The Score-Difference Flow for Implicit Generative Modeling

Anonymous authors
Paper under double-blind review

Abstract

Implicit generative modeling (IGM) aims to produce samples of synthetic data matching the characteristics of a target data distribution. Recent work (e.g. score-matching networks, diffusion models) has approached the IGM problem from the perspective of pushing synthetic source data toward the target distribution via dynamical perturbations or flows in the ambient space. We introduce the *score difference* (SD) between arbitrary target and source distributions as a flow that optimally reduces the Kullback-Leibler divergence between them. We apply the SD flow to convenient proxy distributions, which are aligned if and only if the original distributions are aligned. We demonstrate the formal equivalence of this formulation to denoising diffusion models under certain conditions. However, unlike diffusion models, SD flow places no restrictions on the prior distribution. We also show that the training of generative adversarial networks includes a hidden data-optimization sub-problem, which induces the SD flow under certain choices of loss function when the discriminator is optimal. As a result, the SD flow provides a theoretical link between model classes that collectively address all three challenges of the *generative modeling trilemma*: high sample quality, mode coverage, and fast sampling.

1 Introduction

The goal of implicit generative modeling (IGM) is to create synthetic data samples by pushing a base (or source) distribution q, representing the synthetic data, toward a target distribution p until the two distributions are indistinguishable. A variety of approaches exist in the literature that address this problem from the perspective of the *dynamics* imposed upon synthetic data during sampling or training. This perspective can be applied to the direct sampling of data—such as in Langevin dynamics (Bussi & Parrinello, 2007; Welling & Teh, 2011) or Hamiltonian Monte Carlo (MacKay, 2003)—in which case a sample of particles from a base distribution is perturbed until it resembles a sample from the target distribution. It can also be applied in the training of parametric models, which either perform these perturbations under the hood, such as in the case of normalizing flows (Rezende & Mohamed, 2015; Papamakarios et al., 2021), or during inference, such as in the case of diffusion models (Ho et al., 2020; Nichol & Dhariwal, 2021).

This perspective would seem to contrast with approaches in which model parameters are optimized to align the generative and target distributions by reducing a loss quantifying the mismatch between them. Perhaps the most famous example of such an approach is a generative adversarial network (GAN) (Goodfellow et al., 2014), which leverages a separately trained discriminator to assess and guide this source-target alignment. However, we will show that such approaches contain a hidden sub-problem that corresponds to inducing a flow on the generated data that is determined by the loss being optimized. This reduces the task of understanding a wide variety of IGM methods to that of analyzing the dynamics imposed upon the synthetic or generative distribution.

The question then becomes one of asking what the *optimal* dynamics might be. In this direction, we introduce the *score difference* (SD)—the difference in the gradients of the log-densities of the target and source data distributions with respect to the data—as the flow direction that optimally reduces the KL divergence between them at each step. We then argue that we can sidestep directly working with the target and source distributions in favor of operating on convenient proxy distributions with common support, since aligning the proxies is equivalent to aligning the original, unmodified distributions.

We derive the score difference from the analysis of the dynamical systems that govern probability flow. But we also show that the score difference is hidden within or relates to various other IGM approaches under certain conditions, most notably GANs and denoising diffusion models. We also outline a flexible algorithmic approach for leveraging SD flow when working with *any* distributions p and q, with no restrictions placed on either distribution.

The paper is organized around its key contributions as follows:

1. In Section 2, we derive the score-difference (SD) flow from the study of *probability flow* dynamical systems and show that SD flow *optimally* reduces the Kullback-Leibler (KL) divergence between the source and target distributions.
2. In Section 3, we consider modified proxy distributions for the source and target distributions, which have common support and are generally easier to manage and estimate than the unmodified distributions. We outline a method for aligning these proxies and show that this alignment occurs if and only if the unmodified distributions are aligned.
3. In Section 4, we show that GAN generator training is composed of two main steps, a *data-optimization* step that induces a flow determined by the loss being optimized and a *model-optimization* step, in which the flow-perturbed particles are fit by the generator via regression. This allows us to specify the evolution of the generative distribution under any loss. We also show that the SD flow is induced in GANs under certain conditions and choices of loss function.
4. In Section 5, we draw a connection between SD flow and denoising diffusion models and show that they are equivalent under certain conditions. However, unlike diffusion models, SD flow places no restrictions on the prior distribution.

We offer concluding remarks in Section 6. In the appendices we provide algorithms for applying SD flow, draw a link to maximum mean discrepancy (MMD) gradient flow, and report several experiments.

2 Probability Flow and the Score Difference

2.1 Derivation from Stochastic Differential Equations

Consider data $\boldsymbol{x} \in \mathbb{R}^d$ drawn from a base distribution $q = q_0$. We can describe a dynamical system that perturbs the data and evolves its distribution $q_0 \to q_t$ over time by the stochastic differential equation

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{\mu}(\boldsymbol{x}, t)\mathrm{d}t + \sigma(t)\mathrm{d}\boldsymbol{\omega}, \tag{1}$$

where $\boldsymbol{\mu} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is a *drift coefficient*, $\sigma(t)$ is a *diffusion coefficient*, and $\mathrm{d}\boldsymbol{\omega}$ denotes the standard Wiener process (Song et al., 2020).

