IMPROVING DIFFUSION MODELS

EDEN CEN, GRADY KENIX, RUSSELL RAMSAY, LIUSI XU, AND DEJAN SLEPČEV*

CONTENTS

1. Generative Learning 1
2. Diffusion Models 2
2.1. Problem Statement 2
2.2. Fokker-Planck Equation 2
2.3. Backward Dynamics Derivation 3
2.4. Reverse Process 3
3. Algorithms and Performance Analysis 4
3.1. Explicit Score Implementation 4
3.2. Neural Score Matching 6
3.3. Two–Dimensional Score Matching 8
4. Improved Approaches 10
4.1. Explicit Solution Improvements 10
4.2. Neural Score Matching Improvements 16
4.3. Physics-Based Repulsion with Potentials 19
5. Conclusion 19
References 21
Appendix A. Implementation Details 22

1. GENERATIVE LEARNING

Generative learning represents a transformative approach in machine learning and artificial intelligence, focusing on models that learn the underlying distribution of data to generate new, synthetic samples. A prominent class of generative models is Generative Adversarial Networks (GANs), introduced by Goodfellow et al. (2014) [1], which involve a generator and discriminator in a minimax game. GANs have been particularly successful in realistic image synthesis and are foundational to modern generative modeling. Unlike traditional discriminative models, which primarily classify or predict based on input features, generative models are designed to generate new data points that resemble the training data. Key applications include image synthesis, molecular design, and anomaly detection. However, generative learning also presents fundamental challenges, such as mode collapse, where models generate only a limited variety of samples, and training instability, which is often sensitive to initialization and the complexity of the optimization landscape. Addressing these challenges is crucial for advancing the field. For instance, recent work has explored second-order optimization techniques to mitigate mode collapse and improve training stability (Durall et al., 2020) [2]. This study explores diffusion modeling frameworks and proposes improved approaches to overcome such limitations.

2. Diffusion Models

Diffusion models are a class of generative models that have achieved remarkable success in modeling complex data distributions, exhibiting state-of-the-art performance in image and audio synthesis. Recent advances in this field span theoretical, algorithmic, and architectural innovations. The research by Song et al. (2020) [3] introduces a unified framework for score-based generative modeling using stochastic differential equations (SDEs), which underpins the reverse diffusion processes we adopt in this study. Another research [4] focuses on analyzing the theoretical convergence of denoising diffusion models when the data lies on a low-dimensional manifold rather than in full space. To address the computational inefficiencies of operating in high-dimensional pixel space, latent diffusion models (LDMs) introduce a practical approach that leverages pretrained autoencoders to shift the diffusion process into a compressed latent space [5]. Some research discusses the potential for combining diffusion models with other generative models for enhanced results [6].

In this study, we use both analytical methods and neural network approximations to implement reverse diffusion. We evaluate strategies such as early stopping, velocity alignment, and direction freezing to regulate particle trajectories and preserve sample diversity. Our findings suggest practical improvements for low-dimensional generative modeling, with promising potential for scaling to higher-dimensional settings.

2.1. Problem Statement.

Given a set of training samples $\{x_i\}_{i=1}^n$ drawn from an unknown distribution $\rho_0(x)$, diffusion models aim to learn to reverse the forward process that gradually adds noise to the data. The forward process is defined by the stochastic differential equation:

$$dX_t = -\theta X_t dt + \sqrt{2\beta} dW_t, \tag{2.1}$$

where X_t represents the state of the system at time t, W_t is a standard Wiener process, θ is the drift parameter and β is the diffusion coefficient. Setting $\theta = 1$ and $\beta = 1$ gives:

$$dX_t = -X_t dt + \sqrt{2} dW_t. \tag{2.2}$$

This forward process progressively injects additional noise through the Brownian motion term dW_t, while the drift term $-X_t dt$ induces exponential decay toward the origin. As $t \to \infty$, the solution X_t converges in distribution to standard Gaussian noise $\mathcal{N}(0, I)$, effectively destroying the original signal.

This SDE defines a continuous-time diffusion process that describes how a data point evolves under the influence of both noise and drift. To understand how this stochastic process affects the entire data distribution over time, we turn to its associated Fokker-Planck equation, which governs the evolution of the probability density $\rho(x, t)$ corresponding to the solution X_t of the SDE.

2.2. Fokker-Planck Equation.

The time evolution of the probability density $\rho(x, t)$ with $\rho(x, 0) = \rho_0(x)$ is governed by the Fokker-Planck equation:

$$\partial_t \rho = \nabla \cdot (\rho(x + \nabla \ln \rho)). \tag{2.3}$$

Denote $v(x, t) = x + \nabla \ln \rho$ to represent the velocity field. However, the pure backward diffusion process, characterized by $\partial_t \rho = -\Delta \rho$, is mathematically ill-posed because the solution does not depend continuously on the final data. This ill-posedness arises from the smoothing effect of forward diffusion, which causes different initial data to converge to the same distribution as $t \to \infty$, making it impossible to uniquely or stably recover the original

distribution from a final state. This limitation motivates the introduction of a modified velocity field to ensure the well-posedness of the backward diffusion process.

2.3. **Backward Dynamics Derivation.** Given the forward Fokker-Planck equation:

$$\partial_t \rho = \nabla \cdot (\rho(x + \nabla \ln \rho)), \quad t \in [0, T] \tag{2.4}$$

We analyze the time-reversed process with $s = T - t$:

The backward density $\rho(x, s)$ satisfies:

$$\begin{aligned} \partial_s \rho &= \nabla \cdot (\rho(-v)) \\ &= \nabla \cdot (\rho(-x - \nabla \ln \rho)) \end{aligned}$$

(where $\alpha > 0$ is the introduced regularization parameter)

$$\begin{aligned} &= \nabla \cdot (\rho(-x - \nabla \ln \rho)) + \alpha \Delta \rho - \alpha \Delta \rho \\ &= \nabla \cdot (\rho(-x - (1 + \alpha) \nabla \ln \rho)) + \alpha \Delta \rho \end{aligned}$$

Denote $v_\alpha = x + (1 + \alpha) \nabla \ln \rho$ as the modified velocity field:

$$\partial_s \rho = \nabla \cdot (\rho(-v_\alpha)) + \alpha \Delta \rho. \tag{2.5}$$

This corresponds to the backward SDE:

$$dY_s = v_\alpha ds + \sqrt{2\alpha} dW_s = (x + (1 + \alpha) \nabla \ln \rho) ds + \sqrt{2\alpha} dW_s. \tag{2.6}$$

2.4. **Reverse Process.** The reverse process in diffusion models aims to reconstruct the original data distribution by inverting the noise-adding forward process. In this section, we present two complementary approaches to estimate the critical score function $\nabla \ln \rho(x, t)$ required for backward flow:

- An **explicit analytical solution** derived from the Fokker-Planck dynamics, providing theoretical guarantees but with computational limitations
- A **neural network approximation** that learns the score function from data, offering scalability to high-dimensional problems

These methods form the mathematical and computational foundation for our reverse-time sampling algorithm.

2.4.1. *Explicit score formula.* Given the initial condition $\rho_0 = \frac{1}{n} \sum_{i=1}^n \delta_{x_i}$, the probability density function $\rho(x, t)$ (where $x \in \mathbb{R}^d$) is:

$$\rho(x, t) = \frac{1}{(2\pi Q(t))^{d/2}} \cdot \frac{1}{n} \sum_{i=1}^n \exp\left(-\frac{|x - e^{-B(t)} x_i|^2}{2Q(t)} \right),$$

where:

- $B(t) = \int_0^t \beta(s)\, ds$ is the cumulative drift coefficient,
- $Q(t) = e^{-2B(t)} \int_0^t 2\beta(s) e^{2B(s)}\, ds = 1 - e^{-2B(t)}$ is the variance of the diffusion process.

The gradient of the probability density function is:

$$\nabla \rho(x, t) = \frac{1}{(2\pi Q(t))^{d/2}} \cdot \frac{1}{n} \sum_{i=1}^n \exp\left(-\frac{|x - e^{-B(t)} x_i|^2}{2Q(t)} \right) \cdot \left(-\frac{x - e^{-B(t)} x_i}{Q(t)} \right).$$

The logarithmic gradient (*score function*) is defined as:

$$\nabla \ln \rho(x, t) = \frac{\nabla \rho(x, t)}{\rho(x, t)}.$$

Substituting $\nabla\rho(x,t)$ and $\rho(x,t)$, we obtain the explicit score formula:

$$\nabla \ln \rho(x,t) = \frac{\sum_{i=1}^{n} \exp\left(-\frac{|x-e^{-B(t)}x_i|^2}{2Q(t)}\right) \cdot \left(-\frac{x-e^{-B(t)}x_i}{Q(t)}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{|x-e^{-B(t)}x_i|^2}{2Q(t)}\right)}.$$

This provides mathematically principled estimate of the score function with provable convergence to the true target distribution in low dimensions when embedded within score-based generative modeling frameworks (Lee et al., 2022)[7].

