbol{\tau}_i, P_\pi \boldsymbol{\tau}_j)$$

where Π is the set of all permutations that keep the graph isomorphic (but do change the torsion angles assignment) and P_π is the permutation matrix corresponding to some permutation π. In practice, these isomorphisms can be precomputed efficiently, however, to limit the overhead from applying the kernel multiple times, whenever there are more than 32 isomorphic graphs leading to a change in dihedral assignments we subsample these to only keep 32.

Batch size The number of conformers one has to generate is given, for every molecule, by the benchmark (2L) and can vary significantly. To avoid significant computational overheads, we use batches of up to $n = 128$ until all the conformers for that particular molecule are generated.

Full Results

We provide in Table B.14 again the results reported in Table 6.1 with the additions of other baselines and ablation experiments. In particular, as ablations, on top of running non-invariant particle guidance i.e. without the minimization over the permutations described in the previous section, we also test low-temperature sampling, another variation of the inference-time procedure that has been proposed for diffusion models that we applied as described below.

Low-temperature sampling. Low-temperature sampling of some distribution $p(\mathbf{x})$ with temperature $\lambda^{-1} < 1$ consists of sampling the distribution $p_\lambda(\mathbf{x}) \propto p(\mathbf{x})^\lambda$. This helps mitigate the overdispersion problem by concentrating more on high-likelihood modes and trading off sample diversity for quality. Exact low-temperature sampling is intractable for diffusion models, however, various approximation schemes exist. We use an adaptation of Hybrid

Langevin-Reverse Time SDE proposed by [261]:

$$d\boldsymbol{\tau} = -\left(\lambda_t + \frac{\lambda\,\psi}{2}\right)\mathbf{s}_{\theta,G}(C,t)\;g^2(t)\;dt + \sqrt{1+\psi}\;g(t)\;d\mathbf{w} \quad \text{with } \lambda_t = \frac{\sigma_d + \sigma_t}{\sigma_d + \sigma_t/\lambda}$$

where λ (the inverse temperature), ψ and σ_d are parameters that can be tuned.

Table B.14: Quality of generated conformer ensembles for the GEOM-DRUGS test set in terms of Coverage (%) and Average Minimum RMSD (**r**A). Minimizing recall and precision refers to the hyperparameter choices that minimize the respective median AMR on the validation set.

	Recall				Precision			
	Coverage ↑		AMR ↓		Coverage ↑		AMR ↓	
Method	Mean	Med	Mean	Med	Mean	Med	Mean	Med
RDKit ETKDG	38.4	28.6	1.058	1.002	40.9	30.8	0.995	0.895
OMEGA	53.4	54.6	0.841	0.762	40.5	33.3	0.946	0.854
CGCF	7.6	0.0	1.247	1.225	3.4	0.0	1.837	1.830
GeoMol	44.6	41.4	0.875	0.834	43.0	36.4	0.928	0.841
GeoDiff	42.1	37.8	0.835	0.809	24.9	14.5	1.136	1.090
Torsional Diffusion	72.7	80.0	0.582	0.565	55.2	56.9	0.778	0.729
TD w/ low temperature								
- minimizing recall	73.3	77.7	0.570	0.551	66.4	73.8	0.671	0.613
- minimizing precision	68.0	69.6	0.617	0.604	72.4	81.3	0.607	0.548
TD w/ non-invariant PG								
- minimizing recall	75.8	81.5	**0.542**	**0.520**	66.2	72.4	0.668	0.607
- minimizing precision	58.9	56.8	0.730	0.746	**76.8**	**88.8**	**0.555**	**0.488**
TD w/ invariant PG								
- minimizing recall	**77.0**	**82.6**	0.543	**0.520**	68.9	78.1	0.656	0.594
- minimizing precision	72.5	75.1	0.575	0.578	72.3	83.9	0.617	0.523

Stable Diffusion

Setup

In this section, we detail the experimental setup on Stable Diffusion. We replace the score function ($\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i)$) in the original particle guidance formula (Equation 6.4) with the

classifier-free guidance formula [159]:

$$\tilde{s}(\mathbf{x}_i, c, t') = w\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i, c) + (1-w)\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i)$$

where c symbolizes the text condition, $w \in \mathbb{R}^+$ is the guidance scale, and $\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i, c)/\log p_{t'}(\mathbf{x}_i)$ is the conditional/unconditional scores, respectively. As probability ODE with classifier-free guidance is the prevailing method employed in text-to-image models [160], we substitute the reverse-time SDE in Equation 6.4 with the marginally equivalent ODE. Assuming that $f(\mathbf{x}_i, t') = 0$, the new backward ODE with particle guidance is

$$d\mathbf{x}_i = \left[\frac{1}{2}g^2(t')\left(\tilde{s}(\mathbf{x}_i, c, t') - \alpha_{t'}\nabla_{\mathbf{x}_i}\sum_{j=1}^{n} k_{t'}(\mathbf{x}_i, \mathbf{x}_j)\right)\right]dt$$

Following SVGD [174], we employ RBF kernel $k_t(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j) = \exp(-\frac{||\boldsymbol{\tau}_i - \boldsymbol{\tau}_j||^2}{h_t})$ with $h_t = m_t^2/\log n$, where m_t is the median of particle distances. We implement the kernel both in the original down-sampled pixel space (the latent of VAE) or the feature space of DINO-VIT-b/8 [186]. Defining the DINO feature extractor as g_{DINO}, the formulation in the feature space becomes:

$$d\mathbf{x}_i = \left[\frac{1}{2}g^2(t')\left(\tilde{s}(\mathbf{x}_i, c, t') - \alpha_{t'}\nabla_{\mathbf{x}_i^0}\sum_{j=1}^{n} k_{t'}\left(g_{\text{DINO}}(\mathbf{x}_i^0), g_{\text{DINO}}(\mathbf{x}_j^0)\right)\right)\right]dt$$

where we set the input to the DINO feature extractor g_{DINO} to the $\mathbf{x}_0$-prediction: $\mathbf{x}_i^0 = \mathbf{x}_i + \sigma(t')^2\tilde{s}(\mathbf{x}_i, c, t')$, as $\mathbf{x}_0$-prediction lies in the data manifold rather than noisy images. $\sigma(t)$ is the standard deviation of Gaussian perturbatio kernel given time t in diffusion models. The gradient w.r.t. $\mathbf{x}_i^0$ can be calculated by forward-mode auto-diff. We hypothesize that defining Euclidean distance in the feature space is markedly more natural and effective compared to the pixel space, allowing the repulsion in a more semantically meaningful way. Our experimental results in Section 6.4.2 corroborate the hypothesis.

To construct the data for evaluation, we randomly sample 500 prompts from the COCO validation set [162]. For each prompt, we generate a batch of four images. To get the average CLIP score/Aesthetic score versus in-batch similarity score curve, for I.I.D. sampling, we use $w \in \{6, 7.5, 8.5, 9\}$. We empirically observed that particle guidance achieved a much lower

in-batch similarity score (better diversity) than IID sampling. As diversity typically improves with smaller guidance weights [159], [160], we chose a set of smaller guidance weights for I.I.D. sampling to further improve its diversity, keeping it in the relatively similar range as particle guidance. Hence for particle guidance, we use a set of larger guidance scales: $w \in \{7.5, 8, 9, 9.5, 10\}$. Indeed, the experimental results suggest that even though I.I.D. sampling used a smaller guidance weight to promote diversity, its in-batch similarity score was still worse than that of particle guidance. . We set the hyper-parameter $\alpha_{t'}$ to $8\sigma(t)$ in particle guidance (feature) and $30\sigma(t)^2$ in particle guidance (pixel). We use an Euler solver with 30 NFE in all the experiments.

In-batch similarity score

We propose in-batch similarity score to capture the diversity of a small set of generated samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ given a prompt c:

$$\text{In-batch similarity score}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{1}{n(n-1)} \sum_{i \neq j} \frac{g_{\text{DINO}}(\mathbf{x}_i)^T g_{\text{DINO}}(\mathbf{x}_j)}{||g_{\text{DINO}}(\mathbf{x}_i)||_2 ||g_{\text{DINO}}(\mathbf{x}_j)||_2}$$

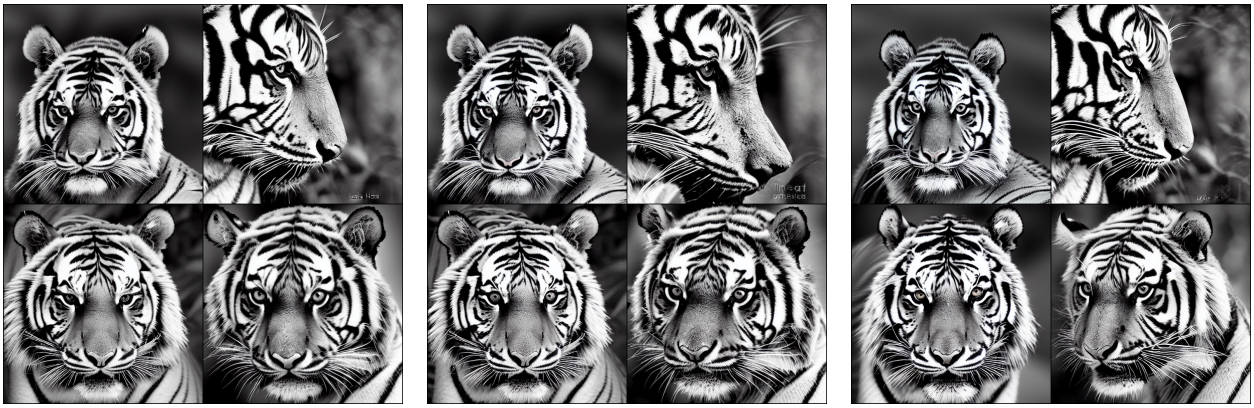
To save memory, we use the DINO-VIT-s/8 [186] as the feature extractor g_{DINO}.

B.4.3 Samples

In Figure B.21-Figure B.25, we visualize samples generated by the I.I.D. sampling process, particle guidance in the pixel space, and particle guidance in the DINO feature space, on four different prompts. For Figure B.21-Figure B.23, we select the prompts in [187], with which Stable Diffusion is shown to replicate content directly from the LAION dataset. We also include the generated samples of SVGD-guidance, in which we replace the particle guidance term with SVGD formula (Equation 6.2). In Figure B.26, we observe that SVGD generally leads to blurry images when increasing the guidance scale α_t. This is predictable as the guidance term in SVGD involves a weighted sum of scores of nearby samples, which will steer the samples toward the mean of samples.

(a) I.I.D. (b) particle guidance (pixel) (c) particle guidance (feature)

Figure B.21: Text prompt: Captain Marvel Exclusive Ccxp Poster Released Online By Marvel



(a) I.I.D. (b) particle guidance (pixel) (c) particle guidance (feature)

Figure B.22: Text prompt: Portrait of Tiger in black and white by Lukas Holas

(a) I.I.D. (b) particle guidance (pixel) (c) particle guidance (feature)

Figure B.23: Text prompt: VAN GOGH CAFE TERASSE copy.jpg

(a) I.I.D. (b) particle guidance (pixel) (c) particle guidance (feature)

Figure B.24: Text prompt: A transparent sculpture of a duck made out of glass

(a) I.I.D. (b) particle guidance (pixel) (c) particle guidance (feature)

Figure B.25: Text prompt: A unicorn in a snowy forest

(a) $\alpha_t = 0.1$ (b) $\alpha_t = 0.3$ (c) $\alpha_t = 1$ (d) $\alpha_t = 2$

Figure B.26: SVGD guidance, with varying α_t

B.5 Chapter 7

B.5.1 Failure of VE/VP-ODE

In Figure 7.6a, we demonstrate the trajectories of cleaner samples/noisier samples/noisier samples w/ corrector. We visualize these three groups in Figure 7.5a and Figure 7.5b. The noisier samples are marked with red boxes in Figure 7.5a and the remaining images in Figure 7.5a are cleaner samples. The samples within green boxes in Figure 7.5b are noisier samples w/ corrector. Samples on the same spatial locations in the two figures are generated by identical initial latents.

The Gaussian kernels in score-based models are $\mathcal{N}(\mathbf{x}, \sigma(t)^2)$ (VE) and $\mathcal{N}(\sqrt{1-\sigma(t)^2}\mathbf{x}, \sigma(t)^2)$ (VP) [34]. When $\sigma(t)$ is large, the norms of perturbed samples are approximately $\sqrt{N}\sigma(t)$. The backward ODE could break down if the trajectories diverge from the norm-$\sigma(t)$ relation, as shown by the noisier samples' trajectories in Figure 7.6a. In contrast, the norm distributions of PFGM is approximately $p(\| \mathbf{x} \|) \propto \| \mathbf{x} \|_2^{N-2} / (\| \mathbf{x} \|_2^2 + z^2)^{\frac{N}{2}}$ when z is large (see deviation for p_{prior} in Appendix A.4.2), which have a wider span for high density region (see Figure 7.3). The weak correlation between norm and z makes PFGM more robust on the lighter NCSNv2 backbone.

B.5.2 Experimental Details

Training

In this section, we include more details about the training of PFGM and other baselines. We show the hyper-parameters settings for all the baselines (Appendix B.5.2). All the experiments are run on a single NVIDIA A100 GPU.

Additional Settings

PFGM We set the hyper-parameters $\gamma = 5$, the larger batch size for calculating normalize field $|\mathcal{B}_L| = 4096$ (CIFAR-10), 256 (CelebA), 64 (LSUN bedroom) in Algorithm 2, and $M = 291$ (CIFAR-10, CelebA)/356 (LSUN bedroom), $\sigma = 0.01$ and $\tau = 0.03$ in Algorithm 3.

We use the a batch size of $|\mathcal{B}| = 128$ (CIFAR-10, CelebA)/32 (LSUN bedroom), the same Adam optimizer and exponential moving average in [34]. We center the data around the origin. The initial z components in the normalized field are approximately zero with small initial $|\epsilon_z|$ values in Algorithm 3. In this case, the trajectories of the forward ODE terminate at points that are unlikely traversed by the backward ODE, *i.e.*, points with large $\| \mathbf{x} \|_2$ and small z. In light of this, we heuristically confine the maximum sampling step to $M = 200$ (CIFAR-10, CelebA)/250 (LSUN bedroom) for points with the initial $|\epsilon_z|$ smaller than 0.005. More principal solutions are left for future works.

For *selecting M in more general settings*, we recommend the following rule-of-thumb. According to analysis in Section A.4.3, given a perturbation point $(\mathbf{y}, z)$ when setting the exponent $m = M$ in Algorithm 3, we can ensure the point source approximation by

$$||\mathbf{y}||^2 \gg \sqrt{N}\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2/2 \tag{B.3}$$

where N is the data dimension and $p(\mathbf{x})$ is the data distribution. By WLLN, we have $||\epsilon_{\mathbf{x}}|| = \sqrt{N}\sigma$, and recall that $\mathbf{y} = \mathbf{x} + \| \epsilon_{\mathbf{x}} \| (1+\tau)^M \mathbf{u}$ where $\epsilon = (\epsilon_{\mathbf{x}}, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I_{N+1\times N+1})$, $\mathbf{u} \sim \mathcal{U}(S_N(1))$. Together, we conclude $||\mathbf{y}|| \approx \sqrt{N}\sigma(1+\tau)^M$. Substituting in Equation B.3, we have

$$M > \frac{1}{2}\log_{1+\tau}\frac{\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2}{2\sqrt{N}\sigma^2} = \frac{1}{2}\frac{\ln\frac{\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2}{2\sqrt{N}\sigma^2}}{\ln 1+\tau}$$

We empirically observe that setting $M = \frac{3}{4}\frac{\ln\frac{\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2}{2\sqrt{N}\sigma^2}}{\ln 1+\tau}$ already gives good results, and the corresponding $||\mathbf{y}|| \approx 3000$. For example, on CIFAR-10 datasets, $N = 3072, \tau = 0.03, \sigma = 0.01, \mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2 \approx 900$, we have $M = \frac{3}{4}\frac{\ln\frac{\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2}{2\sqrt{N}\sigma^2}}{\ln 1+\tau} \approx 291$.