When $\boldsymbol{\mu}(\boldsymbol{x}, t) = \frac{\gamma(t)}{2}\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$ and $\sigma(t) = \sqrt{\gamma(t)}$, equation 1 describes *Langevin dynamics*, which for a suitable decreasing noise schedule $\gamma(t)$ can be shown to produce samples from a target distribution p as $t \to \infty$ (Bussi & Parrinello, 2007; Welling & Teh, 2011). That is, $q_\infty = p$.

A result due to Anderson (1982) shows that the dynamics in equation 1 can be *reversed*, effectively undoing the evolution of q to p. These reverse dynamics are given by

$$\mathrm{d}\boldsymbol{x} = \left[\boldsymbol{\mu}(\boldsymbol{x}, t) - \sigma(t)^2\nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})\right]\mathrm{d}t + \sigma(t)\mathrm{d}\hat{\boldsymbol{\omega}}, \tag{2}$$

where now $\mathrm{d}t$ is a *negative* time step, and $\hat{\boldsymbol{\omega}}$ is a time-reversed Wiener process. The reverse of a diffusion process is therefore another diffusion process.

A remarkable recent result shows that there is a *deterministic* process with the same marginal densities as those prescribed by equation 2. The corresponding dynamics are given by the *probability flow* ODE (Song et al., 2020)

$$\mathrm{d}\boldsymbol{x} = \left[\boldsymbol{\mu}(\boldsymbol{x}, t) - \frac{\sigma(t)^2}{2}\nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})\right]\mathrm{d}t. \tag{3}$$

If we substitute in the Langevin drift and diffusion terms from above, equation 3 becomes

$$\mathrm{d}\boldsymbol{x} = \frac{\gamma(t)}{2}\left[\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})\right] \mathrm{d}t. \tag{4}$$

Since $\mathrm{d}t$ is assumed to be a negative time step in the reverse process, equation 4 as written provides the dynamics of the *forward* process—pushing q_t *toward* the target distribution p—when $\mathrm{d}t$ is positive. Equation 4 represents the *score-difference* (SD) flow of a probability distribution q_t evolving toward p (or away from p, depending on the sign). Note that this is no longer a diffusion but rather defines a deterministic trajectory.

2.2 SD Flow Optimally Reduces KL Divergence

A continuous flow $\mathrm{d}\boldsymbol{x} = \mathbf{f}(\boldsymbol{x})\mathrm{d}t$ can be approximated by defining a transformation $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{x} + \varepsilon\mathbf{f}(\boldsymbol{x})$ for some small step $\varepsilon > 0$.[1] If $\boldsymbol{x} \sim q$ and $\boldsymbol{x}' = \boldsymbol{T}(\boldsymbol{x}) \sim q_{[\boldsymbol{T}]}$, then Liu & Wang (2016) show that the Kullback-Leibler (KL) divergence between $q_{[\boldsymbol{T}]}$ and p,

$$\mathbb{D}_{\mathrm{KL}}(q_{[\boldsymbol{T}]} \| p) = \mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\log q_{[\boldsymbol{T}]}(\boldsymbol{x}) - \log p(\boldsymbol{x})\right], \tag{5}$$

varies according to its functional derivative,

$$\nabla_{\varepsilon} \mathbb{D}_{\mathrm{KL}}(q_{[\boldsymbol{T}]} \| p)|_{\varepsilon=0} = -\mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\mathrm{Tr}\left(\mathcal{A}_p \mathbf{f}(\boldsymbol{x})\right)\right], \tag{6}$$

where

$$\mathcal{A}_p \mathbf{f}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \mathbf{f}(\boldsymbol{x})^\top + \nabla_{\boldsymbol{x}} \mathbf{f}(\boldsymbol{x}) \tag{7}$$

is the *Stein operator* (Gorham & Mackey, 2015).

By applying Stein's identity to equation 6, we obtain

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\mathrm{Tr}\left(\mathcal{A}_p \mathbf{f}(\boldsymbol{x})\right)\right] &= \mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})^\top \mathbf{f}(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\nabla_{\boldsymbol{x}} \log q_{[\boldsymbol{T}]}(\boldsymbol{x})^\top \mathbf{f}(\boldsymbol{x})\right] \\
&= \mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\left(\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_{[\boldsymbol{T}]}(\boldsymbol{x})\right)^\top \mathbf{f}(\boldsymbol{x})\right],
\end{aligned} \tag{8}$$