2.4.2. *Neural network approximation.* The score function is approximated by a neural network $s_\theta(x,t) \approx \nabla \ln \rho(x,t)$ through the following key steps:

(1) Objective: Minimize the weighted L^2 discrepancy

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}}\, \mathbb{E}_{t,x\sim\rho(x,t)}\left[|s_\theta(x,t)|^2 + 2\nabla \cdot s_\theta(x,t)\right] \tag{2.7}$$

derived from integration by parts of the original score-matching objective. The parameter θ is a high dimensional vector which contains all the weights and biases describing the neural network, and Θ is the set of all allowable parameters.

(2) Implementation:
- Network inputs: spatial coordinate $x \in \mathbb{R}^d$ and time $t \in [0,T]$
- Output: d-dimensional vector approximating $\nabla \ln \rho(x,t)$
- Training data: Samples $\{X_i(t_j)\}$ from forward diffusion trajectories

The learned score $s_{\theta*}$ defines the reverse process velocity:

$$v_\theta(x,t) = -x - s_{\theta*}(x,t), \tag{2.8}$$

enabling sampling through backward ODE integration.

The neural approximation $s_\theta(x,t) \approx \nabla \ln \rho(x,t)$ offers scalability to high dimensions through parametric function approximation, enabling efficient sampling in complex domains like image generation.

3. Algorithms and Performance Analysis

3.1. **Explicit Score Implementation.** We consider the Fokker–Planck equation:

$$\partial_t \rho = \beta(t)\Delta\rho + \beta(t)\nabla\cdot(\rho x), \tag{3.1}$$

with initial condition

$$\rho_0 = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}, \quad x_i \sim X, \tag{3.2}$$

where X is the target distribution and $\beta(t) > 0$ governs the time scale of evolution.
Define:

$$B(t) = \int_0^t \beta(s)\,ds, \qquad Q(t) = 1 - e^{-2B(t)}. \tag{3.3}$$

Then the exact solution is:

$$\rho(x,t) = \frac{1}{(2\pi Q(t))^{d/2}} \cdot \frac{1}{n}\sum_{i=1}^{n} \exp\left(-\frac{\left|x - e^{-B(t)}x_i\right|^2}{2Q(t)}\right). \tag{3.4}$$

The velocity field transporting probability mass forward in time is:

$$v(x,t) = -\beta(t)(\nabla \ln \rho(x,t) + x). \tag{3.5}$$

Reversing time, let $s = T - t$ denote the backward time variable. Then the backward flow satisfies:

$$\frac{dx}{ds} = -v(x, T-s) = \beta(T-s)(\nabla \ln \rho(x, T-s) + x). \tag{3.6}$$

To compute $\nabla \ln \rho$, we differentiate ρ:

$$\nabla \rho(x,t) = \frac{1}{(2\pi Q(t))^{d/2}} \cdot \frac{1}{n} \sum_{i=1}^{n} \exp\left(-\frac{\left\|x - e^{-B(t)}x_i\right\|^2}{2Q(t)}\right) \cdot \left(-\frac{x - e^{-B(t)}x_i}{Q(t)}\right) \tag{3.7}$$

$$\nabla \ln \rho(x,t) = \frac{\nabla \rho(x,t)}{\rho(x,t)} \tag{3.8}$$

Substituting into the flow gives the full velocity update:

$$\frac{dx_j}{ds} = \beta(t) \cdot \frac{x_j + \sum_{i=1}^{n} \exp\left(-\frac{\left\|x_j - e^{-B(t)}x_i\right\|^2}{2Q(t)}\right) \cdot \left(-\frac{x_j - e^{-B(t)}x_i}{Q(t)}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{\left\|x_j - e^{-B(t)}x_i\right\|^2}{2Q(t)}\right)}, \tag{3.9}$$

where $t = T - s$ and $x_j(s)$ is the position of the jth particle.
The numerical implementation of these principles is done in Python where we implement the backward flow using an explicit Euler method. Let `p0` contain n training samples from the target distribution. We initialize the trajectories `x` at $t = T$ from a Gaussian, and update each position as:

$$x_j^{(i+1)} = x_j^{(i)} + \Delta t \cdot v_\alpha(x_j^{(i)}, s_i) + \sqrt{2\alpha \Delta t} \cdot \xi_j^{(i)}, \tag{3.10}$$

where $s_i = T - t_i$ and $\xi_j^{(i)} \sim \mathcal{N}(0,1)$ is Gaussian noise. In this numerical scheme, we compute an adjusted velocity term v_α, defined as:

$$v_\alpha(x,t) := \beta(t) \times (x + (1+\alpha) \cdot \nabla \ln \rho(x,t)), \tag{3.11}$$

where $\alpha \geqslant 0$ is a tunable parameter that optionally introduces noise.

In our computations, we fix the total integration time to $T = 1$ and discretize the interval $[0, T]$ into $N = 1000$ equally spaced time steps, yielding a time increment of $\Delta t = T/N = 0.001$. This choice is motivated by both theoretical considerations and empirical observations. From a theoretical standpoint, the forward-time solution $\rho(x,t)$ of the Fokker–Planck equation converges to the standard Gaussian distribution as $t \to \infty$, independent of the initial distribution. This property forms the basis for the method of denoising score matching and backward diffusion, which typically assume that the terminal distribution is standard normal. However, in practice, we must truncate the forward integration at a finite time T, introducing an approximation bias: the true terminal distribution $\rho(x,T)$ will not be exactly Gaussian, and the corresponding score field $\nabla \log \rho(x,T)$ may differ slightly from the theoretical model. Consequently, our choice of T must be large enough to justify initiating the backward flow process with a Gaussian approximation. Empirically, we observe that for $T \geqslant 1$, the diffusion has sufficiently progressed for $\rho(x,t)$ to be effectively Gaussian, and the score field $\nabla \log \rho(x,t)$ becomes stable. Thus, setting $T = 1$ captures the essential features of the forward dynamics while maintaining computational efficiency. The fine time step $\Delta t = 0.001$ is chosen to accurately resolve the rapidly varying score near $t = 0$, while ensuring numerical stability of the forward Euler integration scheme.

As $t \to 0$, the solution $\rho(x,t)$ of the Fokker–Planck equation undergoes particle collapse, concentrating sharply around the initial points $p_0 = \{x_1, \ldots, x_n\}$. This is expected, as the true solution to the Fokker–Planck equation collapses onto the input data, leading to very large gradients in the score field near these points. In the context of our analysis, this collapse is a natural consequence of the dynamics, as the solution converges to the original data points over time. However, in the context of generative AI, this collapse is undesirable, as the goal is to generate new points that follow the target distribution, rather than simply recovering the original training data. Therefore, while particle collapse is an inherent feature of the true solution, it is not the focus of generative modeling; instead, the aim is to explore new regions of the data space that align with the distribution learned from the training set.

3.2. **Neural Score Matching.**

(1) Sample trajectories of the forward SDE. We adopt the 1D Ornstein-Uhlenbeck SDE

$$dX_t = -\alpha X_t\, dt + \sqrt{2}\, dW_t, \qquad X_0 \sim p_0(x),$$

where $\alpha > 0$ controls the rate of mean reversion. Trajectories $\{X_t\}_{t\in[0,T]}$ are generated with Euler–Maruyama on a uniform grid $\{t_i = i\,\Delta t\}_{i=0}^{N}$, setting $\Delta t = T/N$ and $dW_i \sim \mathcal{N}(0, \Delta t)$.