Since we are operating in the augmented space, we add minor modifications to the DDPM++/DDPM++ deep architectures to accommodate the extra dimension. More specifically, we replace the conditioning time variable in VP/sub-VP with the additional dimension z in PFGM as the input to the positional embedding. We also need to add an extra scalar output representing the z direction. To this end, we add an additional output channel to the final convolution layer and take the global average pooling of this channel

to obtain the scalar. For LSUN bedroom dataset, we both experiments with the channel configurations suggested in NSCN++ [34] and DDPM [167].

VE/VP/sub-VP We use the same set of hyper-parameters and the NCSN++/DDPM++ (deep) backbone and the continuous-time training objectives for forward SDEs in [34].

Sampling

We provide more details of PFGM and VE/VP sampling implementations in Appendix B.5.2. We further discuss two techniques used in PFGM ODE sampler: change-of-variable formula (Appendix B.5.2) and the substitution of ground-truth Poisson field direction on z (Appendix B.5.2).

Additional settings

PFGM For RK-45 sampler, we use the function implemented in `scipy.integrate.solve_ivp` with `atol`$=1e-4$, `rtol`$=1e-4$. For forward Euler method, we discretize the ODE with constant step size determined by the number of steps, *i.e.*, step size $= (\log z_{\max} - \log z_{\min})/$number of steps for the backward ODE (Equation 7.6). As in [1], we set the terminal value of $z_{min} = 1e-3$. We choose $z_{\max} = 40$ (CIFAR-10), 60 (CelebA 64^2), 100 (LSUN bedroom) to satisfy the condition $\kappa \gg 1$ by the multipole expansion analysis in Appendix A.4.3. The condition ensures that the data distribution can be viewed roughly as a point source at origin. For example, we set $z_{max} = 40$ on CIFAR-10, and the corresponding κ is greater than 50 with high probability. The hyperparameters work well without further fine tuning. Hence, we hypothesize that PFGM is insensitive to the choice of hyperparameters in a reasonable range, as shown in Table B.15. We clip the norms of initial samples into $(0, 3000)$ for CIFAR-10, $(0, 6000)$ for CelebA and $(0, 30000)$ for LSUN bedroom.

For selecting $z_{\max}$ and clipping upper bound of norms for general datasets, we recommend the following rule-of-thumb. Recall that during the training perturbations (Equation 7.5), given a random initial value $\epsilon_z \sim \mathcal{N}(0, \sigma^2)$, maximum z is

$$z = |\epsilon_z|(1+\tau)^M$$

Hence we set $z_{\max} = \mathbb{E}[|\epsilon_z|(1+\tau)^M] = \sqrt{\frac{2}{\pi}}\sigma(1+\tau)^M$. For example, on CIFAR-10, $\tau = 0.03, M = 291$, and $z_{\max} \approx 43$. The clipping upper value is similarity derived, by setting it to $\mathbb{E}[||\epsilon_{\mathbf{x}}||(1+\tau)^M] = \sqrt{N}\sigma(1+\tau)^M \approx 3000$, where $\epsilon_{\mathbf{x}} \sim \mathcal{N}(0, \sigma^2 I_{N\times N})$. By combining Equation B.3, we further have

$$z_{\max} = \sqrt{\frac{2}{\pi}}\sigma(1+\tau)^M = \sqrt{\frac{2}{\sigma\pi}}\left(\frac{\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2}{2\sqrt{N}}\right)^{\frac{3}{4}}$$
$$\text{clipping upper value} = \sqrt{N}\sigma(1+\tau)^M = \sqrt{\frac{N}{\sigma}}\left(\frac{\mathbb{E}_{p(\mathbf{x})}||\mathbf{x}||^2}{2\sqrt{N}}\right)^{\frac{3}{4}}$$

where N is the data dimension and $p(\mathbf{x})$ is the data distribution. These formulas are easier for practitioner to apply PFGM on new datasets.

VE/VP/sub-VP For the PC sampler in VE, we follow [34] to set the reverse diffusion process as the predictor and the Langevin dynamics (MCMC) as the corrector. For VP/sub-VP, we drop the corrector in PC sampler since it only gives slightly better results [34].

Table B.15: FID scores versus z_{max} on PFGM w/ DDPM++

$\boldsymbol{z_{max}}$	30	40	50
FID score	2.49	2.48	2.48

Exponential Decay on z Dimension

Recall that in Section 7.3.3, we replace the vanilla backward ODE with a new ODE anchored by z:

$$d(\mathbf{x}, z) = (\frac{d\mathbf{x}}{dt}\frac{dt}{dz}dz, dz) = (\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}, 1)dz$$

We further use the change-of-variable formula, *i.e.*, $t' = -\log z$, to achieve exponential decay on the z dimension:

$$d(\mathbf{x}, z) = (\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}z, z)dt'$$

The trajectories of the two ODEs above are the same when $dt, dt' \to 0$. We compare the NFE and the sample quality of different ODEs in Table B.16. We measure the NFE/FID of generating 50000 CIFAR-10 samples with the RK45 method in Scipy package [199]. The batch size is set to 1000. All the numbers are produced on a single NVIDIA A100 GPU. We observe that the ODE with the anchor variable t' not only accelerates the vanilla by 2 times, but has almost no harm to the sample quality measured by FID score.

Table B.16: NFE and FID scores of different backward ODEs in PFGM

Algorithm	$d(\mathbf{x}, z)/dz$	$d(\mathbf{x}, z)/dt'$
NFE	242	104
FID score	2.53	2.48

Substitute the Predicted z Direction with the Ground-truth

Since the neural network cannot perfectly learn the ground-truth z direction, we replace the predicted $f_\theta(x)_z$ with the ground-truth direction when z is small. More specifically, given $\tilde{\mathbf{x}} = (\mathbf{x}, z) \in \mathbb{R}^{N+1}$, recall that the empirical field is $\hat{\mathbf{E}}(\tilde{\mathbf{x}}) = c(\tilde{\mathbf{x}}) \sum_{i=1}^n \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}}$ where $c(\tilde{\mathbf{x}}) = 1/\sum_{i=1}^n \frac{1}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}}$. Hence we can rewrite the empirical field as

$$\hat{\mathbf{E}}(\tilde{\mathbf{x}}) = \sum_{i=1}^n w(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}_i)(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i)$$

where $\sum_{i=1}^n w(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}_i) = \sum_{i=1}^n \frac{\frac{1}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}}}{\sum_{j=1}^n \frac{1}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_j||^{N+1}}} = 1$. Furthermore we have $\forall i, (\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i)_z = z - 0 = z$. Together, the z component in the empirical field is $\hat{\mathbf{E}}(\tilde{\mathbf{x}})_z = \sum_{i=1}^n w(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}_i)(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i)_z = z$. The predicted normalized field (on $\mathbf{x}$) is trained to approximate the normalized field (on $\mathbf{x}$),

i.e.,

$$f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \approx -\sqrt{N}\hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}}/(\sqrt{\| \hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2^2 + z^2} + \gamma)$$
$$\approx -\sqrt{N}\hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}}/(\sqrt{\| \hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2^2} + \gamma)$$

The last approximation is due to $\| \hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2 \gg z$. Solving for $\| \hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2$, we get $\| \hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2 \approx \frac{\gamma \| f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2/\sqrt{N}}{1-\| f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2/\sqrt{N}}$. Hence the z component in the normalized field after substituting the ground-truth is $\hat{\mathbf{E}}(\tilde{\mathbf{x}})_z/(\sqrt{\| \hat{\mathbf{E}}(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2^2 + z^2} + \gamma) = z/(\sqrt{(\frac{\gamma \| f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2/\sqrt{N}}{1-\| f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2/\sqrt{N}})^2 + z^2} + \gamma)$. In our experiments, we therefore replace the original prediction $f_\theta(\tilde{\mathbf{x}})_z$ with $-\sqrt{N}z/(\sqrt{(\frac{\gamma \| f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2/\sqrt{N}}{1-\| f_\theta(\tilde{\mathbf{x}})_{\mathbf{x}} \|_2/\sqrt{N}})^2 + z^2} + \gamma)$ when $z < 5/1/0.1$ during the backward ODE sampling for CIFAR-10/CelebA 64^2/LSUN bedroom 256^2.

Table B.17 reports the NFE and FID score w/o and w/ the above substitution. We observe that the usage of ground-truth z direction in the near field accelerates the sampling speed.

Table B.17: NFE and FID scores of w/ and w/o substitution

Algorithm	w/o substitution	w/ substitution
NFE	134	104
FID score	2.48	2.48

Evaluation

We use FID [41] and Inception scores [55] to quantitatively measure the sample quality, and NFE (number of evaluation steps) for the inference speed. FID (Fréchet Inception Distance) score is the Fréchet distance between two multivariate Gaussians, whose means and covariances are estimated from the 2048-dimensional activations of the Inception-v3 [262] network for real and generated samples respectively. Inception score is the exponential mutual information between the predicted labels of the Inception network and the images. We also report bits/dim for likelihood evaluation. It is computed by dividing the negative log-likelihood by the data dimension, *i.e.*, bits/dim $= -\log p_{\text{prior}}(\mathbf{x})/N$.

For CIFAR-10, we compute the Fréchet distance between 50000 samples and the pre-computed statistics of CIFAR-10 dataset in [41]. For CelebA 64×64, we follow the setting in [49] where the distance is computed between 10000 samples and the test set. For model selection, we follow [49] and pick the checkpoint with smallest FID every 50k iterations on 10k samples for computing all the scores.

Effects of Step Size: FID versus NFE

For preciseness, Table B.18 reports the exact numbers in Figure 7.6c.

Table B.18: The FID scores in Figure 7.6c of different methods and NFE.

Method / NFE	10	20	50	100
VP-ODE	192.36	72.25	38.18	19.73
DDIM	13.36	6.48	4.67	4.16
PFGM	14.98	6.46	3.48	2.89

Since in the ODE $d(\mathbf{x}, z) = -(\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}z, z)dt'$ of PFGM, the z variable is a function of t' ($z = e^{t'}$), we integrate the z in the Euler method to reduce the discretization error. The vanilla update from time t'_i to time t'_{i+1} is $(\mathbf{x}_{i+1}, z_{i+1}) = (\mathbf{x}_i, z_i) - (\mathbf{v}(\tilde{\mathbf{x}}_i)_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}}_i)_{z_i}^{-1}z_i, z_i)(t'_{i+1} - t'_i)$, and the new update is $(\mathbf{x}_{i+1}, z_{i+1}) = (\mathbf{x}_i, z_i) - (\mathbf{v}(\tilde{\mathbf{x}}_i)_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}}_i)_{z_i}^{-1}\int_{t'_i}^{t'_{i+1}} z(t')dt', \int_{t'_i}^{t'_{i+1}} z(t')dt')$. We empirically observe that the new update scheme significantly improve the FID score.

B.5.3 Extra Experiments

LSUN Bedroom 256×256

We report the FID scores and NFEs for LSUN bedroom dataset in Table B.19. We adopt the code base of [34] in our experiments. In [34], they experimented on the LSUN bedroom 256×256 dataset only on VE-SDE using a deeper NCSN++ backbone. In our DDPM++ architecture, we directly borrow the configuration of channels from the NCSN++ architecture [34] in each residual block (PFGM w/ NCSN++ channel). We further change z_{max} to 100, as it empirically gives better sample quality.

We also evaluate the performance when using the configuration of channels in the DDPM [167] architecture (PFGM w/ DDPM channel). We use the RK45 [53] solver in the Scipy library [199] for PFGM sampling. We report the FID score using the evaluation protocol in [90].

Table B.19: FID/NFE on LSUN bedroom 256×256

	FID ↓	NFE ↓
StyleGAN [263]	**2.65**	**1**
DDPM [167]	6.86	1000
VE-SDE [34]	11.75	2000
PFGM w/ NCSN++ channel	17.01	134
PFGM w/ DDPM channel	13.66	122

Table B.19 shows that PFGM has comparable performance with VE-SDE when using DDPM channel, while achieving around 15× acceleration. We observe that PFGM achieves a better FID score using the similar configuration in the DDPM model, and converges faster — 150k over the total 2.4M training iterations suggested in [34]. Remarkably, the VE-ODE baseline — the method most comparable to ours — only produces noisy samples on this dataset. It suggests that PFGM is able to scale up to high resolution images when using advanced architectures. We also compare with the number reported in [167] using similar architecture. Note that DDPM requires 1000 NFE during sampling, and doesn't possess invertibility compared to flow models.

Results on NCSNv2 Architecture

In this section, we demonstrate the image generation on CIFAR-10 and CelebA 64×64, using NCSNv2 architecture [49], which is the predecessor of NCSN++ and DDPM++ [34] and has smaller capacity. Since the VE/VP-ODE has poor performance (FID greater than 90), with the RK45 solver, we also apply the forward Euler method (**Euler**) with fixed number of steps. We explicitly name the sampler, with forward Euler method as predictor and Langevin dynamics as corrector, as **Euler w/ corrector**. For Euler w/ corrector in VE/VP-ODE, we use the probability flow ODE (reverse-time ODE) as the predictor and the Langevin

dynamics (MCMC) as the corrector. We borrow all the hyper-parameters from [34] except for the signal-to-noise ratio. We empirically observe the new configurations in Table B.20 give better results on the NCSNv2 architecture.

To accommodate the extra dimension z on NCSNv2, we concatenate the image with an additional constant channel with value z and thus the first convolution layer takes in four input channels. We also add an additional output channel to the final convolution layer and take the global average pooling of this channel to obtain the direction on z.

Table B.20: Signal-to-noise ratio of different dataset-method pairs

Dataset-Method	CIFAR-10 - VE	CIFAR-10 - VP	CelebA - VE	CelebA - VP
signal-to-noise ratio	0.16	0.27	0.12	0.27

CIFAR-10

Table B.21 reports the image quality measured by Inception/FID scores and the inference speed measured by NFE on CIFAR-10, using a weaker architecture NCSNv2 [49]. We show that PFGM with the RK45 solver has competitive FID/Inception scores with the Langevin dynamics, which was the best model on the NCSNv2 architecture before, and requires 10× less NFE. In addition, PFGM performs better than all the other ODE samplers. Our method is more tolerant of sampling error. Among the compared ODEs, our backward ODE (Equation 7.6) is the only one that successfully generates high quality samples while the VE/VP-ODE fail w/o the Langevin dynamics corrector. The backward ODE still beats the baselines w/ corrector.

CelebA

In Table B.22, we report the quality of images generated by models trained on CelebA 64×64, as measured by the FID scores, and the sampling speed, as measured by NFE. We use this dataset as our preliminary experiments hence we only apply NCSNv2 [49] for different baselines. As shown in Table B.22, PFGM achieves best FID scores than all the baselines on CelebA dataset, while accelerating the inference speed around 20×. Remarkably,

Table B.21: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE). All the methods below the *NCSNv2 backbone* separator use the NCSNv2 [49] network architecture as the backbone.