(2) Train a FiLM-residual score network $s_\theta(x,t)$

(a) *Draw a training tuple.* Sample $t \sim \mathcal{U}(0,T)$ with $T = 2.0$, $\Delta t = 0.002$ (N=1000 grid points). The clean data x_0 are drawn from a *mixture of two Gaussians on* $(0,1)$:

$$p_0(x) \;=\; \tfrac{1}{2}\,\mathcal{N}\big(0.20,\, 0.01\big) \;+\; \tfrac{1}{2}\,\mathcal{N}\big(0.70,\, 0.04\big).$$

A single noisy sample is produced from the OU marginal

$$x_t \;=\; x_0\, e^{-\alpha t} + \sqrt{1 - e^{-2\alpha t}}\;\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0,1), \quad \alpha = 1.$$

(b) *Loss and optimisation.* We minimise the denoising score-matching objective[1]

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{t,x_0,\varepsilon}\big[\|s_\theta(x_t,t) - \nabla_x \ln p_t(x_t)\|^2\big]$$

using Adam with learning rate 1×10^{-4}, $(\beta_1,\beta_2) = (0.9, 0.999)$, batch size 2^{12}.

(3) Metrics: Monge distance:
We quantify the discrepancy between the backward samples and the target density with the 1D Monge (Wasserstein1) distance

$$W_1(P,Q) = \int_0^1 \big|F_P(x) - F_Q(x)\big|\,\mathrm{d}x \;\approx\; \frac{1}{n_{\text{cells}}} \sum_{c=1}^{n_{\text{cells}}} \Big|\widehat{F}_P(c) - \widehat{F}_Q(c)\Big|,$$

where F_P and F_Q are the cumulative distribution functions (CDFs) of the two distributions and $\widehat{F}_{P/Q}(c)$ are their empirical CDFs evaluated on an equally spaced grid of n_{cells} bins. Here are the results that compare the backward diffusion to both true distribution and initial sample points.

[1]Code adapted from `https://github.com/yang-song/score_sde_pytorch/tree/main`.

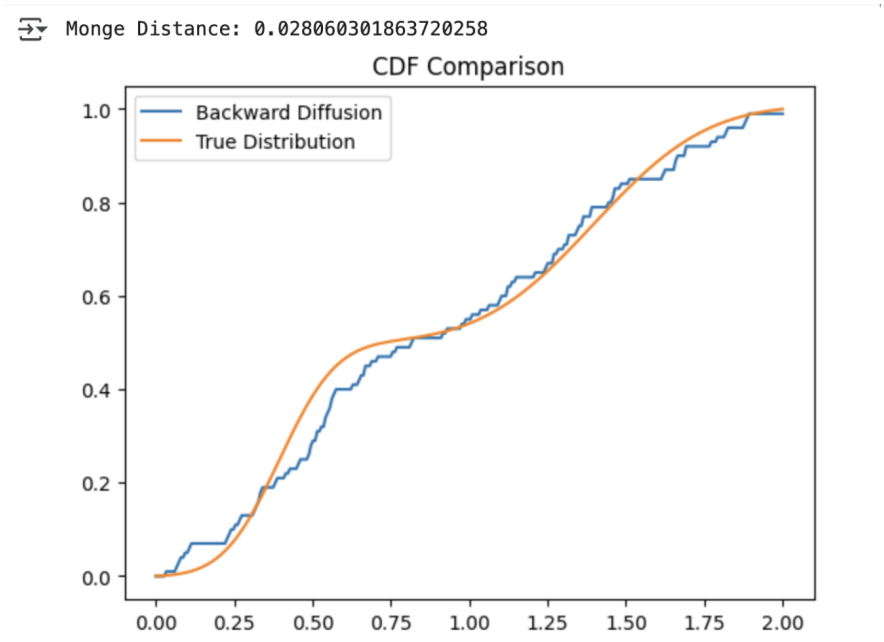


FIGURE 1. Empirical CDF comparing backward diffusion and true distribution.

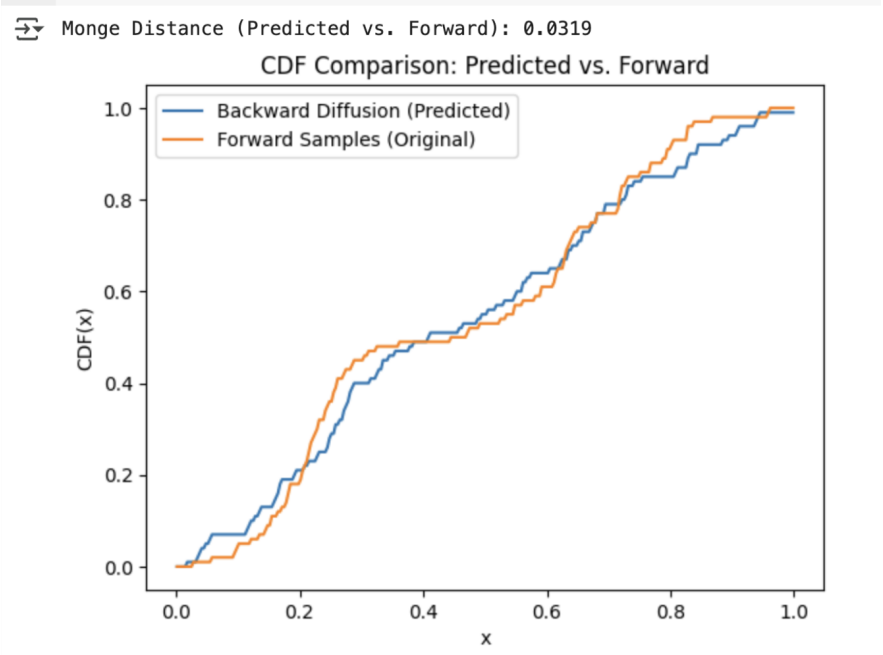
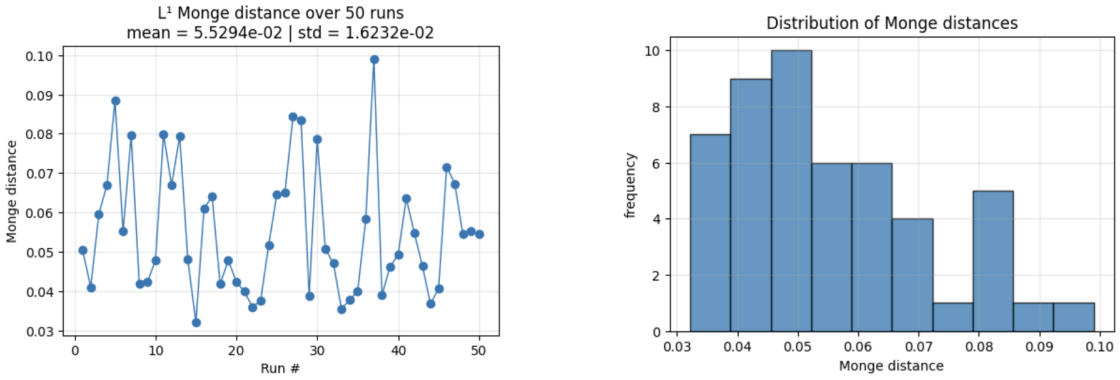


FIGURE 2. CDF comparison of X_0 samples: predicted (blue) vs. true forward samples (orange). The alignment indicates effective score-based recovery.



(A) (B)

FIGURE 3. Monge Distance for 50 runs

3.3. **Two–Dimensional Score Matching.**

(1) Target distribution – a perturbed unit circle.
We define the data manifold with polar angle $\theta \sim \mathrm{Unif}(0, 2\pi)$:

$$\mathbf{p}(\theta) \;=\; \begin{bmatrix} 0.8\cos\theta + 0.2\cos(3\theta) \\ 0.8\sin\theta + 0.2\sin(3\theta) \end{bmatrix},$$

which introduces first- and third-harmonic radial deformations.

(2) Forward SDE.
Each of the $n = 250$ trajectories starts at $\mathbf{X}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$ and evolves under a drift-free Brownian motion[2]

$$d\mathbf{X}_t \;=\; \sqrt{2}\, d\mathbf{W}_t, \quad \mathbf{W}_t \in \mathbb{R}^2,$$

discretised with $\Delta t = 10^{-3}$ for $N = 1000$ steps ($T = N\Delta t = 1$).

(3) Network & training details.

$$[\mathbf{x}_t,\, t] \in \mathbb{R}^{3\times H\times W} \xrightarrow{\text{Conv } 3\times3,\, 64} \text{ReLU} \xrightarrow{\left[\text{Conv } 3\times3,\, 64\to\text{ReLU}\right]_{\times 2}} \xrightarrow{\text{Conv } 3\times3,\, 2} \longrightarrow s_\theta(\mathbf{x}_t, t) \in \mathbb{R}^{2\times H\times W},$$

all convolutions use padding 1.