	Inception ↑	FID ↓	NFE ↓
PixelCNN [15]	4.60	65.93	1024
IGEBM [21]	6.02	40.58	60
WGAN-GP [203]	$7.86 \pm .07$	36.4	1
SNGAN [204]	$8.22 \pm .05$	21.7	1
NCSN [197]	$\mathbf{8.87 \pm .12}$	25.32	1001
NCSNv2 backbone			
Langevin dynamics [49]	$8.40 \pm .07$	**10.87**	1161
VE-SDE [34]	$8.23 \pm .02$	10.94	1000
VP-SDE [34]	$6.85 \pm .01$	44.05	1000
VE-ODE (Euler w/ corrector)	$8.05 \pm .03$	11.33	1000
VP-ODE (Euler w/ corrector)	$7.33 \pm .07$	37.74	1000
PFGM (Euler)	$8.00 \pm .09$	11.78	200
PFGM (RK45)	$8.30 \pm .05$	11.22	**118**

PFGM outperforms the Langevin dynamics and reverse-time SDE samplers, which are usually considered better than their deterministic counterparts.

Remark: On the FID scores on CelebA 64×64 One interesting observation is that the samples of PFGM (RK45) (Figure B.27b) contain more obvious artifacts than Langevin dynamics (Figure B.27a), although PFGM has a lower FID score on the same architecture. We hypothesize that the diversity of samples has larger effects on the FID scores than the artifacts. As shown in Figure B.27a and Figure B.27b, samples generated by PFGM have more diverse background colors and hair colors than samples of Langevin dynamics. In addition, we evaluate the performance of PFGM on the DDPM++ architecture. We show that the FID score can be further reduced to 3.68 using the more advanced DDPM++ architecture. By examining the generated samples of PFGM on DDPM++ (Figure B.31), we observe that the samples are diverse and exhibit fewer artifacts than PFGM on NCSNv2. It suggests that by using a more powerful architecture like DDPM++, we can remove the artifacts while retaining the diversity in PFGM.

(a) Langevin dynamics [197] (b) PFGM (RK45)

Figure B.27: Uncurated samples from Langevin dynamics [197] and PFGM (RK45), both using the NCSNv2 architecture.

Table B.22: FID/NFE on CelebA 64×64

	FID ↓	NFE ↓
NCSN [197]	26.89	1001
NCSNv2 backbone		
Langevin dynamics [49]	10.23	2501
VE-SDE [34]	8.15	1000
VP-SDE [34]	34.52	1000
VE-ODE (Euler w/ corrector)	8.30	200
VP-ODE (Euler w/ corrector)	41.81	200
PFGM (Euler)	7.85	**100**
PFGM (RK45)	7.93	110
DDPM++ backbone		
PFGM (RK45)	**3.68**	110

Wall-clock Sampling Time

The main bottleneck of sampling time in each ODE step is the function evaluation of the neural network. Hence, for different ODE equations using similar neural network architectures, their inference times per ODE step are approximately the same.

We implement PFGM on the NCSNv2 [49], DDPM++ [34], and DDPM++ deep [34] architectures, with sight modifications to account for the extra dimension z. In Table B.23, we report the sampling time per ODE step method with the DDPM++ backbone, as well as the total sampling time. We measure the sampling time of generating a batch of 1000 images on CIFAR-10. We compare PFGM, VP/sub-VP ODEs using the RK45 solver. As a reference, we also report the results of VP-SDE using the predictor-corrector sampler [34]. All the numbers are produced on a single NVIDIA A100 GPU.

Table B.23: Wall-clock sampling time (second)

Method	PFGM	VP-ODE	sub-VP-ODE	VP-SDE (PC)
NFE	110	134	146	1000
Wall-clock time per step	0.526	0.522	0.520	0.491
Total wall-clock time	57.81	69.97	75.92	490.65

As expected, ODEs using similar architectures and the same solver have nearly the same wall-clock time per ODE step. The table also shows that PFGM achieves the smallest total wall-clock sampling time.

Image Interpolations

The invertibility of the ODE in PFGM enables the interpolations between pairs of images. As shown in Figure B.28, we adopt the spherical interpolations between the latent representations of the images in the first and last column.

Temperature Scaling

To demonstrate more utilities of the meaningful latent space of PFGM, we include the experiments of temperature scaling on CelebA 64×64 dataset. We linearly increase the norm

Figure B.28: Interpolation on CelebA 64×64 by PFGM

of latent codes from 1000 to 6000 to get the samples in Figure B.29.

B.5.4 Samples

We provide extended samples from PFGM on CIFAR-10 (Figure B.30), CelebA 64×64 (Figure B.31) and LSUN bedroom 256×256 (Figure B.32) datasets.

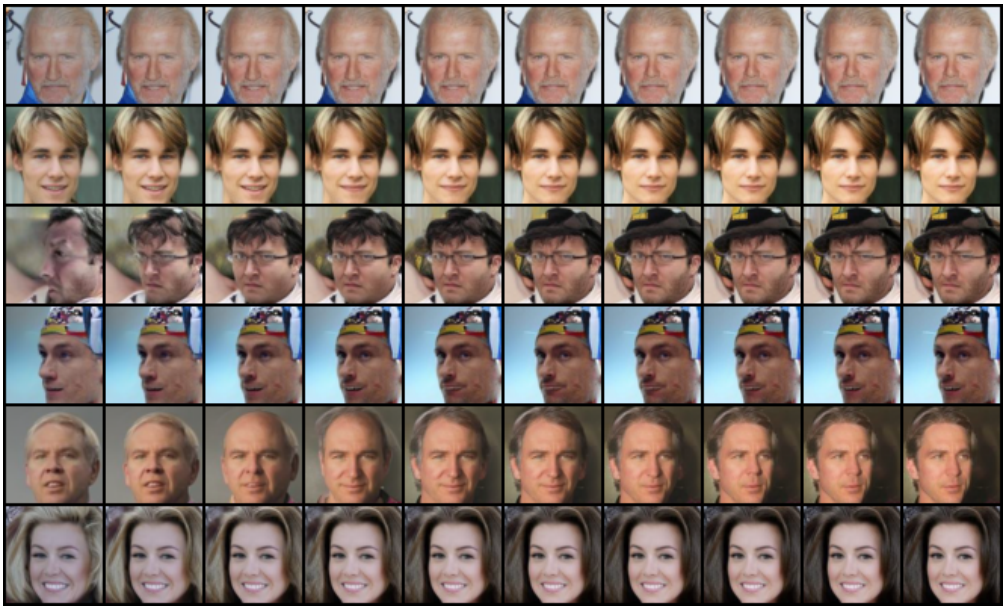


Figure B.29: Temperature scaling on CelebA 64×64 by PFGM

Figure B.30: CIFAR-10 samples from PFGM (RK45)

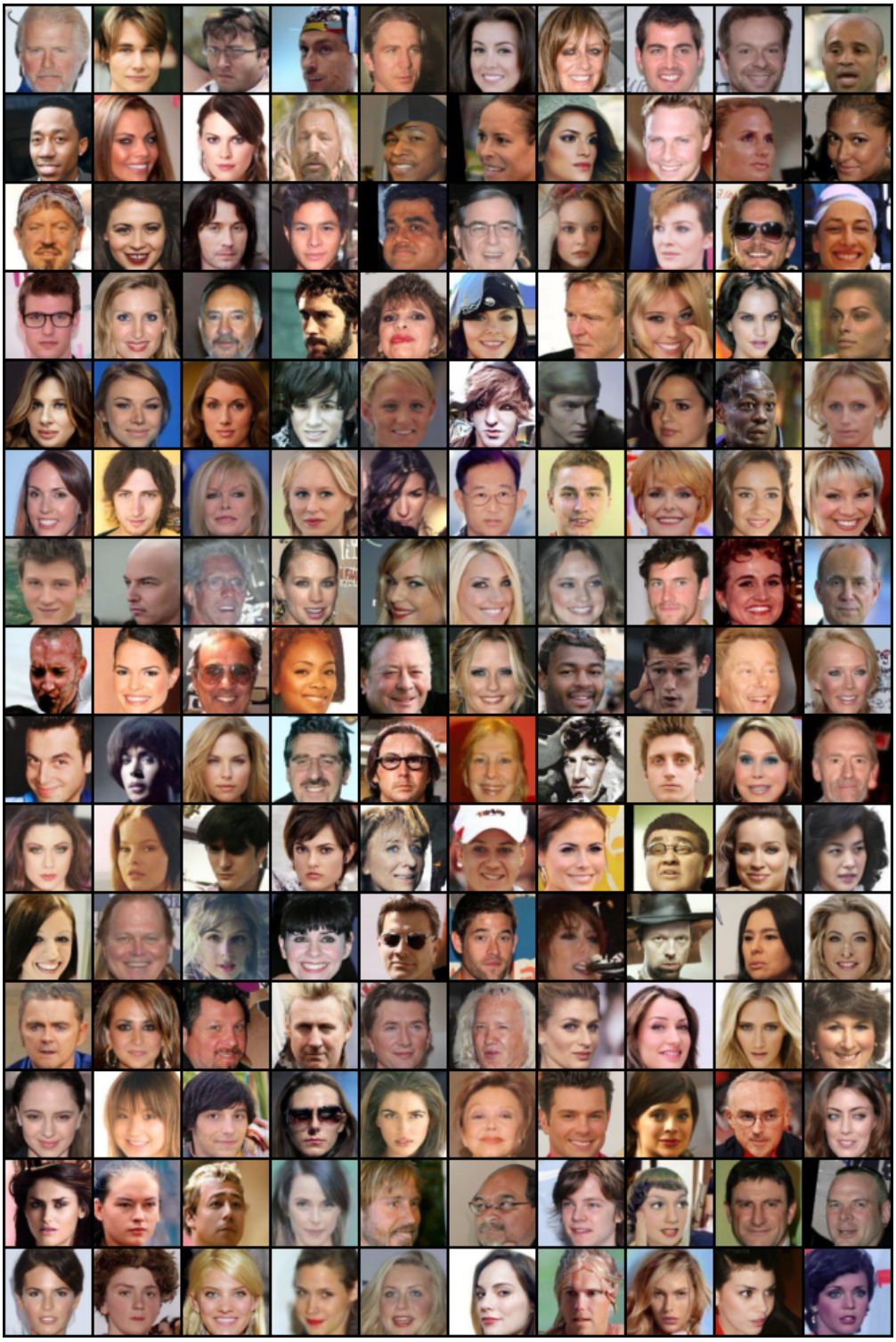


Figure B.31: CelebA 64×64 samples from PFGM (RK45, NCSNv2 architecture)

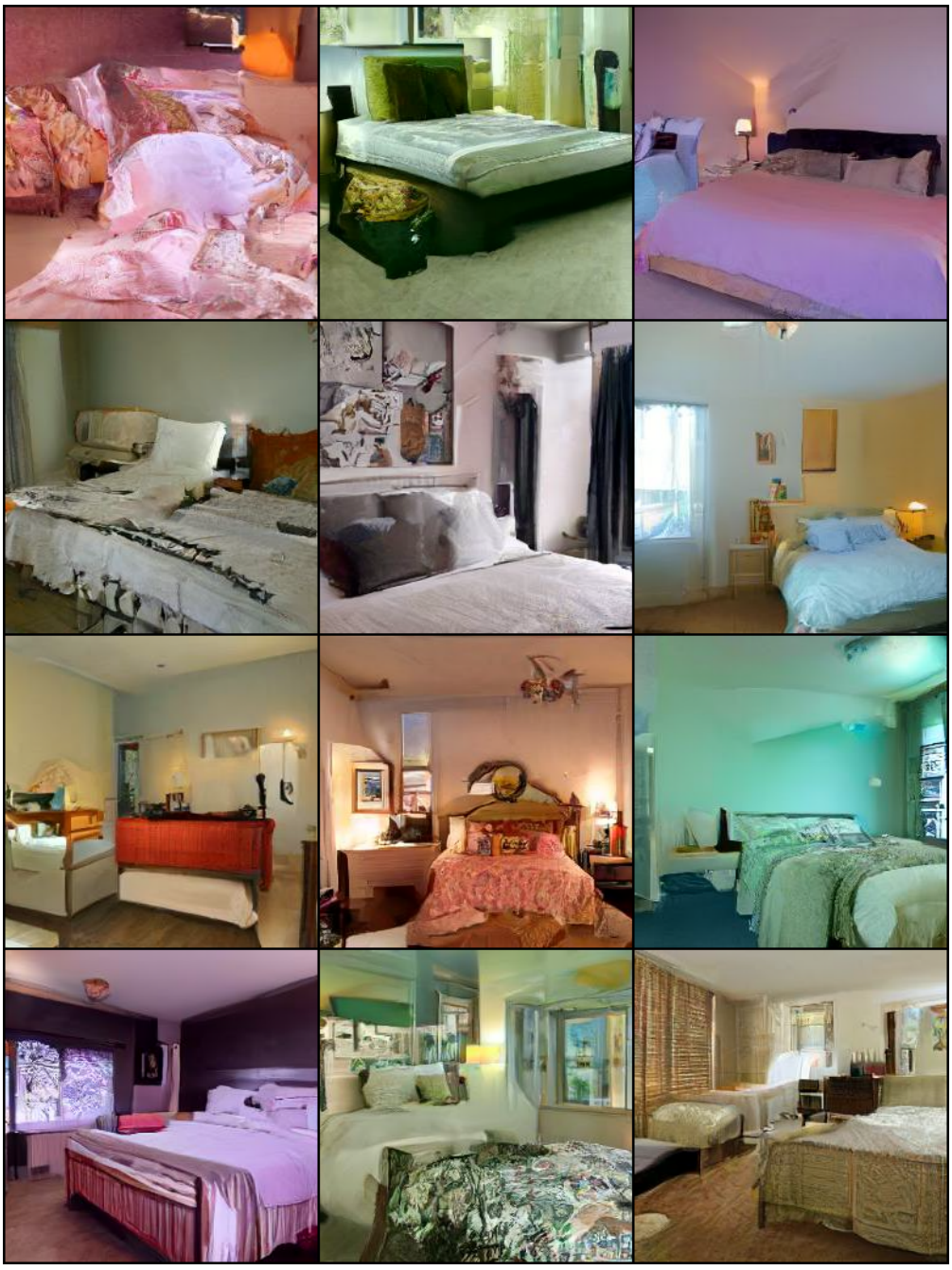


Figure B.32: LSUN bedroom 256×256 samples from PFGM (RK45) using DDPM channel configuration.

B.6 Chapter 8

B.6.1 Aligning the Training in PFGM++ Family

Analysis

In this section, we examine the phase of intermediate marginal distribution p_r under different Ds to derive an alignment method for hyper-parameters. Consider a N-dimensional dataset $\mathcal{D}$ in which the average distance to the nearest neighbor is about l. We consider an arbitrary datapoint $\mathbf{x}_1 \in \mathcal{D}$ and denote its nearest neighbor as $\mathbf{x}_2$. We assume $\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = l$, and uniform prior on $\mathcal{D}$.