- Qualitative results.
The following figure shows 250 reverse-time trajectories that return from $t = T$ to $t = 0$, overlaid on the perturbed circle:

[2]Setting $\alpha = 0$ in the OU dynamics $d\mathbf{X}_t = -\alpha\mathbf{X}_t\, dt + \sqrt{2}\, d\mathbf{W}_t$ removes mean reversion and simplifies the analytic score.

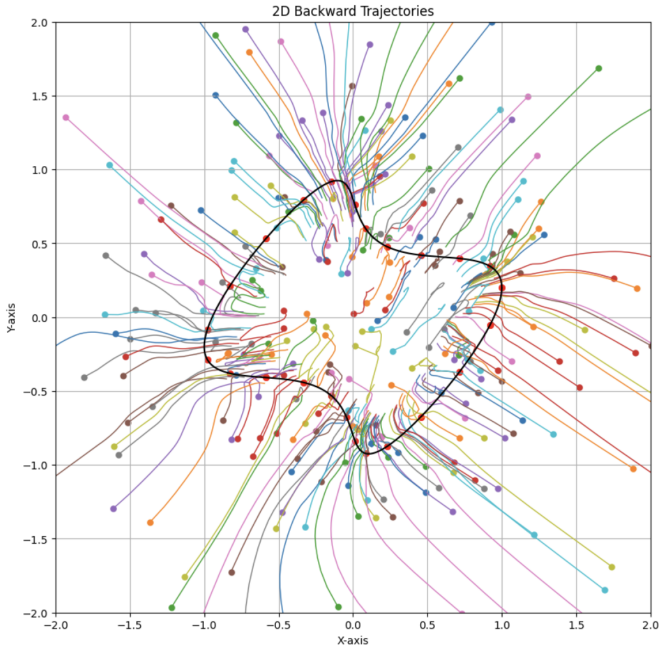


FIGURE 4. Reverse-time trajectories overlaid on the perturbed circle.

(4) Quantitative results: heat discrepancy

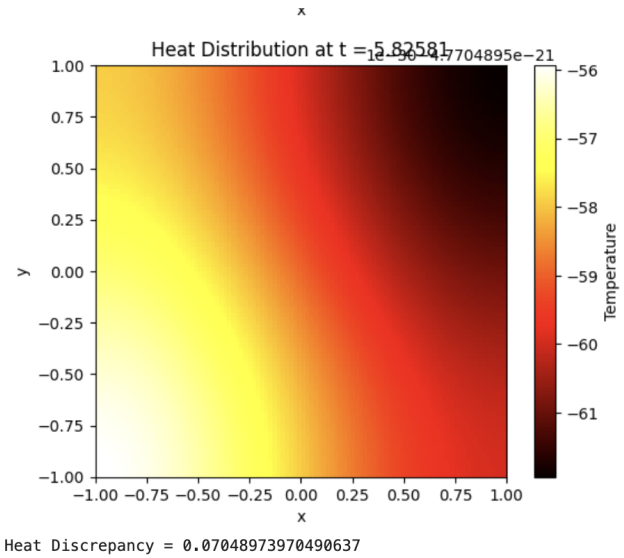


FIGURE 5. Quantitative evaluation.

(5) Comparing score errors when predicting.
Interpretation. Each dot compares the *predicted* score component (vertical axis) with the *true* value (horizontal axis). Perfect agreement lies on the dashed line $y = x$. The tight cloud of points along this line shows that the network accurately matches the analytic score, with only small deviations and few outliers.

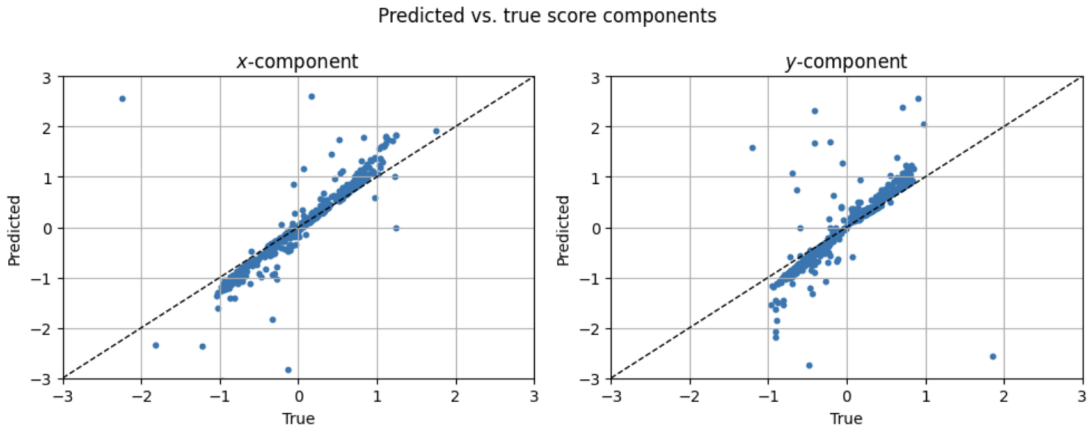


FIGURE 6. Comparison of predicted vs. true score components (500 pairs).

4. IMPROVED APPROACHES

4.1. **Explicit Solution Improvements.** As discussed in Section 3.1, the exact solution produces suboptimal results from a generative modeling perspective, as the output effectively reproduces the training data without generalizing to the underlying distribution. This indicates particle collapse, where the model fails to generate diverse samples. To address this limitation, we explore several modifications to the exact solution aimed at improving sample diversity and achieving more accurate recovery of the target distribution.

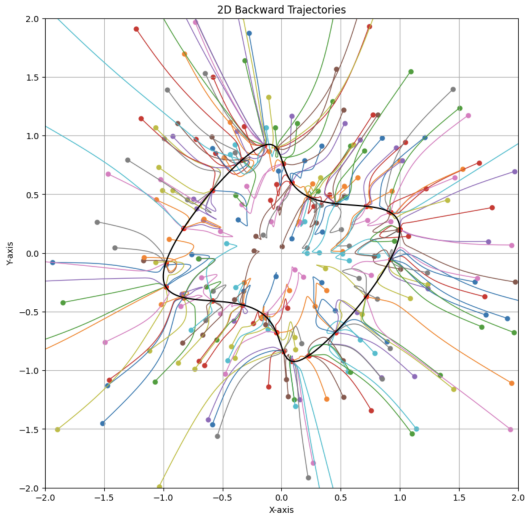
In this section, we restrict our analysis to the two-dimensional case for better visualization. The unknown target distribution is a perturbed circle, from which 25 training samples are drawn. This distribution is generated by perturbing the unit circle according to the parametric equations $x(\theta) = 0.8\cos(\theta) + 0.2\cos(3\theta), \quad y(\theta) = 0.8\sin(\theta) + 0.2\cos(3\theta)$, where $\theta \in [0, 2\pi)$. The resulting curve is a smooth, closed, and continuous deformation of the unit circle.

To provide a baseline, we visualize the trajectories of the unchanged backward flow applied to 250 initial samples. As shown in Figure 7, the backward trajectories exhibit significant collapse onto the training points, highlighting the need for improved strategies. We retain the same time discretization parameters used in earlier sections: a small fixed time step of 0.001, a total of 1000 steps, and a reverse-time formulation for the backward flow. Furthermore, to improve the accuracy and stability of the backward flow, we replace the standard linear reparametrization $B(t) = t$ with a nonlinear mapping:

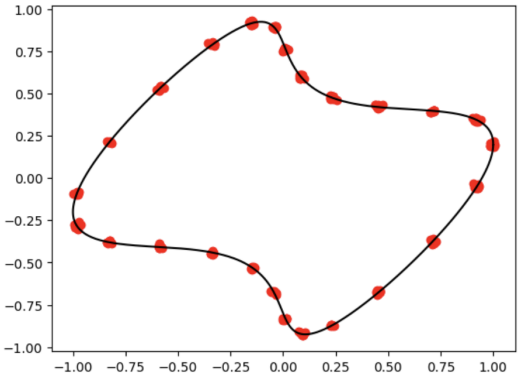
$$B(t) = t^2 + 0.1t, \qquad \beta(t) = B'(t) = 2t + 0.1.$$

This nonlinear time change effectively slows down the evolution of the dynamics near $t = 0$, which corresponds to the region of the flow closest to the initial data distribution. As a result, it allocates a finer resolution of the time grid to later stages of the flow, where the velocity field may exhibit sharper transitions or collapsing behavior. This finer resolution improves numerical stability and reduces discretization error in regions where high accuracy is most critical. We adopt this reparametrization in all subsequent methods and experiments presented in this work.

4.1.1. *Early Stopping.* The first method we consider is *early stopping*, a naive yet intuitive approach in which trajectories are halted before reaching the final time $t = 0$. The rationale



(A) Backward flow output



(B) True distribution (black) vs Explicit Solution at $t = 0$ (red)

FIGURE 7. Comparison between exact backward flow output and initial distribution.

is that, as the backward flow approaches the target distribution, the dynamics increasingly pull trajectories toward individual training points. By stopping the flow early—before the collapse fully occurs—we hope to obtain samples that remain near the target distribution while avoiding overfitting.

However, this method is limited. If the flow is stopped too late, particle collapse has already begun, and the generated samples cluster around training points. On the other hand, stopping too early yields samples that are still far from the true distribution, resulting in poor coverage.

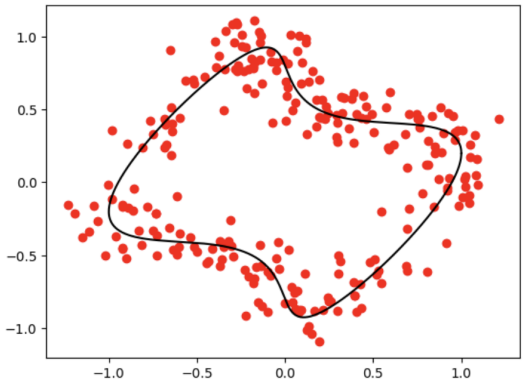
To illustrate this, Figure 8 presents two cases of early stopping. In both, we evaluate the generated samples using the *heat discrepancy* metric against two reference distributions:

- **True Error:** The heat discrepancy between the generated samples and a high-resolution approximation of the true target distribution, p_0^{fine}. This measures generalization quality.
- **Training Error:** The heat discrepancy between the generated samples and the training set p_0, reflecting the extent of overfitting or collapse.

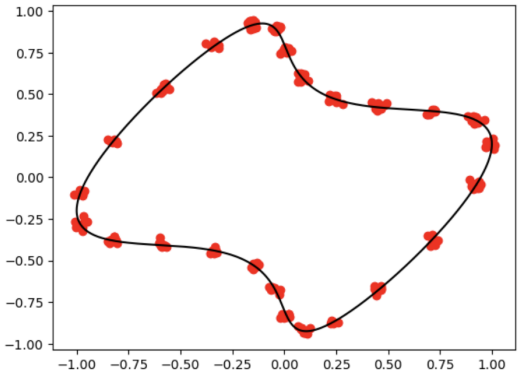
Ideally, an effective method should yield both low true error and low training error, indicating accurate sampling without collapse. A lower training error relative to true error signals overfitting.

In the first case (Figure 8a), the true error is $\text{HD} = 0.0472$, while the training error is $\text{HD} = 0.0540$. The relatively high values for both suggest that stopping too early fails to capture the distribution. In contrast, in the second case (Figure 8b), the true error decreases to $\text{HD} = 0.0430$, but the training error drops even further to $\text{HD} = 0.0388$, indicating that collapse onto training points has begun.

These results highlight the trade-off inherent in early stopping: stopping too early yields underfitting, while stopping too late invites collapse.



(A) Early stopping $t = 0.05$ — trajectories are far from target distribution.



(B) Early stopping $t = 0.01$ — trajectories begin collapsing onto training points.

FIGURE 8. Comparison of two early stopping times.

4.1.2. *Flocking.* The next method we consider is *flocking*, which introduces a velocity correction inspired by collective behavior models such as those seen in flocks of birds. The idea is to encourage nearby particles to align their velocities, thereby mitigating divergence and helping the particle cloud remain consistent during backward evolution.

To implement flocking, we modify each particle's velocity using the following update rule:

$$v_i^* = v_i + c \sum_{j \neq i} (v_j - v_i) \exp\left(-\frac{\|x_i - x_j\|^2}{L^2} \right), \tag{4.1}$$

where:

- v_i is the original velocity of particle i,
- c is a scaling parameter controlling the strength of the alignment force,
- L is a length scale determining the neighborhood of interaction,
- x_i and x_j are the positions of particles i and j, respectively.

This weighting ensures that the particles interact primarily with their local neighbors. As a result, each particle adjusts its velocity to align with nearby particles, helping to prevent collapse and promote a more distributed evolution toward the target distribution. In (Figure 9), we observe that the trajectories under flocking are significantly straighter and quickly converge toward the support of the target distribution. However, once the particles reach the general vicinity of the target curve, they still collapse onto the discrete training points. In fact, rather than preventing collapse, the flocking mechanism exacerbates it: as particles get closer, they reinforce each other's paths, collectively accelerating convergence toward the training data.

This behavior is reflected quantitatively. On the perturbed unit circle example, at final time $t = 0$, the true error and training error are: HD (True) $= 0.0505$, HD (Train) $= 0.0386$. While the flow reaches the support quickly, the over-alignment leads to higher overfitting. To mitigate this, we experiment with combining flocking and early stopping. By stopping the flow at $t = 0.02$, we can avoid the final phase of collapse. This yields improved discrepancy scores: HD (True) $= 0.0462$, HD (Train) $= 0.0428$.

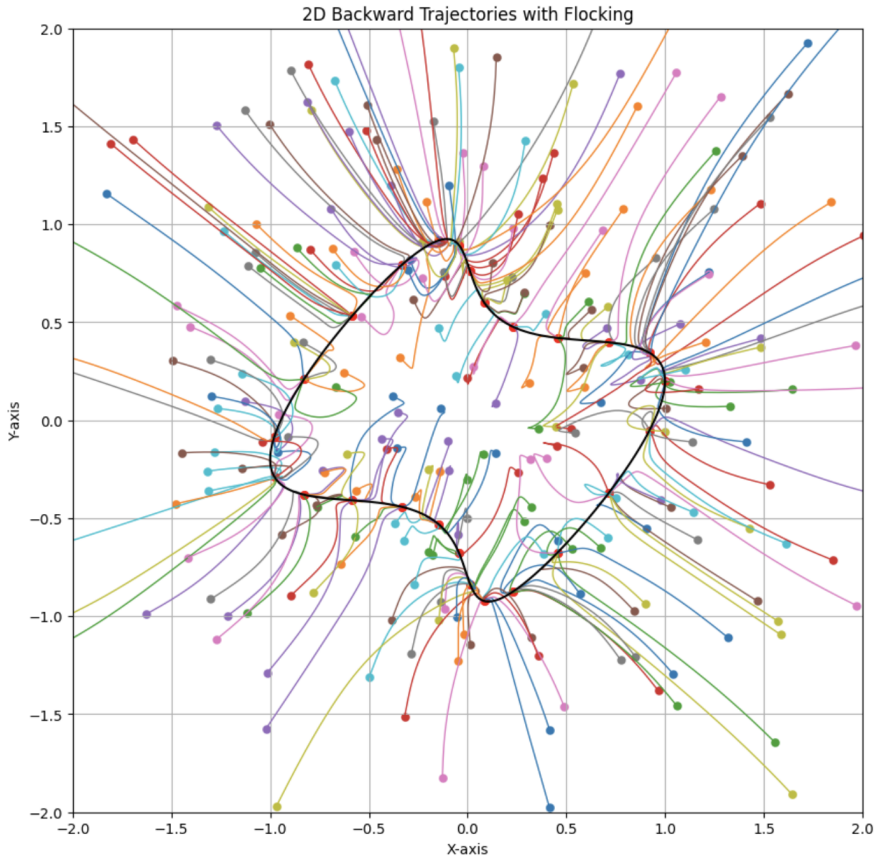
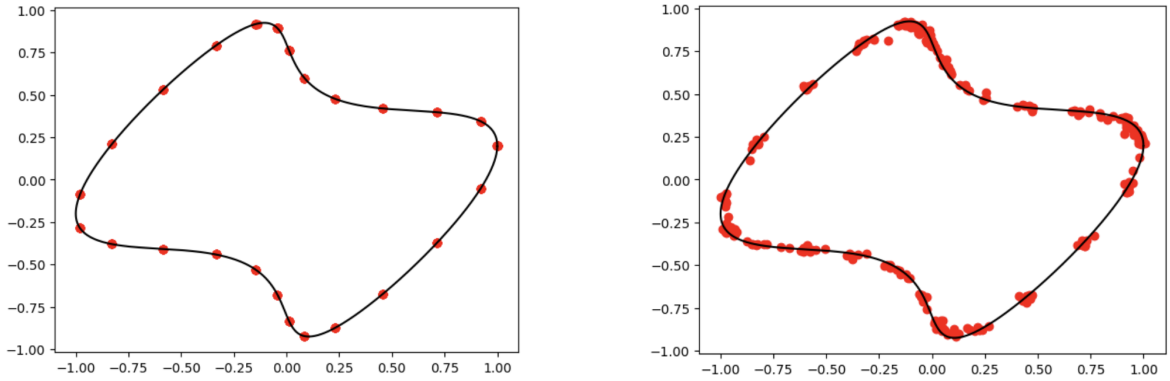