To characterize the phases of $p_r, \forall r > 0$, we study the perturbation point $\mathbf{y} \sim p_r(\mathbf{y}|\mathbf{x}_1)$. According to Appendix B.6.2, the distance $\|\mathbf{x}_1 - \mathbf{y}\|$ is roughly $r\sqrt{\frac{N}{D-1}}$. Since $p_r(\mathbf{y}|\mathbf{x}_1)$ is isotropic, with high probability, the two vectors $\mathbf{y} - \mathbf{x}_1, \mathbf{x}_2 - \mathbf{x}_1$ are approximately orthogonal. In particular, the vector product $(\mathbf{y} - \mathbf{x}_1)^T(\mathbf{x}_1 - \mathbf{x}_2) = O(\frac{1}{\sqrt{N}}\|\mathbf{y} - \mathbf{x}_1\|\|\mathbf{x}_1 - \mathbf{x}_2\|) = O(\frac{rl}{\sqrt{D}})$ w.h.p. It reveals that $\|\mathbf{y} - \mathbf{x}_2\| = \sqrt{l^2 + r^2\frac{N}{D-1} + O(\frac{rl}{\sqrt{D}})}$. Figure B.33 depicts the relative positions of $\mathbf{x}_1, \mathbf{x}_2$ and the perturbed point $\mathbf{y}$.

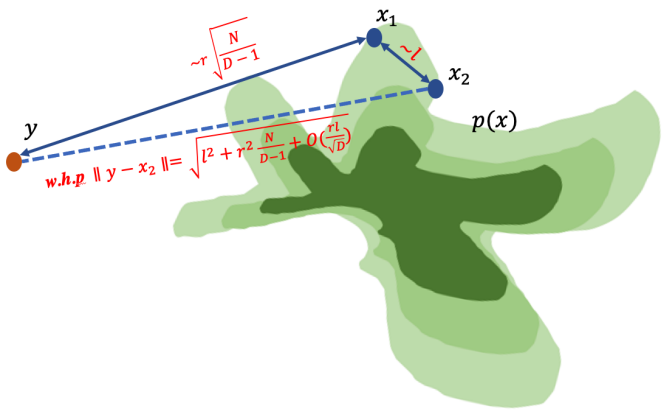


Figure B.33: Illustration of the phase alignment analysis

The ratio of the posterior of the $\mathbf{x}_2$ and $\mathbf{x}_1$ — $\frac{p_r(\mathbf{x}_2|\mathbf{y})}{p_r(\mathbf{x}_1|\mathbf{y})}$ — is an indicator of different phases of field [23]: point in the nearer field tends to have a smaller ratio. Indeed, the ratio would gradually decay from 1 to 0 when moving from the far to the near field. We can calculate the

ratio of the coefficients after approximating the distance $\|\mathbf{y}-\mathbf{x}_2\|$:

$$\begin{aligned}
\frac{p_r(\mathbf{x}_2|\mathbf{y})}{p_r(\mathbf{x}_1|\mathbf{y})} = \frac{p_r(\mathbf{y}|\mathbf{x}_2)}{p_r(\mathbf{y}|\mathbf{x}_1)} &= \left(\frac{l^2 + r^2\frac{N}{D-1} + O(\frac{rl}{\sqrt{D}}) + r^2}{r^2\frac{N}{D-1} + r^2}\right)^{\frac{N+D}{2}} \\
&= \left(1 + \frac{l^2 + O(\frac{rl}{\sqrt{D}})}{r^2\frac{N}{D-1} + r^2}\right)^{\frac{N+D}{2}} \\
&= \exp\left(\ln(1 + \frac{l^2 + O(\frac{rl}{\sqrt{D}})}{r^2\frac{N}{D-1} + r^2}) \cdot \frac{N+D}{2}\right) \\
&\approx \exp\left(\frac{l^2 + O(\frac{rl}{\sqrt{D}})}{r^2\frac{N}{D-1} + r^2} \cdot \frac{N+D}{2}\right) \\
&= \exp\left(\frac{l^2 + O(\frac{rl}{\sqrt{D}})}{r^2} \cdot \frac{N+D}{2(N+D-1)} \cdot (D-1)\right) \\
&\approx \exp\left(\frac{l^2 + O(\frac{rl}{\sqrt{D}})}{r^2} \cdot D\right) \qquad \text{(B.4)}
\end{aligned}$$

Hence the relation $r \propto \sqrt{D}$ should hold to keep the ratio invariant of the parameter D. On the other hand, by Theorem 8 we know that p_σ is equivalent to $p_{r=\sigma\sqrt{D}}$ when $D \to \infty$. To achieve phase alignment on the dataset, one should roughly set $r = \sigma\sqrt{D}$.

Practical Hyperparameter Transfer from Diffusion Models

Transfer EDM training and sampling

We list out and compare the EDM training algorithm (Alg 11) and the PFGM++ with transferred hyper-parameters (Alg 12). The major modification is to replace the Gaussian noise $\mathbf{n}_i \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$ with the additive noise $R_i\mathbf{v}_i \sim \mathcal{U}_\psi(\psi)p_r(R)$, where $r = \sigma\sqrt{D}$. We highlight the major modifications in blue.

We also show the sampling algorithms of EDM (Alg 13) and PFGM++ (Alg 14). Note that we only change the prior sampling process while the for-loop is identical for both algorithms, since EDM [27] sets $\sigma = t$, and $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}r} = \frac{\mathbf{x}-f_\theta(\mathbf{x},r)}{r} = \frac{\mathbf{x}-f_\theta(\mathbf{x},r)}{\sigma\sqrt{D}} = \frac{\mathrm{d}\mathbf{x}}{\sqrt{D}\mathrm{d}\sigma} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\sigma}\frac{\mathrm{d}\sigma}{\mathrm{d}r} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\sigma} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. Thus we can use the original samplers of EDM without further modification.

Algorithm 11 EDM training

Sample a batch of data $\{\mathbf{y}_i\}_{i=1}^{\mathcal{B}}$ from $p(\mathbf{y})$
Sample standard deviations $\{\sigma_i\}_{i=1}^{\mathcal{B}}$ from $p(\sigma)$
Sample noise vectors $\{\mathbf{n}_i \sim \mathcal{N}(0, \sigma_i^2 \boldsymbol{I})\}_{i=1}^{\mathcal{B}}$
Get perturbed data $\{\hat{\mathbf{y}}_i = \mathbf{y}_i + \mathbf{n}_i\}_{i=1}^{\mathcal{B}}$
Calculate loss $\ell(\theta) = \sum_{i=1}^{\mathcal{B}} \lambda(\sigma_i) \|f_\theta(\hat{\mathbf{y}}_i, \sigma_i) - \mathbf{y}_i\|_2^2$
Update the network parameter θ via Adam optimizer

Algorithm 12 PFGM++ training with hyper-parameter transferred from EDM

Sample a batch of data $\{\mathbf{y}_i\}_{i=1}^{\mathcal{B}}$ from $p(\mathbf{y})$
Sample standard deviations $\{\sigma_i\}_{i=1}^{\mathcal{B}}$ from $p(\sigma)$
Sample r from p_r: $\{r_i = \sigma_i \sqrt{D}\}_{i=1}^{\mathcal{B}}$
Sample radiuses $\{R_i \sim p_{r_i}(R)\}_{i=1}^{\mathcal{B}}$
Sample uniform angles $\{\mathbf{v}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|_2}\}_{i=1}^{\mathcal{B}}$, with $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
Get perturbed data $\{\hat{\mathbf{y}}_i = \mathbf{y}_i + R_i \mathbf{v}_i\}_{i=1}^{\mathcal{B}}$
Calculate loss $\ell(\theta) = \sum_{i=1}^{\mathcal{B}} \lambda(\sigma_i) \|f_\theta(\hat{\mathbf{y}}_i, \sigma_i) - \mathbf{y}_i\|_2^2$
Update the network parameter θ via Adam optimizer

Algorithm 13 EDM sampling (Heun's 2nd order method)

$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \boldsymbol{I})$
for $i = 0, \ldots, T-1$ **do**
  $\mathbf{d}_i = (\mathbf{x}_i - f_\theta(\mathbf{x}_i, t_i))/t_i$
  $\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i)\mathbf{d}_i$
  if $t_{i+1} > 0$ **then**
    $\mathbf{d}'_i = (\mathbf{x}_{i+1} - f_\theta(\mathbf{x}_{i+1}, t_{i+1}))/t_{i+1}$
    $\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i)(\frac{1}{2}\mathbf{d}_i + \frac{1}{2}\mathbf{d}'_i)$
  end if
end for

Algorithm 14 PFGM++ training with hyper-parameter transferred from EDM

Set $r_{\max} = \sigma_{\max}\sqrt{D}$
Sample radius $R \sim p_{r_{\max}}(R)$ and uniform angle $\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|_2}$, with $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
Get initial data $\mathbf{x}_0 = R\mathbf{v}$
for $i = 0, \ldots, T-1$ **do**
  $\mathbf{d}_i = (\mathbf{x}_i - f_\theta(\mathbf{x}_i, t_i))/t_i$
  $\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i)\mathbf{d}_i$
  if $t_{i+1} > 0$ **then**
    $\mathbf{d}'_i = (\mathbf{x}_{i+1} - f_\theta(\mathbf{x}_{i+1}, t_{i+1}))/t_{i+1}$
    $\mathbf{x}_{i+1} = \mathbf{x}_i + (t_{i+1} - t_i)(\frac{1}{2}\mathbf{d}_i + \frac{1}{2}\mathbf{d}'_i)$
  end if
end for

Transfer DDPM (continuous) training and sampling: Here we demonstrate the "zero-shot" transfer of hyperparameters from DDPM to PFGM++, using the $r = \sigma\sqrt{D}$ formula. We highlight the modifications in blue. In particular, we list the DDPM training/sampling algorithms (Alg 15/Alg 17), and their counterparts in PFGM++ (Alg 16/Alg 18) for comparisons. Let β_T and β_1 be the maximum/minimum values of β in DDPM [167]. Similar to [34], we denote $\alpha_t = e^{-\frac{1}{2}t^2(\bar{\beta}_{\max} - \bar{\beta}_{\min}) - t\bar{\beta}_{\min}}$, with $\bar{\beta}_{\max} = \beta_T \cdot T$ and $\bar{\beta}_{\min} = \beta_1 \cdot T$. For example, on CIFAR-10, $\bar{\beta}_{\min} = 1e-1$ and $\bar{\beta}_{\max} = 20$ with $T = 1000$. We would like to note that the t_is in the sampling algorithms (Alg 17 and Alg 18) monotonically decrease from 1 to

0 as i increases.

Algorithm 15 DDPM training

1: Sample a batch of data $\{\mathbf{y}_i\}_{i=1}^{\mathcal{B}}$ from $p(\mathbf{y})$
2: Sample time $\{t_i{=}t_i'/T\}_{i=1}^{\mathcal{B}}$ with $t_i'{\sim}\mathcal{U}(\{1,\dots,T\})$
3: Get perturbed data $\{\hat{\mathbf{y}}_i = \sqrt{\alpha_{t_i}}\mathbf{y}_i + \sqrt{1-\alpha_{t_i}}\boldsymbol{\epsilon}_i\}_{i=1}^{\mathcal{B}}$, where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: Calculate loss $\ell(\theta) = \sum_{i=1}^{\mathcal{B}} \lambda(t_i)\|f_\theta(\hat{\mathbf{y}}_i, t_i) - \boldsymbol{\epsilon}_i\|_2^2$
5: Update the network parameter θ via Adam optimizer

Algorithm 16 PFGM++ training with hyper-parameter transferred from DDPM

1: Sample a batch of data $\{\mathbf{y}_i\}_{i=1}^{\mathcal{B}}$ from $p(\mathbf{y})$
2: Sample time $\{t_i\}_{i=1}^{\mathcal{B}}$ from $\mathcal{U}[0,1]$
3: Get σ_i from t_i: $\{\sigma_i = \sqrt{\frac{1-\alpha_{t_i}}{\alpha_{t_i}}}\}$
4: Sample r from p_r: $\{r_i = \sigma_i\sqrt{D}\}_{i=1}^{\mathcal{B}}$
5: Sample radiuses $\{R_i \sim p_{r_i}(R)\}_{i=1}^{\mathcal{B}}$
6: Sample uniform angles $\{\mathbf{v}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|_2}\}_{i=1}^{\mathcal{B}}$, with $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
7: Get perturbed data $\{\hat{\mathbf{y}}_i = \sqrt{\alpha_{t_i}}(\mathbf{y}_i + R_i\mathbf{v}_i)\}_{i=1}^{\mathcal{B}}$
8: Calculate loss $\ell(\theta) = \sum_{i=1}^{\mathcal{B}} \lambda(t_i)\|f_\theta(\hat{\mathbf{y}}_i, t_i) - \frac{\sqrt{D}R_i\mathbf{v}_i}{r}\|_2^2$
9: Update the network parameter θ via Adam optimizer

Algorithm 17 DDIM sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
2: **for** $i = T, \dots, 1$ **do**
3: $\quad \mathbf{x}_{i-1} = \sqrt{\frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}}\mathbf{x}_i + (\sqrt{1-\alpha_{t_{i-1}}} - \sqrt{\frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}}\sqrt{1-\alpha_{t_i}})f_\theta(\mathbf{x}_i, t_i)$
4: **end for**

Algorithm 18 PFGM++ sampling transferred from DDIM

1: Set $\sigma_{\max} = \sqrt{\frac{1-\alpha_1}{\alpha_1}}, r_{\max} = \sigma_{\max}\sqrt{D}$
2: Sample radius $R \sim p_{r_{\max}}(R)$ and uniform angle $\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|_2}$, with $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: Get initial data $\mathbf{x}_T = \sqrt{\alpha_1}R\mathbf{v}$
4: **for** $i = T, \dots, 1$ **do**
5: $\quad \mathbf{x}_{i-1} = \sqrt{\frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}}\mathbf{x}_i + (\sqrt{1-\alpha_{t_{i-1}}} - \sqrt{\frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}}\sqrt{1-\alpha_{t_i}})f_\theta(\mathbf{x}_i, t_i)$
6: **end for**

B.6.2 Experimental Details

We show the experimental setups in section 8.4, as well as the training, sampling, and evaluation details for PFGM++.

Experiments for the Analysis in Sec 8.4

In the experiments of section 8.3 and section 8.4.1, we need to access the posterior $p_{0|r}(\mathbf{y}|\mathbf{x}) \propto p_r(\mathbf{x}|\mathbf{y})p(\mathbf{y})$ to calculate the mean TVD. We sample a large batch $\{\mathbf{y}_i\}_{i=1}^n$ with $n = 1024$ on

CIFAR-10 to empirically approximate the posterior:

$$p_{0|r}(\mathbf{y}_i|\mathbf{x}) = \frac{p_r(\mathbf{x}|\mathbf{y}_i)p(\mathbf{y}_i)}{p_r(\mathbf{x})} \approx \frac{p_r(\mathbf{x}|\mathbf{y}_i)}{\sum_{j=1}^n p_r(\mathbf{x}|\mathbf{y}_j)} = \frac{1/(\|\mathbf{x}-\mathbf{y}_i\|_2^2 + r^2)^{\frac{N+D}{2}}}{\sum_{j=1}^n 1/(\|\mathbf{x}-\mathbf{y}_j\|_2^2 + r^2)^{\frac{N+D}{2}}}$$

We sample a large batch of 256 to approximate all the expectations in section 8.4, such as the average TVDs.

Training Details

We borrow the architectures, preconditioning techniques, optimizers, exponential moving average (EMA) schedule, and hyper-parameters from previous state-of-the-art diffusion model EDM [27]. We apply the alignment method in section 8.3 to transfer their well-tuned hyper-parameters.

For architecture, we use the improved NCSN++ [27] for the CIFAR-10 dataset (batch size 512), and the improved DDPM++ for the FFHQ dataset (batch size 256). Since [27] does not experiment on LSUN Churches dataset, we set the number of blocks to 2, and the feature maps ($\times\frac{1}{128}$) to 1-1-2-2-2-2-2 without augmentation, inspired by the architecture in [34]. For optimizers, following EDM, we adopt the Adam optimizer with a learning rate of $10e-4$. We further incorporate the EMA schedule, learning rate warm-up, and data augmentations in EDM. Please refer to Appendix F in EDM paper [27] for details.