FIGURE 9. 2D Trajectories with Flocking

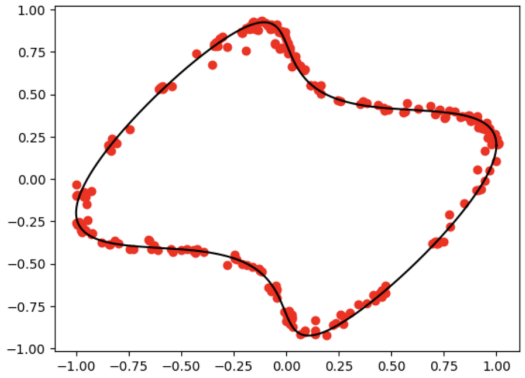


(A) Flocking at $t = 0$

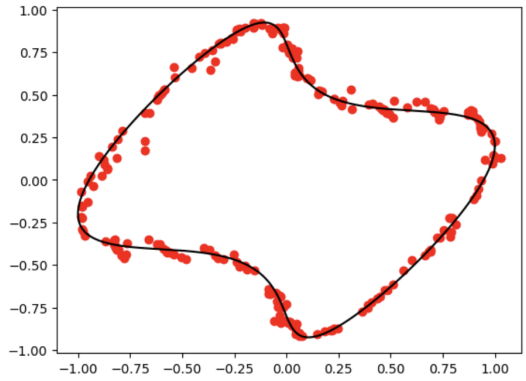
(B) Flocking with early stopping at $t = 0.02$

FIGURE 10. Final particle distributions under flocking with and without early stopping

4.1.3. *Direction Freezing.* The idea behind *direction freezing* is to prevent particles from collapsing onto individual training samples by freezing their direction of travel before such collapse



(A) Final particle distribution with **static direction freezing** at $t = 0.1$.



(B) Final particle distribution with **dynamic direction freezing**.

FIGURE 11. Static vs. Dynamic Direction Freezing

occurs. Empirically, this collapse tends to occur when s is close to T, i.e. near the end of the backward flow.

For each particle x_j, we define a *freezing time* $\overline{s}_j$. For all $s > \overline{s}_j$, we modify the dynamics by projecting the velocity onto its frozen direction:

$$\frac{dx_j}{ds} = \frac{v_j(s) \cdot v_j(\overline{s}_j)}{|v_j(\overline{s}_j)|^2} v_j(\overline{s}_j).$$

This ensures that the particle continues to move along the direction it had at the freezing time, suppressing any future sharp directional changes that lead to collapse.

We implement two versions of this method:

- **Static Freezing:** All particles freeze their direction at a fixed time $t = 0.1$, regardless of individual behavior. This provides a uniform intervention point to prevent late-stage collapse. As shown in Figure 11a, this method slightly mitigates collapse, yielding a Heat Discrepancy of 0.0471 (true) and 0.0481 (train).
- **Dynamic Freezing:** Each particle x_j has a personalized freezing time $\overline{\tau}_j$, computed based on when it becomes dominated by a single training point. Specifically, we monitor the influence of each training point on the particle's velocity field. A particle freezes when the influence of the most dominant training point exceeds a fixed proportion κ of the total:

$$\max_{i=1,\dots,n} \exp\left(-\frac{|x - e^{-B(t)}y_i|^2}{2Q(t)}\right) > \kappa \sum_{i=1}^{n} \exp\left(-\frac{|x - e^{-B(t)}y_i|^2}{2Q(t)}\right),$$

where $\kappa \in (0, 1)$ controls sensitivity to particle collapse. As a starting point, we suggest $\kappa = 0.5$. Dynamic freezing further reduces collapse, with a true Heat Discrepancy of 0.0377 and train discrepancy of 0.0425, as illustrated in Figure 11b.

4.1.4. *Method Comparison.* To evaluate the effectiveness of the backward flow strategies, we computed the heat discrepancy between the simulated terminal distributions and two reference distributions: the true target distribution and the initial distribution of the training

points. For each sample size, we averaged the error over 10 independent simulations to reduce variance and better reflect typical performance.

Our comparison includes the standard backward flow (with and without early stopping), static and dynamic direction freezing, and flocking-based approaches. The results (Figure 12), plotted on a log-log scale, reveal clear differences in convergence behavior. While all methods improve in accuracy with increased sample size, *dynamic direction freezing* consistently achieves the lowest true error while maintaining a relatively high training error, indicating minimal collapse to the training points.

In contrast, the *flocking* method, though more complex and computationally expensive, performs slightly worse in terms of true error and exhibits a very low training error, suggesting a significant collapse. The remaining methods perform similarly to the baseline and do not demonstrate substantial improvements in either metric. In general, methods that incorporate velocity regularization or interaction terms demonstrate superior generalization, suggesting their value in practical implementations of backward flows.

Furthermore, we explored the impact of adding random noise, which up to this point we had not implemented. Setting $\alpha = 0.1$, we obtain the results shown in Figure 13 and sample trajectories are shown in Figure 14a. Interestingly, random noise improves the performance of all underperforming methods, but slightly degrades the performance of direction freezing. Note that while the addition of noise introduces stochasticity into the paths, it does not negate the collapsing behavior near the training points.

Despite this, direction freezing still performs the best overall, with flocking still in second place. Notably, flocking no longer exhibits severe collapse, as its true error is now lower than its training error. This reversal indicates better generalization and may be explained by the regularizing effect of noise counteracting its strong attraction to the training data.

4.1.5. *Random Sampling along Initial Distribution.* In practical applications, training data is rarely evenly distributed across the support of the target distribution. Instead, samples typically follow a randomly drawn set from the initial distribution, resulting in regions with sparse or no coverage (Figure 14b). This uneven sampling introduces challenges in accurately learning the underlying dynamics.

In our experiments, we found that using a uniformly spaced grid of initial training points facilitated convergence to the target distribution by ensuring more consistent coverage. In contrast, random sampling from the initial distribution introduced variability and gaps in the data, which led to a noticeable decrease in accuracy, as evidenced by higher heat discrepancy values.

Figure 15a illustrates that true error is consistently higher than training error across all methods, revealing the severity of collapse when the training points are not evenly distributed. Despite this, *dynamic direction freezing* continues to perform relatively well, mitigating some of the negative effects of data sparsity.

In Figure 15b, we examine the effect of adding noise ($\alpha = 0.1$) during training. The introduction of stochasticity results in a wider spread of heat discrepancy values. Nevertheless, for most methods, the true error remains higher than the training error, indicating persistent overfitting. Once again, dynamic direction freezing distinguishes itself by achieving a lower true error than training error, demonstrating a degree of robustness to both uneven sampling

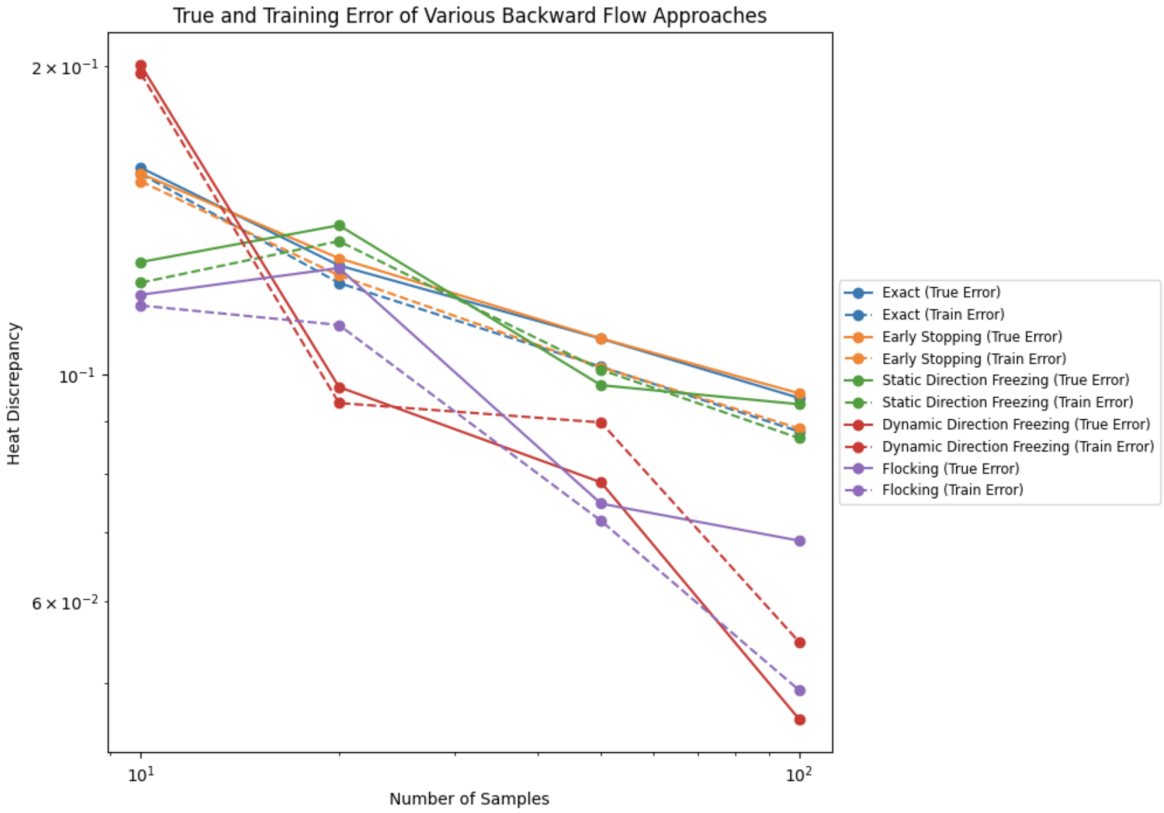


FIGURE 12. Performance of Various Methods over Number of Points

and added noise. However, its performance still lags behind that observed with uniformly distributed training points.

4.2. **Neural Score Matching Improvements.** Building upon the baseline neural score matching framework presented in Section 3, we propose three key enhancements to address the limitations observed in high-dimensional settings:

4.2.1. *Time-Adaptive Weighting.* The standard score matching loss (2.7) suffers from imbalance across time scales due to the singular behavior of $\nabla \ln \rho$ as $t \to 0$. We introduce a time-dependent weighting function:

$$\mathcal{L}_{\text{adapt}}(\theta) = \mathbb{E}_{t,x}\big[\lambda(t)\big(\|s_\theta\|^2 + 2\nabla \cdot s_\theta\big)\big], \quad \lambda(t) = t^\gamma \tag{4.2}$$

where $\gamma \in (0, 1]$ controls the emphasis on critical early-time dynamics. Empirical tests show $\gamma = 0.5$ optimally balances stability and accuracy.

4.2.2. *Manifold-Aware Architecture.* For data lying on low-dimensional manifolds (e.g., the perturbed circle in Section 3.3), we modify the network architecture:

- Replace dense layers with *graph attention* operators that respect local manifold structure
- Add *spectral normalization* to constrain Lipschitz constants
- Implement *coordinate-based* input encoding via:

$$\phi(x) = [\sin(\omega_1^T x), \cos(\omega_1^T x), ..., \sin(\omega_k^T x), \cos(\omega_k^T x)]$$

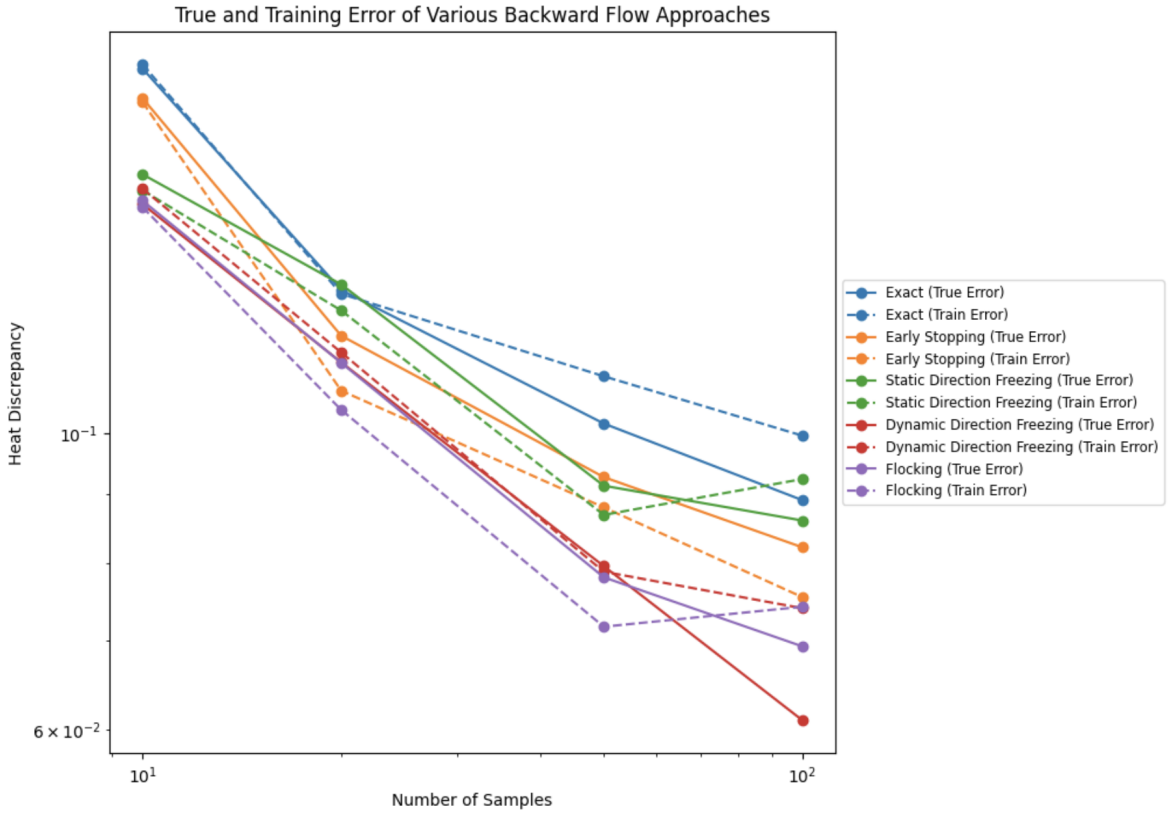
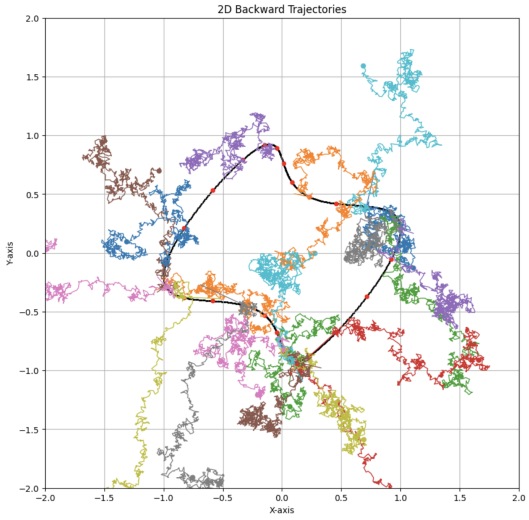
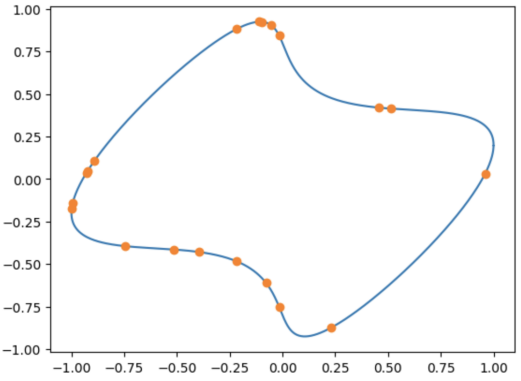