The most prominent improvements in EDM are the preconditioning and the new training distribution for σ, *i.e.*, $p(\sigma)$. Specifically, adding these two techniques to the vanilla diffusion objective (Equation 8.6), their effective training objective can be written as:

$$\mathbb{E}_{\sigma\sim p(\sigma)}\lambda(\sigma)c_{\text{out}}(\sigma)^2\mathbb{E}_{p(\mathbf{y})}\mathbb{E}_{p_\sigma(\mathbf{x}|\mathbf{y})}\left[\left\|F_\theta(c_{\text{in}}(\sigma)\cdot\mathbf{x}, c_{\text{noise}}(\sigma)) - \frac{1}{c_{\text{out}}(\sigma)}(\mathbf{y} - c_{\text{skip}}(\sigma)\cdot\mathbf{x})\right\|_2^2\right] \tag{B.5}$$

with the predicted normalized score function in the vanilla diffusion objective (Equation 8.6) re-parameterized as

$$f_\theta(\mathbf{x},\sigma) = \frac{c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)\mathbf{x}, c_{\text{noise}}(\sigma)) - x}{\sigma} \approx \sigma\nabla_{\mathbf{x}}\log p_\sigma(x)$$

$c_{\text{in}}(\sigma) = 1/\sqrt{\sigma^2 + \sigma_{\text{data}}^2}, c_{\text{out}}(\sigma) = \sigma \cdot \sigma_{\text{data}}/\sqrt{\sigma^2 + \sigma_{\text{data}}^2}, c_{\text{skip}}(\sigma) = \sigma_{\text{data}}^2/(\sigma^2 + \sigma_{\text{data}}^2), c_{\text{noise}}(\sigma) = \frac{1}{4}\ln(\sigma)$, with $\sigma_{\text{data}} = 0.5$. $\{c_{\text{in}}(\sigma), c_{\text{out}}(\sigma), c_{\text{skip}}(\sigma), c_{\text{data}}, c_{\text{noise}}(\sigma)\}$ are all the hyper-parameters in the preconditioning. The training distribution $p(\sigma)$ is the log-normal distribution with $\ln(\sigma) \sim \mathcal{N}(-1.2, 1.2^2)$, and the loss weighting $\lambda(\sigma) = 1/c_{\text{out}}(\sigma)^2$.

Recall that the hyper-parameter alignment rule $r = \sigma\sqrt{D}$ can transfer the hyper-parameter from diffusion models ($D \to \infty$) to finite Ds. Hence we can directly set $\sigma = r/\sqrt{D}$ in those hyper-parameters for preconditioning. In addition, the training distribution $p(r)$ can be derived via the change-of-variable formula, *i.e.*, $p(r) = p(\sigma = r/\sqrt{D})/\sqrt{D}$. The final PFGM++ objective after incorporating these techniques into Equation 8.4 is:

$$\mathbb{E}_{r\sim p(r)}\lambda(r/\sqrt{D})c_{\text{out}}(r/\sqrt{D})^2\mathbb{E}_{p(\mathbf{y})}\mathbb{E}_{p_r(\mathbf{x}|\mathbf{y})}\left[\left\|F_\theta(c_{\text{in}}(r/\sqrt{D})\cdot\mathbf{x}, c_{\text{noise}}(r/\sqrt{D})) - \frac{1}{c_{\text{out}}(\sigma)}(\mathbf{y} - c_{\text{skip}}(r/\sqrt{D})\cdot\mathbf{x})\right\|_2^2\right]$$

with the predicted normalized electric field in the vanilla PFGM++ objective (Equation 8.4) re-parameterized as

$$f_\theta(\tilde{\mathbf{x}}) = \frac{c_{\text{skip}}(r/\sqrt{D})\mathbf{x} + c_{\text{out}}(r/\sqrt{D})F_\theta(c_{\text{in}}(r/\sqrt{D})\mathbf{x}, c_{\text{noise}}(r/\sqrt{D})) - x}{r/\sqrt{D}} \approx \sqrt{D}\frac{\mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}}{E(\tilde{\mathbf{x}})_r}$$

Sampling Details

For sampling, following EDM [27], we also use Heun's 2nd method (improved Euler method) [154] as the ODE solver for $\mathrm{d}\mathbf{x}/\mathrm{d}r = \mathbf{E}(\tilde{\mathbf{x}})_{\mathbf{x}}/E(\tilde{\mathbf{x}})_r = f_\theta(\tilde{\mathbf{x}})/\sqrt{D}$.

We adopt the same parameterized scheme in EDM to determine the evaluation points during N-step ODE sampling:

$$r_i = ({r_{\max}}^{\frac{1}{\rho}} + \frac{i}{N-1}({r_{\min}}^{\frac{1}{\rho}} - {r_{\max}}^{\frac{1}{\rho}}))^\rho \quad \text{and} \quad r_N = 0$$

where ρ controls the relative density of evaluation points in the near field. We set $\rho = 7$ as in EDM, and $r_{\max} = \sigma_{\max}\sqrt{D} = 80\sqrt{D}, r_{\min} = \sigma_{\min}\sqrt{D} = 0.002\sqrt{D}$ ($\sigma_{\max}, \sigma_{\min}$ are the hyper-parameters in EDM, controlling the starting/terminal evaluation points) following the $r = \sigma\sqrt{D}$ alignment rule.

Practical Sampling Procedures of Perturbation Kernel and Prior Distribution

In this section, we discuss how to simple from the perturbation kernel $p_r(\mathbf{x}|\mathbf{y})$ $\propto 1/(\|\mathbf{x}-\mathbf{y}\|_2^2 + r^2)^{\frac{N+D}{2}}$ in practice. We first decompose $p_r(\cdot|\mathbf{y})$ in hyperspherical coordinates to $\mathcal{U}_\psi(\psi)p_r(R)$, where $\mathcal{U}_\psi$ is the uniform distribution over the angle component and the distribution of the perturbed radius $R = \|\mathbf{x}-\mathbf{y}\|_2$ is

$$p_r(R) \propto \frac{R^{N-1}}{(R^2+r^2)^{\frac{N+D}{2}}} \tag{B.6}$$

The sampling procedure of the radius distribution encompasses three steps:

$$\begin{aligned} R_1 &\sim \text{Beta}(\alpha{=}\frac{N}{2}, \beta = \frac{D}{2}) \\ R_2 &= \frac{R_1}{1-R_1} \\ R_3 &= \sqrt{r^2 R_2} \end{aligned}$$

Next, we prove that $p(R_3) = p_r(R_3)$. Note that the pdf of the inverse beta distribution is

$$p(R_2) \propto R_2^{\frac{N}{2}-1}(1+R_2)^{-\frac{N+D}{2}}$$

By change-of-variable, the pdf of $R_3 = \sqrt{r_{max}^2 R_2}$ is

$$\begin{aligned} p(R_3) &\propto R_2^{\frac{N}{2}-1}(1+R_2)^{-\frac{N}{2}-\frac{D}{2}} * \frac{2R_3}{r_{max}^2} \\ &\propto \frac{R_3 R_2^{\frac{N}{2}-1}}{(1+R_2)^{\frac{N+D}{2}}} \\ &= \frac{(R_3/r)^{N-1}}{(1+(R_3^2/r^2))^{\frac{N+D}{2}}} \\ &\propto \frac{R_3^{N-1}}{(1+(R_3^2/r^2))^{\frac{N+D}{2}}} \\ &\propto \frac{R_3^{N-1}}{(r^2+R_3^2)^{\frac{N+D}{2}}} \propto p_r(R_3) \qquad \text{(By Equation B.6)} \end{aligned}$$

Note that R_1 has mean $\frac{N}{N+D}$ and variance $O(\frac{ND}{(N+D)^3})$. Hence when $D = O(N)$, $p_r(R)$

would highly concentrate on a specific value, resolving the heavy-tailed problem. We can sample the uniform angel component by $\mathbf{u} = \mathbf{w}/\|\mathbf{w}\|, \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{N\times N})$. Together, sampling from the perturbation kernel $p_r(\mathbf{x}|\mathbf{y})$ is equivalent to setting $\mathbf{x} = \mathbf{y} + R_3\mathbf{u}$. On the other hand, the prior distribution is

$$p_{r_{\max}}(\mathbf{x}) \propto \lim_{r_{\max}\to\infty} \int r_{\max}^D/\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|^{N+D} p(\mathbf{y}) d\mathbf{y} = \lim_{r_{\max}\to\infty} r_{\max}^D/(\|\mathbf{x}\|^2 + r_{\max}^2)^{\frac{N+D}{2}}$$

We observe that $p_{r_{\max}}(\mathbf{x})$ the same as the perturbation kernel $p_{r_{\max}}(\mathbf{x}|\mathbf{y} = \mathbf{0})$. Hence we can sample from the prior following $\mathbf{x} = R_3\mathbf{u}$ with $R_3, \mathbf{u}$ defined above and $r = r_{\max}$.

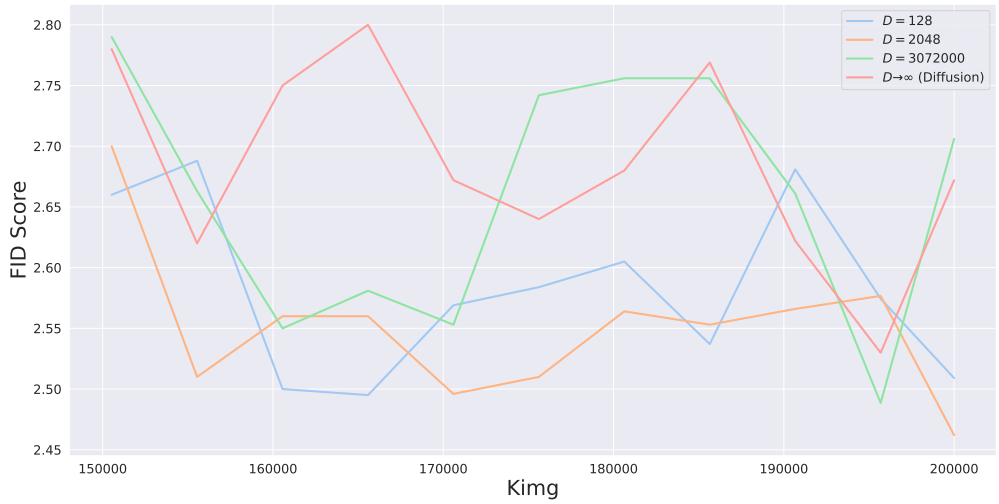
Evaluation Details

For the evaluation, we compute the Fréchet distance between 50000 generated samples and the pre-computed statistics of CIFAR-10 and FFHQ. On CIFAR-10, we follow the evaluation protocol in EDM [27], which repeats the generation three times with different seeds for each checkpoint and reports the minimum FID score. However, we observe that the FID score has a large fluctuation across checkpoints, and the minimum FID score of EDM in our re-run experiment does not align with the original results reported in [27]. Figure B.34a shows that the FID score could have a variation of ± 0.2 during the training of a total of 200 million images [27]. To better evaluate the model performance, Table 8.2 reports the average FID over the Top-3 checkpoints instead. In Figure B.34b, we further demonstrate the moving average of the FID score with a window of 10000K images. It shows that $D = 2048$ consistently outperforms other baselines in the same training iterations, in agreement with the results in Table 8.2.

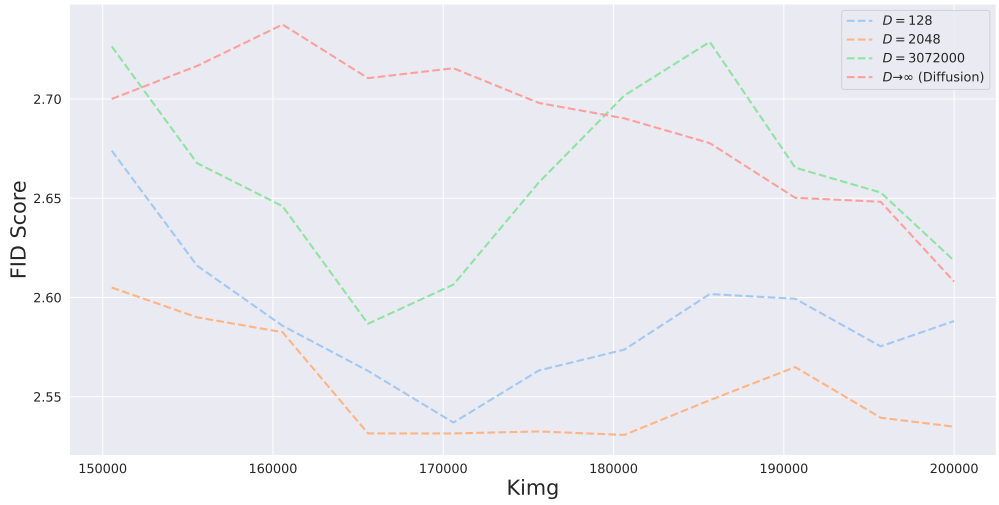
We further report the variation of FID scores in Table B.24 for the best checkpoint across different D values, by repeating the sampling process three times using different seeds. We observe that the standard deviation of FID is approximately in the range of $0.5\% \sim 1\%$ of the average FID, which is much smaller than the performance gain of $D = 128/2048$ in terms of Min or Average FID. Additionally, in Table 7.1 and Table 8.2 in the main text, we can see that the median $D = 128/2048$ consistently improves over the baseline ($D = \infty$) when using the Top-3 Average FID of checkpoints as a metric.

Table B.24: Min, Average and standard deviation of FID on CIFAR-10 using three different sets of random seeds for sampling

	Min FID ↓	Average FID ↓	Standard deviation
$D = 2048$	1.92	1.94	0.02
$D = 2048$	1.91	1.92	0.01
$D \to \infty$ [27]	1.98	2.00	0.02



(a) w/o moving average



(b) w/ moving average

Figure B.34: FID score in the training course when varying D, **(a)** w/o and **(b)** w/ moving average.

Experiments for Robustness

Controlled experiments with α In the controlled noise setting, we inject noise into the intermediate point $\mathbf{x}_r$ in each of the 35 ODE steps by $\mathbf{x}_r = \mathbf{x}_r + \alpha\boldsymbol{\epsilon}_r$ where $\boldsymbol{\epsilon}_r \sim \mathcal{N}(\mathbf{0}, r/\sqrt{D}\boldsymbol{I})$. Since p_r has roughly the same phase as $p_{\sigma=r/\sqrt{D}}$ in diffusion models, we pick $r/\sqrt{D}$ standard deviation of $\boldsymbol{\epsilon}_r$ when the intermediate step is r.

Post-training quantization In the post-training quantization experiments on CIFAR-10, we quantize the weights of convolutional layers excluding the 32×32 layers, as we empirically observe that these input/output layers are more critical for sample quality.

B.6.3 Extra Experiments

Stable Target Field

[23] propose a Stable Target Field objective for training the diffusion models:

$$\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}) \approx \mathbb{E}_{\mathbf{y}_1 \sim p_{0|t}(\cdot|\mathbf{x})} \mathbb{E}_{\{\mathbf{y}_i\}_{i=2}^n \sim p^{n-1}} \left[\sum_{k=1}^n \frac{p_{t|0}(\mathbf{x}|\mathbf{y}_k)}{\sum_j p_{t|0}(\mathbf{x}|\mathbf{y}_j)} \nabla_{\mathbf{x}} \log p_{t|0}(\mathbf{x}|\mathbf{y}_k) \right]$$

where they sample a large batch of samples $\{\mathbf{y}_i\}_{i=2}^n$ from the data distribution to approximate the score function at $\mathbf{x}$. They show that the new target can enhance the stability of converged models in different runs/seeds. PFGM++ can be trained in a similar fashion by replacing the target $\frac{\mathbf{x}-\mathbf{y}}{r/\sqrt{D}}$ in perturbation-based objective (Equation 8.4) with

$$\begin{aligned}
&\frac{1}{r/\sqrt{D}} \left(\mathbf{x} - \mathbb{E}_{p_{0|r}(\mathbf{y}|\mathbf{x})}[\mathbf{y}] \right) \approx \\
&\quad \frac{1}{r/\sqrt{D}} \left(\mathbf{x} - \mathbb{E}_{\mathbf{y}_1 \sim p_{0|r}(\cdot|\mathbf{x})} \mathbb{E}_{\{\mathbf{y}_i\}_{i=2}^n \sim p^{n-1}} \left[\sum_{k=1}^n \frac{1/(\|\mathbf{x}-\mathbf{y}_k\|_2^2 + r^2)^{\frac{N+D}{2}}}{\sum_j 1/(\|\mathbf{x}-\mathbf{y}_j\|_2^2 + r^2)^{\frac{N+D}{2}}} \mathbf{y}_k \right] \right)
\end{aligned}$$

When $n = 1$, the new target reduces to the original target. Similar to [23], one can show that the bias of the new target together with its trace-of-covariance shrinks to zero as we increase the size of the large batch. This new target can alleviate the variations between random seeds. With the new STF-style target, Table B.25 shows that when setting $D = 3072000 \gg N = 3072$,

the model obtains the same FID score as the diffusion models (EDM [27]). It aligns with the theoretical results in Sec 8.3, which states that PFGM++ recover the diffusion model when $D \to \infty$.

Table B.25: FID and NFE on CIFAR-10, using the Stable Target Field [23] in training objective.

	FID ↓	NFE ↓
$D = 3072000$	1.90	35
$D \to \infty$ [27]	1.90	35

Toy Dataset

In this section, we construct a 1000-dimensional toy dataset to systematically investigate the behaviors of models with different D values. We synthesize the data in three steps: first, we randomly sample the data $\mathbf{y}$ from a 10-dimensional Gaussian mixture $\frac{1}{2}\mathcal{N}(\mathbf{1}, 0.2^2 * \boldsymbol{I}_{10\times 10}) + \frac{1}{2}\mathcal{N}(-\mathbf{1}, 0.2^2 * \boldsymbol{I}_{10\times 10})$. Next, we map the 10-dimensional data to 1000-dimensional space using a random matrix $W \in \mathbb{R}^{1000\times 10}$: $\hat{\mathbf{y}} = W\mathbf{y}$. The entries in W are i.i.d sampled from standard normal distribution. Finally, we perturbed the data with a small Gaussian noise: $\mathbf{x} = \hat{\mathbf{y}} + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{1}, 0.01^2 * \boldsymbol{I}_{1000\times 1000})$. The synthetic dataset contains 2000 data points sampled using this procedure.

We design a four-layer UNet architecture, with widths corresponding to *data dimension—latent dimension—latent dimension—latent dimension—data dimension*. The latent dimension directly controls the capacity of the neural network. We also incorporate the residual connection, time-embedding and preconditioning techniques in EDM [27].

We examine the generated samples when varying D and the latent dimension. We visualize the first two coordinates $(\mathbf{x}_0, \mathbf{x}_1)$ of the true data (Figure B.35) and generated data (Figure B.36) for illustration. In Figure B.36, we show that when the latent dimension is set to 4, both the $D = 100$ and $D = \infty$ (diffusion model) fail to recover the data distribution, while model with intermediate $D = 1000$ well captures the underlying data distribution. On weaker architecture (smaller latent dimension), the non-robustness of large D and the non-rigidity of small D would be amplified. It corroborates the arguments that median Ds

better balance the robustness and rigidity. As we enlarge the neural network capacity by increasing the latent dimension to 32, all the models with different Ds faithfully recover the data distribution. For quantitative comparison, in Table B.26 we report the maximum mean discrepancy between the generated data and the true data for different models. We exclude the $D = 1$ case (PFGM) since the perturbation kernel is extremely heavy-tailed in 1000-dimensional space, preventing the use of the perturbation-based objective.

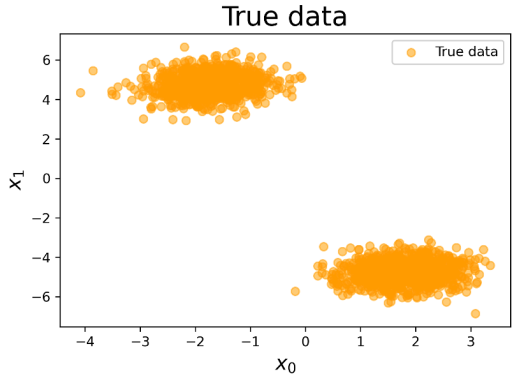


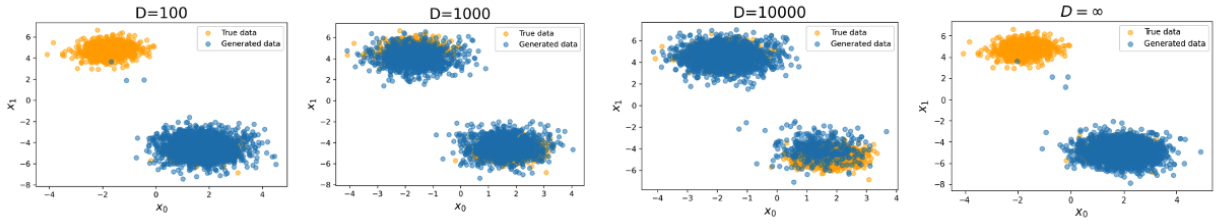
Figure B.35: Visualization of the first two coordinates $(\mathbf{x}_0, \mathbf{x}_1)$ for the 1000-dimensional synthetic data.

Table B.26: Maximum mean discrepancy between the generated data and the true data.

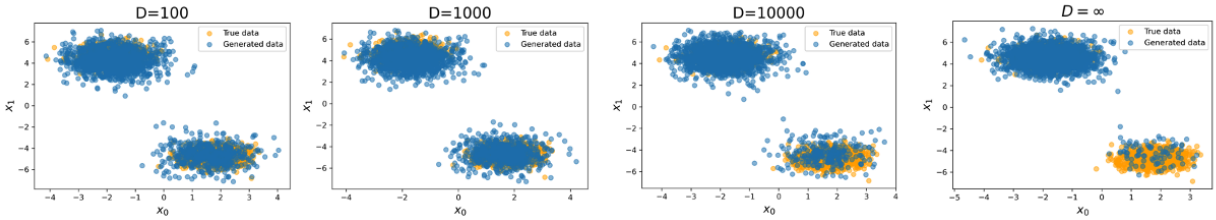
	$D = 100$	$D = 1000$	$D = 10000$	$D = \infty$
Latent Dimension $= 4$	1.75	**0.17**	0.79	1.82
Latent Dimension $= 8$	0.33	**0.16**	0.43	1.46
Latent Dimension $= 32$	0.12	**0.01**	0.14	0.07

Extended CIFAR-10 Samples when varying α

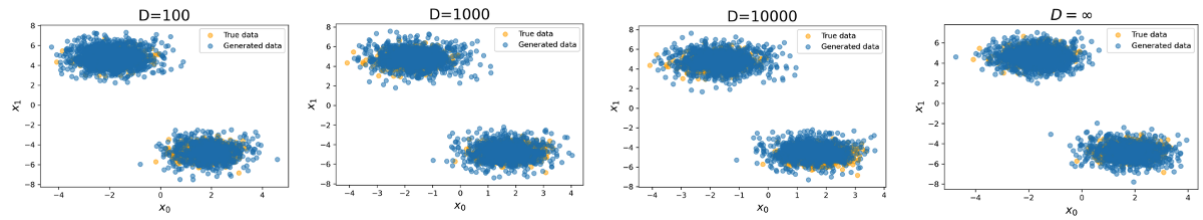
To see how the sample quality varies with α, we visualize the generative samples of models trained with $D \in \{64, 128, 2048\}$ and $D \to \infty$. We pick $\alpha \in \{0, 0.1, 0.2\}$. Figure B.37 shows that the smaller Ds produce better samples compared to larger D. Diffusion models ($D \to \infty$) generate noisy images that appear to be out of the data distribution when α=0.2, in contrast to the clean images by $D = 64, 128$.



(a) Latent dimension = 4

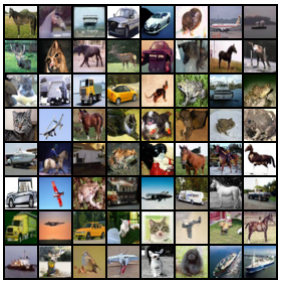


(b) Latent dimension = 8

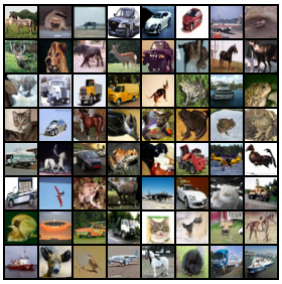


(c) Latent dimension = 32

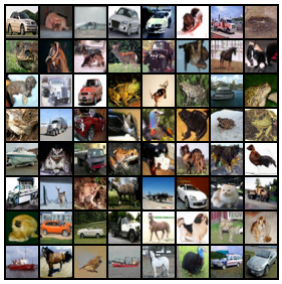
Figure B.36: Visualization of the first two coordinates $(\mathbf{x}_0, \mathbf{x}_1)$ for the generated data (blue) versus true data (orange). From the top row to the bottom row: the latent dimension of the neural network is set to 4 (a), 8 (b), and 32 (c).



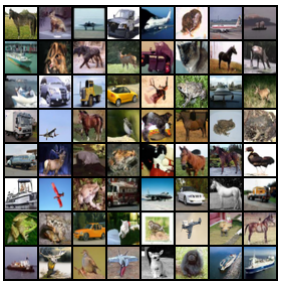
(a)
D=64, α=0 (FID=1.96)



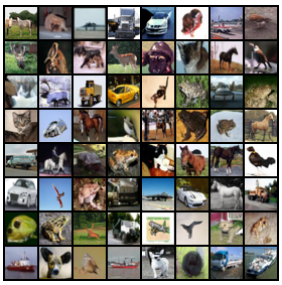
(b)
D=64, α=0.1 (FID=1.97)



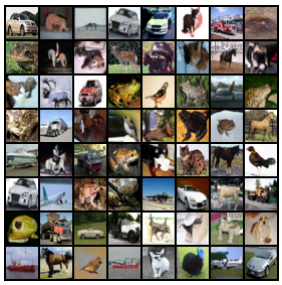
(c)
D=64, α=0.2 (FID=2.07)



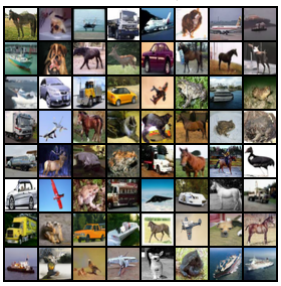
(d)
D=128, α=0 (FID=1.92)



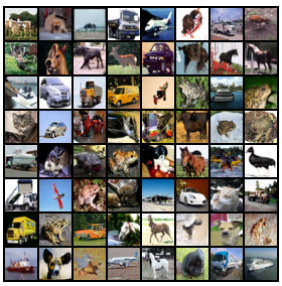
(e)
D=128, α=0.1 (FID=1.95)



(f)
D=128, α=0.2 (FID=2.19)

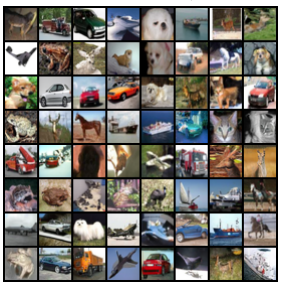


(g)
D=2048, α=0 (FID=1.92)

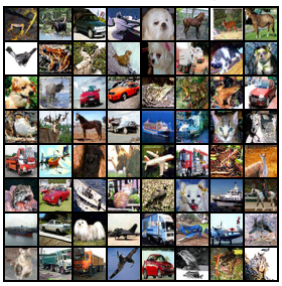


(h)
D=2048, α=0.1 (FID=1.95)

(i)
D=2048, α=0.2 (FID=2.19)



(j)
$D\to\infty$, α=0 (FID=1.98)



(k)
$D\to\infty$, α=0.1 (FID=9.27)

(l)
$D\to\infty$, α=0.2 (FID=92.41)

Figure B.37: Generated samples on CIFAR-10 with varied hyper-parameter for noise injection (α). Images from top to bottom rows are produced by models trained with $D = 64/128/2048/\infty$. We use the same random seeds for finite Ds during image generation.

(a) $D = 128$ (FID=2.43)

(b) EDM ($D \to \infty$) (FID=2.53)

Figure B.38: Generated images on FFHQ 64×64 dataset, by **(left)** $D = 128$ and **(right)** EDM ($D \to \infty$).

Extended FFHQ Samples

In Figure B.38, we provide samples generated by the $D = 128$ case and EDM (the $D \to \infty$ case).

Appendix C

Code

Below, we provide open-source implementations of the methods discussed in this thesis, as well as pre-trained checkpoints.

- Chapter 3: https://github.com/Newbeeer/stf
- Chapter 5: https://github.com/Newbeeer/diffusion_restart_sampling
- Chapter 6: https://github.com/gcorso/particle-guidance
- Chapter 7: https://github.com/Newbeeer/Poisson_flow
- Chapter 8: https://github.com/Newbeeer/pfgmpp

References

[1] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *ArXiv*, vol. abs/2005.14165, 2020.

[2] C. Saharia, W. Chan, S. Saxena, *et al.*, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," *arXiv preprint arXiv:2204.06125*, 2022.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[5] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv:2209.14988*, 2022.

[6] H. Jun and A. Nichol, "Shap-e: Generating conditional 3d implicit functions," *arXiv preprint arXiv:2305.02463*, 2023.

[7] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," 2022. arXiv: 2204.03458 [cs.CV].

[8] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, *et al.*, "Make-a-video: Text-to-video generation without text-video data," *arXiv:2209.14792*, 2022.

[9] V. N. Vapni, "The nature of statistical learning theory," in *Statistics for Engineering and Information Science*, 2000. URL: https://api.semanticscholar.org/CorpusID:7138354.

[10] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*, PMLR, 2015, pp. 2256–2265.

[11] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[13] Z. Liu, D. Luo, Y. Xu, T. S. Jaakkola, and M. Tegmark, "Genphys: From physical processes to generative models," *ArXiv*, vol. abs/2304.02637, 2023.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[15] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *NIPS*, 2016.

[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *NeurIPS*, 2018.

[17] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *ArXiv*, vol. abs/1605.08803, 2017.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2013.

[19] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[20] Y. LeCun, S. Chopra, R. Hadsell, A. Ranzato, and F. J. Huang, "A tutorial on energy-based learning," 2006.

[21] Y. Du and I. Mordatch, "Implicit generation and generalization in energy-based models," *ArXiv*, vol. abs/1903.08689, 2019.

[22] B. D. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[23] Y. Xu, S. Tong, and T. S. Jaakkola, "Stable target field for reduced variance score estimation in diffusion models," in *International Conference on Learning Representations*, 2023.

[24] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," in *International Conference on Learning Representations (ICLR)*, 2021.