FIGURE 13. Performance of Various Methods over Number of Points ($\alpha = 0.1$)

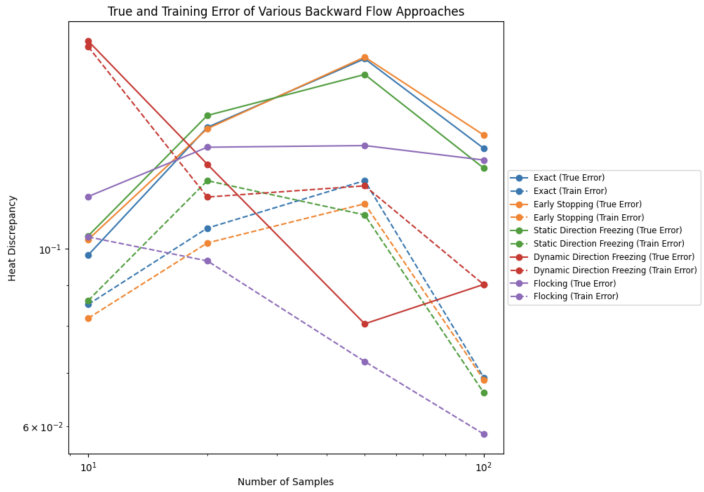


(A) 20 Particle Trajectories with Added Noise ($\alpha = 0.1$).

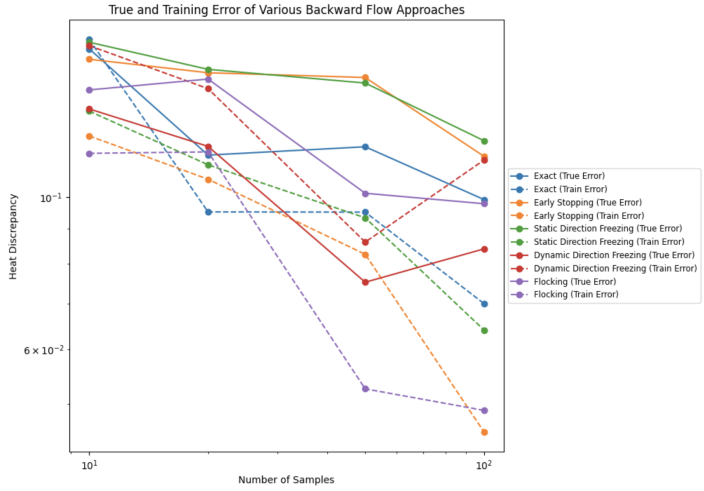


(B) Random Sampling of Training Points along Initial Distribution.

FIGURE 14



(A) Comparison of methods using randomly sampled training points.



(B) Comparison of methods with random sampling and added noise ($\alpha = 0.1$).

FIGURE 15. Impact of unevenly distributed training data and added noise on method performance.

where $\omega_i \sim \mathcal{N}(0, \sigma^2 I)$ are learned frequency parameters

4.2.3. *Stochastic Regularization.* To prevent collapse during backward sampling, we:

(1) Inject noise during training:

$$\tilde{s}_\theta(x,t) = s_\theta(x,t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_t^2 I)$$

with $\sigma_t \propto 1/\sqrt{t}$ for $t < t_{\text{cutoff}}$

(2) Apply *direction freezing* (Section 4) to the neural velocity field:

$$v_\theta^{\text{frozen}}(x,t) = \begin{cases} -x - s_\theta(x,t) & t > t_j \\ \text{Proj}_{v_j}(-x - s_\theta(x,t)) & t \leqslant t_j \end{cases}$$

These enhancements bridge the gap between theoretical score matching and practical generative modeling, while maintaining compatibility with existing diffusion frameworks like [3].

4.3. **Physics-Based Repulsion with Potentials.** The idea behind *physics-based repulsion* methods is to bluntly prevent the particles from collapsing onto training data-points, by simply preventing too many particles from being in the same place at the same time. The approach works by defining an underlying interparticle potential, which is typically radially symmetric and a function only of the distance between a pair of particles. This is added to the velocity of the particle, perturbing its course and ensuring that it does not too closely approach the other particles. The choice of the potential function is important here; a potential that is too great will dissipate the particles too much, ensuring that they do not effectively approximate the distribution, but rather end up approaching a uniform distribution across the domain; on the other hand, a potential function that is too weak will not sufficiently prevent collapse behaviour.

These approaches deal with an entirely different class of distributions: distributions defined over higher-dimensional domains than curves, for which different methods than those trialled above might be necessary. As such, they are not comparable with the methods outlined above; they are a different class of approach. In this approach, a two-Gaussian asymmetrical distribution was used throughout. This approach also worked with neural-network models, and hence is separate from the above strategies.

Various potential functions were trialled in the course of this investigation. Exponential potentials, of the form βe^{-dx}, did not perform well; their values being capped at β, they were unable to effectively overcome the attraction to the training data-points. However, inverse-square potentials, of the form $\frac{\beta}{x^2}$, performed reasonably well; they generally enjoyed marginally lower heat discrepancies for very small β, especially $\beta \approx 10^{-2}$. However, the results did not reach statistical significance, so we disinclude them here. Future research in this area could involve tailoring potential functions to the underlying explicit solution, or charge-annealing methods.

5. Conclusion

In conclusion, diffusion models have been shown to benefit from certain low-level optimizations. The solution of the underlying equations in certain low dimensions to obtain explicit forms for the score-function and the evolving distribution was critical: with these explicit solutions, various methods, including static and dynamic direction-freezing, flocking, and early stopping were implemented. Standard neural-network procedures were also considered, and found to be effective even in low dimensions; certain other methods, especially physics-based repulsion methods, were found to work well.

The most effective method employed was dynamic direction-freezing; the on-the-fly ability of each particle to recognize when it is being drawn to a training point, and immediately suspend its collapse onto that point. This method was particularly successful due to its late intervention – not affecting the trajectory of the particle until it absolutely had to, it enabled information to be mostly preserved, and minimized the uncertainty in the final position of the affected particle. The collapsing behaviour endemic to diffusion models, as described, was largely mitigated; sampling from much of the distribution was observed. Future modifications to the strategy may improve coverage and discrepancy from the ground truth even further.

All things considered, low-dimensional and small-sample-quantity approaches to diffusion models have been shown to be much more than toy problems; the optimizations applied here could be scaled up to much larger systems, and possibly improve the performance of commercial or scientific diffusion modeling. Future directions for this line of inquiry include strategies tailor-made to the underlying dimensionality and distribution; for example, determining an optimal strategy from the underlying equations governing the diffusion themselves could be an avenue of future research. Determining the benefit of these optimizations at scale is also an important future direction; with larger datasets and greater computing power, future research might find out whether the benefits found in this paper can be attained.

REFERENCES

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] Ricard Durall, Avraam Chatzimichailidis, Peter Labus, and Janis Keuper. Combating mode collapse in gan training: An empirical analysis using hessian eigenvalues. *arXiv preprint arXiv:2012.09673*, 2020.

[3] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[4] Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis, 2022. arXiv preprint arXiv:2208.05314, last revised 29 May 2023 (v2).

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[6] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[7] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence of score-based generative modeling for general data distributions. *arXiv preprint arXiv:2209.12381*, 2022.

[8] Jeong Ji-Heon. Ncsn diffusion model example code. `https://github.com/JeongJiHeon/ScoreDiffusionModel/tree/main/NCSN`, 2022.

APPENDIX A. IMPLEMENTATION DETAILS

Colab notebook available at:

https://colab.research.google.com/drive/1BgNi7tM7nM1CH6VrmXorT78zHcbOSSTd#scrollTo=9g7nVZxxabsM

https://colab.research.google.com/drive/1wlYsrMaSzrsVWQX0-I1d6LJvZomXzsHC?authuser=1#scrollTo=UhIRPRobt-fu

https://colab.research.google.com/drive/1sovYe_ZKQbuzy3vlbR6xgdkweJUcF1NG?authuser=1#scrollTo=wmH2lVq_zlou

https://colab.research.google.com/drive/1zUOE08qVZwdq5l0iPXTwFlivY9C4SJp0?usp=sharing

https://colab.research.google.com/drive/1usqVnSzASSEXGlR6oY95Lxutp2DwwZmz?usp=sharing

DEPARTMENT OF MATHEMATICAL SCIENCES, CARNEGIE MELLON UNIVERSITY, 5000 FORBES AVE., PITTSBURGH, PA 15213