[25] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *ArXiv*, vol. abs/2010.02502, 2020.

[26] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," *arXiv preprint arXiv:2206.00927*, 2022.

[27] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *ArXiv*, vol. abs/2206.00364, 2022.

[28] A. Jolicoeur-Martineau, K. Li, R. Piche-Taillefer, T. Kachman, and I. Mitliagkas, "Gotta go fast when generating data with score-based models," *ArXiv*, vol. abs/2105.14080, 2021.

[29] Y. Xu, M. Deng, X. Cheng, Y. Tian, Z. Liu, and T. Jaakkola, "Restart sampling for improving generative processes," *ArXiv*, vol. abs/2306.14878, 2023.

[30] Y. Xu, Z. Liu, M. Tegmark, and T. Jaakkola, "Poisson flow generative models," *ArXiv*, vol. abs/2209.11178, 2022.

[31] Y. Xu, Z. Liu, Y. Tian, S. Tong, M. Tegmark, and T. Jaakkola, "Pfgm++: Unlocking the potential of physics-inspired generative models," in *International Conference on Machine Learning*, 2023.

[32] Y. Xu, G. Corso, T. Jaakkola, A. Vahdat, and K. Kreis, "Disco-diff: Enhancing continuous diffusion models with discrete latents," 2024.

[33] G. Corso, Y. Xu, V. D. Bortoli, R. Barzilay, and T. Jaakkola, "Particle guidance: Non-i.i.d. diverse sampling with diffusion models," *ArXiv*, vol. abs/2310.13102, 2023. URL: https://api.semanticscholar.org/CorpusID:264405842.

[34] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *ArXiv*, vol. abs/2011.13456, 2021.

[35] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021. URL: https://openreview.net/forum?id=PxTIG12RRHS.

[36] Y. Song, C. Durkan, I. Murray, and S. Ermon, "Maximum likelihood training of score-based diffusion models," in *Neural Information Processing Systems*, 2021. URL: https://api.semanticscholar.org/CorpusID:235352469.

[37] Y. Song, J. N. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *ArXiv*, vol. abs/2011.13456, 2020.

[38] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Computation*, vol. 23, pp. 1661–1674, 2011.

[39] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[40] S. Li, L. Liu, Z. Chai, R. Li, and X. Tan, "Era-solver: Error-robust adams solver for fast sampling of diffusion probabilistic models," *arXiv preprint arXiv:2301.12935*, 2023.

[41] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *NIPS*, 2017.

[42] A. B. Owen, *Monte Carlo theory, methods and examples.* 2013.

[43] V. Elvira and L. Martino, "Advances in importance sampling," *Wiley StatsRef: Statistics Reference Online*, 2021.

[44] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion GANs," in *International Conference on Learning Representations*, 2022. URL: https://openreview.net/forum?id=JprM0p-q0Co.

[45] C. Wang, X. Chen, A. Smola, and E. Xing, "Variance reduction for stochastic gradient optimization," in *NIPS*, 2013.

[46] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[47] T. Hesterberg, "Weighted average importance sampling and defensive mixture distributions," *Technometrics*, vol. 37, pp. 185–194, 1995.

[48] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *ArXiv*, vol. abs/2006.06676, 2020.

[49] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *ArXiv*, vol. abs/2006.09011, 2020.

[50] Y. Xu, Z. Liu, M. Tegmark, and T. Jaakkola, "Poisson flow generative models," *Advances in Neural Information Processing Systems*, 2022.

[51] S. Yang, P. Luo, C. C. Loy, and X. Tang, "From facial parts responses to face detection: A deep learning approach," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3676–3684, 2015.

[52] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *ArXiv*, vol. abs/1710.10196, 2018.

[53] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, pp. 19–26, 1980.

[54] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *ArXiv*, vol. abs/2010.02502, 2021.

[55] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *ArXiv*, vol. abs/1606.03498, 2016.

[56] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in neural information processing systems*, vol. 27, 2014.

[57] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems*, vol. 26, 2013.

[58] J. Choi, J. Lee, C. Shin, S. Kim, H. Kim, and S. Yoon, "Perception prioritized training of diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 472–11 481.

[59] A. Grover, J. Song, A. Kapoor, K. Tran, A. Agarwal, E. J. Horvitz, and S. Ermon, "Bias correction of learned generative models using likelihood-free importance weighting," *Advances in neural information processing systems*, vol. 32, 2019.

[60] A. Swaminathan and T. Joachims, "The self-normalized estimator for counterfactual learning," in *NIPS*, 2015.

[61] A. M. Metelli, M. Papini, F. Faccio, and M. Restelli, "Policy optimization via importance sampling," in *NeurIPS*, 2018.

[62] J. Bornschein and Y. Bengio, "Reweighted wake-sleep," *arXiv preprint arXiv:1406.2751*, 2014.

[63] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," *arXiv preprint arXiv:1509.00519*, 2015.

[64] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "wake-sleep" algorithm for unsupervised neural networks," *Science*, vol. 268, no. 5214, pp. 1158–1161, 1995.

[65] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[66] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*, PMLR, 2014, pp. 1278–1286.

[67] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching.," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[68] Z. Wang, S. Cheng, L. Yueru, J. Zhu, and B. Zhang, "A wasserstein minimum velocity approach to learning unnormalized models," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 3728–3738.

[69] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[70] F. Bao, C. Li, J. Sun, and J. Zhu, "Why are conditional generative models better than unconditional ones?" *arXiv preprint arXiv:2212.00362*, 2022.

[71] V. T. Hu, D. W. Zhang, Y. M. Asano, G. J. Burghouts, and C. G. M. Snoek, "Self-guided diffusion models," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[72] W. Harvey and F. Wood, "Visual chain-of-thought diffusion models," *arXiv preprint arXiv:2303.16187*, 2023.

[73] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[74] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[75] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman, "Maskgit: Masked generative image transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[76] J. Yu, Y. Xu, J. Y. Koh, *et al.*, "Scaling autoregressive models for content-rich text-to-image generation," *Transactions on Machine Learning Research (TMLR)*, 2022.

[77] P. Pernias, D. Rampas, M. L. Richter, C. J. Pal, and M. Aubreville, "Wuerstchen: An efficient architecture for large-scale text-to-image diffusion models," *arXiv preprint arXiv:2306.00637*, 2023.

[78] H. Chang, H. Zhang, J. Barber, *et al.*, "Muse: Text-to-image generation via masked generative transformers," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[79] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[80] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[81] Y. Xu, Z. Liu, M. Tegmark, and T. Jaakkola, "Poisson flow generative models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[82] Y. Xu, Z. Liu, Y. Tian, S. Tong, M. Tegmark, and T. Jaakkola, "PFGM++: Unlocking the potential of physics-inspired generative models," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[83] E. Jang, S. S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *ArXiv*, vol. abs/1611.01144, 2016.

[84] J. Ho and T. Salimans, "Classifier-Free Diffusion Guidance," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[85] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *The International Conference on Learning Representations*, 2014.

[86] A. van den Oord, O. Vinyals, and k. kavukcuoglu koray, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[87] J. T. Rolfe, "Discrete variational autoencoders," in *International Conference on Learning Representations*, 2017.

[88] C.-W. Huang, J. H. Lim, and A. Courville, "A variational perspective on diffusion-based generative models and score matching," in *Neural Information Processing Systems (NeurIPS)*, 2021.

[89] D. P. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," in *Advances in Neural Information Processing Systems*, 2021.

[90] P. Dhariwal and A. Q. Nichol, "Diffusion Models Beat GANs on Image Synthesis," in *Advances in Neural Information Processing Systems*, 2021.

[91] D. P. Kingma and R. Gao, "Understanding diffusion objectives as the ELBO with simple data augmentation," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[92] J. Ho, W. Chan, C. Saharia, *et al.*, "Imagen Video: High Definition Video Generation with Diffusion Models," *arXiv preprint arXiv:2210.02303*, 2022.

[93] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[94] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[95] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[96] T. Li, D. Katabi, and K. He, "Self-conditioned image generation via generating representations," *arXiv preprint arXiv:2312.03701*, 2023.

[97] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[98] Y. Wang, Y. Schiff, A. Gokaslan, W. Pan, F. Wang, C. De Sa, and V. Kuleshov, "InfoDiffusion: Representation learning using information maximizing diffusion models," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[99] A. Casanova, M. Careil, J. Verbeek, M. Drozdzal, and A. Romero-Soriano, "Instance-conditioned gan," in *Neural Information Processing Systems*, 2021.

[100] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *ArXiv*, vol. abs/1809.11096, 2018.

[101] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," *ArXiv*, vol. abs/2102.09672, 2021.

[102] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *J. Mach. Learn. Res.*, vol. 23, 47:1–47:33, 2021.

[103] A. Jabri, D. J. Fleet, and T. Chen, "Scalable adaptive computation for iterative generation," in *International Conference on Machine Learning*, 2022.

[104] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon, "Consistency trajectory models: Learning probability flow ode trajectory of diffusion," *ArXiv*, vol. abs/2310.02279, 2023.

[105] A. Sauer, K. Schwarz, and A. Geiger, "Stylegan-xl: Scaling stylegan to large diverse datasets," *ACM SIGGRAPH 2022 Conference Proceedings*, 2022.

[106] E. Hoogeboom, J. Heek, and T. Salimans, "Simple Diffusion: End-to-End Diffusion for High Resolution Images," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[107] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola, "Diffdock: Diffusion steps, twists, and turns for molecular docking," *International Conference on Learning Representations (ICLR)*, 2023.

[108] M. Geiger and T. Smidt, "E3nn: Euclidean neural networks," *arXiv preprint arXiv:2207.09453*, 2022.

[109] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems*, 2017.

[110] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, "Structured denoising diffusion models in discrete state-spaces," in *Advances in Neural Information Processing Systems*, 2021.

[111] A. Campbell, J. Benton, V. D. Bortoli, T. Rainforth, G. Deligiannidis, and A. Doucet, "A continuous time framework for discrete denoising models," in *Advances in Neural Information Processing Systems*, 2022.

[112] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[113] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, 2017.

[114] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[115] Y. Balaji, S. Nah, X. Huang, *et al.*, "eDiff-I: Text-to-Image Diffusion Models with Ensemble of Expert Denoisers," *arXiv preprint arXiv:2211.01324*, 2022.

[116] G. Corso, A. Deng, N. Polizzi, R. Barzilay, and T. Jaakkola, "The discovery of binding modes requires rethinking docking generalization," in *International Conference on Learning Representations*, 2024.

[117] H. Stärk, O. Ganea, L. Pattanaik, R. Barzilay, and T. Jaakkola, "Equibind: Geometric deep learning for drug binding structure prediction," in *International Conference on Machine Learning*, 2022.

[118] Z. Liu, M. Su, L. Han, J. Liu, Q. Yang, Y. Li, and R. Wang, "Forging the basis for developing protein–ligand interaction scoring functions," *Accounts of Chemical Research*, 2017.

[119] A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, R. Meli, M. Ragoza, J. Sunseri, and D. R. Koes, "Gnina 1.0: Molecular docking with deep learning," *Journal of cheminformatics*, 2021.

[120] D. R. Koes, M. P. Baumgartner, and C. J. Camacho, "Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise," *Journal of chemical information and modeling*, 2013.

[121] T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, and J. L. Banks, "Glide: A new approach for rapid, accurate docking and scoring. 2. enrichment factors in database screening," *Journal of medicinal chemistry*, 2004.

[122] W. Lu, Q. Wu, J. Zhang, J. Rao, C. Li, and S. Zheng, "Tankbind: Trigonometry-aware neural networks for drug-protein binding structure prediction," *Advances in neural information processing systems*, 2022.

[123] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[124] T. Dockhorn, A. Vahdat, and K. Kreis, "Score-based generative modeling with critically-damped langevin diffusion," in *International Conference on Learning Representations (ICLR)*, 2022.

[125] T. Dockhorn, A. Vahdat, and K. Kreis, "Genie: Higher-order denoising diffusion solvers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[126] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis," *arXiv preprint arXiv:2307.01952*, 2023.

[127] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis, "Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[128] U. Singer, A. Polyak, T. Hayes, *et al.*, "Make-A-Video: Text-to-Video Generation without Text-Video Data," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[129] S. Ge, S. Nah, G. Liu, T. Poon, A. Tao, B. Catanzaro, D. Jacobs, J.-B. Huang, M.-Y. Liu, and Y. Balaji, "Preserve Your Own Correlation: A Noise Prior for Video Diffusion Models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[130] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen, *Point-E: A System for Generating 3D Point Clouds from Complex Prompts*, 2022.

[131] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "LION: Latent Point Diffusion Models for 3D Shape Generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[132] S. W. Kim, B. Brown, K. Yin, K. Kreis, K. Schwarz, D. Li, R. Rombach, A. Torralba, and S. Fidler, "NeuralField-LDM: Scene Generation with Hierarchical Latent Diffusion Models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[133] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "DreamFusion: Text-to-3D using 2D Diffusion," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[134] K. Schwarz, S. W. Kim, J. Gao, S. Fidler, A. Geiger, and K. Kreis, "WildFusion: Learning 3D-Aware Latent Diffusion Models in View Space," *arXiv preprint arXiv:2311.13570*, 2023.

[135] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot One Image to 3D Object," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[136] U. Singer, S. Sheynin, A. Polyak, *et al.*, "Text-to-4D Dynamic Scene Generation," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[137] H. Ling, S. W. Kim, A. Torralba, S. Fidler, and K. Kreis, "Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models," *arXiv preprint arXiv:2312.13763*, 2023.

[138] S. Bahmani, I. Skorokhodov, V. Rong, G. Wetzstein, L. Guibas, P. Wonka, S. Tulyakov, J. J. Park, A. Tagliasacchi, and D. B. Lindell, "4d-fy: Text-to-4d generation using hybrid score distillation sampling," *arXiv preprint arXiv:2311.17984*, 2023.

[139] Y. Zheng, X. Li, K. Nagano, S. Liu, K. Kreis, O. Hilliges, and S. D. Mello, "A unified approach for text- and image-guided 4d scene generation," *arXiv preprint arXiv:2311.16854*, 2023.

[140] J. Yim, B. L. Trippe, V. D. Bortoli, E. Mathieu, A. Doucet, R. Barzilay, and T. Jaakkola, "Se(3) diffusion model with application to protein backbone generation," *arXiv preprint arXiv:2302.02277*, 2023.

[141] J. Ingraham, M. Baranov, Z. Costello, *et al.*, "Illuminating protein space with a programmable generative model," *Nature*, vol. 623, pp. 1070–1078, 2023.

[142] J. L. Watson, D. Juergens, N. R. Bennett, *et al.*, "De novo design of protein structure and function with rfdiffusion," *Nature*, vol. 620, pp. 1089–1100, 2023.

[143] A. Vahdat, K. Kreis, and J. Kautz, "Score-based Generative Modeling in Latent Space," in *Neural Information Processing Systems (NeurIPS)*, 2021.

[144] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, 2009.

[145] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 599–619.

[146] A. Vahdat, E. Andriyash, and W. G. Macready, "Dvae#: Discrete variational autoencoders with relaxed Boltzmann priors," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[147] A. Vahdat, W. Macready, Z. Bian, A. Khoshaman, and E. Andriyash, "DVAE++: Discrete variational autoencoders with overlapping transformations," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[148] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.

[149] M. R. Masters, A. H. Mahmoud, and M. A. Lill, "Fusiondock: Physics-informed diffusion model for molecular docking," *ICML Workshop on Computational Biology*, 2023.

[150] M. Plainer, M. Toth, S. Dobers, H. Stärk, G. Corso, C. Marquet, and R. Barzilay, "Diffdock-pocket: Diffusion for pocket-level docking with sidechain flexibility," in *NeurIPS 2023 Workshop on New Frontiers of AI for Drug Discovery and Development*, 2023.

[151] H. Guo, S. Liu, H. Mingdi, Y. Lou, and B. Jing, "Diffdock-site: A novel paradigm for enhanced protein-ligand predictions through binding site identification," in *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

[152] R. Krivák and D. Hoksza, "P2rank: Machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure," *Journal of cheminformatics*, vol. 10, pp. 1–12, 2018.

[153] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10 674–10 685, 2021.

[154] U. M. Ascher and L. R. Petzold, "Computer methods for ordinary differential equations and differential-algebraic equations," in *SIAM*, 1998.

[155] A. S. Dalalyan and A. Karagulyan, "User-friendly guarantees for the langevin monte carlo with inaccurate gradient," *Stochastic Processes and their Applications*, vol. 129, no. 12, pp. 5278–5311, 2019.

[156] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[157] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Citeseer*, 2009.

[158] C. Schuhmann, R. Beaumont, R. Vencu, *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *ArXiv*, vol. abs/2210.08402, 2022.

[159] J. Ho, "Classifier-free diffusion guidance," *ArXiv*, vol. abs/2207.12598, 2022.

[160] C. Saharia, W. Chan, S. Saxena, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *ArXiv*, vol. abs/2205.11487, 2022.

[161] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," in *International Conference on Machine Learning*, 2021.

[162] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014.

[163] J. Hessel, A. Holtzman, M. Forbes, R. J. L. Bras, and Y. Choi, "Clipscore: A reference-free evaluation metric for image captioning," in *Conference on Empirical Methods in Natural Language Processing*, 2021.

[164] G. Ilharco, M. Wortsman, R. Wightman, *et al.*, "Openclip," version 0.1, *Zenodo*, 2021. DOI: 10.5281/zenodo.5143773. URL: https://doi.org/10.5281/zenodo.5143773.

[165] L.-A. Team, *Laion-aesthetics predictor v2*, https://github.com/christophschuhmann/improved-aesthetic-predictor, 2022.

[166] C. Meng, R. Gao, D. P. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," *ArXiv*, vol. abs/2210.03142, 2022.

[167] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *ArXiv*, vol. abs/2006.11239, 2020.

[168] B. Jing, G. Corso, J. Chang, R. Barzilay, and T. Jaakkola, "Torsional diffusion for molecular conformer generation," *arXiv preprint arXiv:2206.01729*, 2022.

[169] G. Hummer and J. Köfinger, "Bayesian ensemble refinement by replica simulations and reweighting," *The Journal of chemical physics*, 2015.

[170] A. P. Pasarkar, G. M. Bencomo, S. Olsson, and A. B. Dieng, "Vendi sampling for molecular simulations: Diversity as a force for faster convergence and better exploration," *The Journal of Chemical Physics*, vol. 159, no. 14, 2023.

[171] A. Laio and M. Parrinello, "Escaping free-energy minima," *Proceedings of the national academy of sciences*, 2002.

[172] A. Barducci, G. Bussi, and M. Parrinello, "Well-tempered metadynamics: A smoothly converging and tunable free-energy method," *Physical review letters*, 2008.

[173] S. Chewi, C. Lu, K. Ahn, X. Cheng, T. L. Gouic, and P. Rigollet, "Optimal dimension dependence of the metropolis-adjusted langevin algorithm," *ArXiv*, vol. abs/2012.12810, 2020.

[174] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," *Advances in neural information processing systems*, vol. 29, 2016.

[175] L. K. Wenliang and H. Kanagawa, "Blindness of score-based methods to isolated components and mixing proportions," 2020.

[176] J. Zhuo, C. Liu, J. Shi, J. Zhu, N. Chen, and B. Zhang, "Message passing stein variational gradient descent," in *International Conference on Machine Learning*, PMLR, 2018.

[177] J. Zhang, R. Zhang, L. Carin, and C. Chen, "Stochastic particle-optimization sampling and the non-asymptotic convergence theory," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020.

[178] F. D'Angelo and V. Fortuin, "Annealed stein variational gradient descent," *arXiv preprint arXiv:2101.09815*, 2021.

[179] W.-C. Chang, C.-L. Li, Y. Mroueh, and Y. Yang, "Kernel stein generative modeling," *arXiv preprint arXiv:2007.03074*, 2020.

[180] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[181] Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. S. Grathwohl, "Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc," in *International Conference on Machine Learning*, PMLR, 2023.

[182] J. Benton, Y. Shi, V. De Bortoli, G. Deligiannidis, and A. Doucet, "From denoising diffusions to denoising markov models," *arXiv preprint arXiv:2211.03595*, 2022.

[183] J. Köhler, L. Klein, and F. Noé, "Equivariant flows: Exact likelihood generative learning for symmetric densities," in *International conference on machine learning*, PMLR, 2020, pp. 5361–5370.

[184] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "Geodiff: A geometric diffusion model for molecular conformation generation," in *International Conference on Learning Representations*, 2021.

[185] Y. Xu, M. Deng, X. Cheng, Y. Tian, Z. Liu, and T. Jaakkola, "Restart sampling for improving generative processes," *arXiv preprint arXiv:2306.14878*, 2023.

[186] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[187] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein, "Diffusion art or digital forgery? investigating data replication in diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[188] S. Axelrod and R. Gomez-Bombarelli, "Geom, energy-annotated molecular conformations for property prediction and molecular generation," *Scientific Data*, 2022.

[189] O. Ganea, L. Pattanaik, C. Coley, R. Barzilay, K. Jensen, W. Green, and T. Jaakkola, "Geomol: Torsional geometric generation of molecular 3d conformer ensembles," *Advances in Neural Information Processing Systems*, 2021.

[190] I. Csiszár, "I-divergence geometry of probability distributions and minimization problems," *The annals of probability*, pp. 146–158, 1975.

[191] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *ArXiv*, vol. abs/1806.07366, 2018.

[192] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. K. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *ArXiv*, vol. abs/1810.01367, 2019.

[193] D. J. Griffiths, *Introduction to electrodynamics*, 2005.

[194] H. Goldstein, C. Poole, and J. Safko, *Classical mechanics*, 2002.

[195] H. Risken, "Fokker-planck equation," 1984.

[196] R. T. Q. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen, "Residual flows for invertible generative modeling," *ArXiv*, vol. abs/1906.02735, 2019.

[197] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *ArXiv*, vol. abs/1907.05600, 2019.

[198] F. S. Lawrence, "Some practical runge-kutta formulas," *Mathematics of Computation*, vol. 46, pp. 135–150, 1986.

[199] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[200] K. Lee, H. Chang, L. Jiang, H. Zhang, Z. Tu, and C. Liu, "Vitgan: Training gans with vision transformers," *ArXiv*, vol. abs/2107.04589, 2021.

[201] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," 2009. URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[202] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *ArXiv*, vol. abs/1506.03365, 2015.

[203] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NIPS*, 2017.

[204] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *ArXiv*, vol. abs/1802.05957, 2018.

[205] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," *ArXiv*, vol. abs/1902.00275, 2019.

[206] C. Jarzynski, "Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach," *Physical Review E*, vol. 56, pp. 5018–5035, 1997.

[207] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *ArXiv*, vol. abs/2209.14988, 2022.

[208] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen, "Point-e: A system for generating 3d point clouds from complex prompts," *ArXiv*, vol. abs/2212.08751, 2022.

[209] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *arXiv: Computer Vision and Pattern Recognition*, 2015.

[210] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Neural Information Processing Systems*, 2018.

[211] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2018.

[212] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[213] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *ArXiv*, vol. abs/1511.06488, 2015.

[214] M. Tegmark, "Does the universe in fact contain almost no information?" *arXiv preprint quant-ph/9603008*, 1996.

[215] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?" *Journal of Statistical Physics*, vol. 168, pp. 1223–1247, 2017.

[216] G. Parisi, "Correlation functions and computer simulations (ii)," *Nuclear Physics*, vol. 205, pp. 337–344, 1981.

[217] A. Hyvärinen, "Some extensions of score matching," *Comput. Stat. Data Anal.*, vol. 51, pp. 2499–2512, 2007.

[218] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "Lion: Latent point diffusion models for 3d shape generation," *ArXiv*, vol. abs/2210.06978, 2022.

[219] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling, "Equivariant diffusion for molecule generation in 3d," *ArXiv*, vol. abs/2203.17003, 2022.

[220] S. Rissanen, M. Heinonen, and A. Solin, "Generative modelling with inverse heat dissipation," *ArXiv*, vol. abs/2206.13397, 2022.

[221] J. Kirkwood, *Mathematical physics with partial differential equations*. Academic Press, 2018.

[222] R. B. Guenther and J. W. Lee, *Partial differential equations of mathematical physics and integral equations*. Courier Corporation, 1996.

[223] R. Courant and D. Hilbert, *Methods of mathematical physics: partial differential equations*. John Wiley & Sons, 2008.

[224] Y. Mroueh and M. Rigotti, "Unbalanced sobolev descent," *ArXiv*, vol. abs/2009.14148, 2020.

[225] K. Fatras, T. S'ejourn'e, N. Courty, and R. Flamary, "Unbalanced minibatch optimal transport; applications to domain adaptation," *ArXiv*, vol. abs/2103.03606, 2021.

[226] Y. Lu, J. Lu, and J. Nolen, "Accelerating langevin sampling with birth-death," *ArXiv*, vol. abs/1905.09863, 2019.

[227] R. M. Martin, L. Reining, and D. M. Ceperley, *Interacting Electrons: Theory and Computational Approaches*. Cambridge University Press, 2016. DOI: 10.1017/CBO9781139050807.

[228] Y. Lu, J. Lu, and J. Nolen, "Accelerating langevin sampling with birth-death," *arXiv preprint arXiv:1905.09863*, 2019.

[229] H. Soodak and M. S. Tiersten, "Wakes and waves in n dimensions," *American journal of physics*, vol. 61, no. 5, pp. 395–401, 1993.

[230] R. Aleixo and E. Capelas de Oliveira, "Green's function for the lossy wave equation," *Revista Brasileira de Ensino de Física*, vol. 30, pp. 1302.1–1302.5, Jan. 2008. DOI: 10.1590/S1806-11172008000100003.

[231] H. YUKAWA and S. SAKATA, "On the interaction of elementary particles ii," *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, vol. 19, pp. 1084–1093, 1937.

[232] D. J. Griffiths and D. F. Schroeter, *Introduction to quantum mechanics*. Cambridge university press, 2018.

[233] D. Aharonov, W. Van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," *SIAM review*, vol. 50, no. 4, pp. 755–787, 2008.

[234] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *CoRR*, vol. abs/1410.8516, 2014. URL: https://api.semanticscholar.org/CorpusID:13995862.

[235] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[236] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola, "Diffdock: Diffusion steps, twists, and turns for molecular docking," *arXiv preprint arXiv:2210.01776*, 2022.

[237] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *ArXiv*, vol. abs/2009.09761, 2020.

[238] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim, "Diff-tts: A denoising diffusion model for text-to-speech," *arXiv preprint arXiv:2104.01409*, 2021.

[239] J. Song, A. Vahdat, M. Mardani, and J. Kautz, "Pseudoinverse-guided diffusion models for inverse problems," in *International Conference on Learning Representations*, 2023. URL: https://openreview.net/forum?id=9_gsMA8MRKQ.

[240] Y. Song, L. Shen, L. Xing, and S. Ermon, "Solving inverse problems in medical imaging with score-based generative models," *arXiv preprint arXiv:2111.08005*, 2021.

[241] C. Huang, Z. Liu, S. Bai, L. Zhang, C. Xu, Z. Wang, Y. Xiang, and Y. Xiong, "Pf-abgen: A reliable and efficient antibody generator via poisson flow," in *ICLR 2023-Machine Learning for Drug Discovery workshop*, 2023.

[242] R. Ge, Y. He, C. Xia, Y. Chen, D. Zhang, and G. Wang, "Jccs-pfgm: A novel circle-supervision based poisson flow generative model for multiphase cect progressive low-dose reconstruction with joint condition," *arXiv preprint arXiv:2306.07824*, 2023.

[243] A. N. Borodin and P. Salminen, *Handbook of Brownian motion-facts and formulae*. Springer Science & Business Media, 2015.

[244] I. Karatzas and S. Shreve, *Brownian motion and stochastic calculus*. Springer Science & Business Media, 1991.

[245] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *Annals of Mathematical Statistics*, vol. 35, pp. 876–879, 1964. URL: https://api.semanticscholar.org/CorpusID:120846714.

[246] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "Mine: Mutual information neural estimation," *arXiv preprint arXiv:1801.04062*, 2018.

[247] J. Yim, B. L. Trippe, V. De Bortoli, E. Mathieu, A. Doucet, R. Barzilay, and T. Jaakkola, "Se(3) diffusion model with application to protein backbone generation," *arXiv preprint*, 2023.

[248] H. Ricardo, "A modern introduction to differential equations," 2002.

[249] M. V. (https://math.stackexchange.com/users/218419/mark-viola), *Fundamental solution for helmholtz equation in higher dimensions*, Mathematics Stack Exchange, URL:https://math.stackexchange.com/q/2255877 (version: 2017-04-30). eprint: https://math.stackexchange.com/q/2255877. URL: https://math.stackexchange.com/q/2255877.

[250] Wikipedia contributors, *Green's function — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Green%27s_function&oldid=1134903008, [Online; accessed 22-January-2023], 2023.

[251] O. Trott and A. J. Olson, "Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of computational chemistry*, 2010.

[252] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling

quantum interactions," *Advances in neural information processing systems*, vol. 30, 2017.

[253] G. Landrum, "Rdkit documentation," *Release*, 2013.

[254] M. A. Ketata, C. Laue, R. Mammadov, H. Stärk, M. Wu, G. Corso, C. Marquet, R. Barzilay, and T. S. Jaakkola, "Diffdock-pp: Rigid protein-protein docking with diffusion models," *arXiv preprint arXiv:2304.03889*, 2023.

[255] J. Geweke, "Antithetic acceleration of monte carlo integration in bayesian inference," *Journal of Econometrics*, vol. 38, no. 1-2, pp. 73–89, 1988.

[256] A. Kulesza, B. Taskar, *et al.*, "Determinantal point processes for machine learning," *Foundations and Trends® in Machine Learning*, vol. 5, no. 2–3, pp. 123–286, 2012.

[257] G. Corso, "Modeling molecular structures with intrinsic diffusion models," *arXiv preprint arXiv:2302.12255*, 2023.

[258] P. Pracht, F. Bohle, and S. Grimme, "Automated exploration of the low-energy chemical space with fast quantum chemical methods," *Physical Chemistry Chemical Physics*, 2020.

[259] P. C. Hawkins and A. Nicholls, "Conformer generation with omega: Learning from the data set and the analysis of failures," *Journal of chemical information and modeling*, 2012.

[260] C. Shi, S. Luo, M. Xu, and J. Tang, "Learning gradient fields for molecular conformation generation," in *International Conference on Machine Learning*, 2021.

[261] J. Ingraham, M. Baranov, Z. Costello, V. Frappier, A. Ismail, S. Tie, W. Wang, V. Xue, F. Obermeyer, A. Beam, *et al.*, "Illuminating protein space with a programmable generative model," *BioRxiv*, 2022.

[262] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[263] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.