which is the inner product of the score difference and the flow vector $\mathbf{f}(\boldsymbol{x})$. Maximizing the *reduction* in the KL divergence (equation 6) corresponds to maximizing this inner product. Since the inner product of two vectors is maximized when they are parallel, choosing $\mathbf{f}(\boldsymbol{x})$ to be parallel to the score difference will decrease the KL divergence as fast as possible. We can also see from equation 6 and equation 8 that the decrease in the KL divergence is then proportional to the *Fisher divergence*,

$$\mathbb{D}_{\mathrm{F}}(q_{[\boldsymbol{T}]} \| p) = \mathbb{E}_{\boldsymbol{x} \sim q_{[\boldsymbol{T}]}}\left[\|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_{[\boldsymbol{T}]}(\boldsymbol{x})\|^2\right], \tag{9}$$

which, never being negative, shows that moving along the SD flow in sufficiently small steps will not increase the KL divergence.

3 Applying SD Flow to Proxy Distributions

One of the difficulties in applying Langevin dynamics or other score-based methods is the requirement that we have access to the true score of the target distribution, $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$, which is almost never available in practice. It is also the case that when operating in the ambient space of $\boldsymbol{x} \in \mathbb{R}^d$, the score may not be well defined in areas of limited support if the data exist on a lower-dimensional manifold, which is generally assumed for a variety of data types of interest, such as image data. A large literature has emerged that is dedicated to the estimation of this score or the design of training procedures that are equivalent to estimating it (Hyvärinen & Dayan, 2005; Song & Ermon, 2020; Song & Kingma, 2021; Karras et al., 2022).

Applying SD flow would appear to be at least twice as difficult, since instead of one score to estimate, now we have two. The distribution q_t is also changing over time, so even a reasonably good estimate at one time would have to be discarded and re-estimated at another. Our approach will be to essentially ignore p and q_t and work instead with modified proxy distributions that are easier to estimate and manage. Importantly, aligning these proxy distributions will automatically align the unmodified source and target distributions.

---

[1] If ε is sufficiently small, then the Jacobian of $\boldsymbol{T}$ is of full rank, meaning that the transformation is bijective.

3.1 Aligning Proxy Distributions

We can assess the alignment of two distributions q and p by computing a *statistical distance* between them.[2] Although this quantity is not always a true "distance" in a strict mathematical sense, it will have the two following properties:

1. $\mathbb{D}(q\|p) \geq 0$ for all distributions p, q
2. $\mathbb{D}(q\|p) = \mathbb{D}(p\|q) = 0 \iff p = q$

Perhaps the best-known statistical distance is the KL divergence, $\mathbb{D}_{\mathrm{KL}}(q\|p)$ (equation 5), although it can diverge to infinity if p and q have unequal support.

One way to equalize the support of two distributions is to corrupt their data with additive noise defined over the whole space $\mathbb{R}^d$. Let us assume a Gaussian noise model. The distribution of $\boldsymbol{z} = \boldsymbol{x} + \sigma\boldsymbol{\epsilon}$, with $\boldsymbol{x} \sim p$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, is given by the convolution $p * \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$:

$$\begin{aligned}\tilde{p}_\sigma(\boldsymbol{z}) = p(\boldsymbol{z};\sigma) &= \int_{\mathbb{R}^d} p(\boldsymbol{x})\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\,\mathrm{d}\boldsymbol{x} \\ &= \mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\right],\end{aligned} \tag{10}$$

with $\tilde{q}_\sigma(\boldsymbol{z}) = q(\boldsymbol{z};\sigma) = \mathbb{E}_{\boldsymbol{y}\sim q}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{y},\sigma^2\boldsymbol{I})\right]$ defined analogously. Although $\mathbb{D}_{\mathrm{KL}}(\tilde{q}_\sigma\|\tilde{p}_\sigma) \leq \mathbb{D}_{\mathrm{KL}}(q\|p)$ and $\mathbb{D}_{\mathrm{KL}}(\tilde{q}_\sigma\|\tilde{p}_\sigma) \to 0$ as $\sigma \to \infty$ (Sriperumbudur et al., 2017), it is easy to show that $\mathbb{D}_{\mathrm{KL}}(\tilde{q}_\sigma\|\tilde{p}_\sigma) = 0$ if and only if $q = p$ (Zhang et al., 2020). As a result, aligning the proxy distributions $\tilde{q}_\sigma$ and $\tilde{p}_\sigma$ is equivalent to aligning the generative distribution q with the target distribution p. Since we have shown that moving parallel to the score difference optimally reduces the KL divergence, we will define an SD flow between $\tilde{q}_\sigma$ and $\tilde{p}_\sigma$ to align q with p.

The score corresponding to equation 10 is given by

$$\begin{aligned}\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z};\sigma) &= \frac{\nabla_{\boldsymbol{z}} p(\boldsymbol{z};\sigma)}{p(\boldsymbol{z};\sigma)} \\ &= \frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\nabla_{\boldsymbol{z}}\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\right]} \\ &= \frac{1}{\sigma^2}\left(\frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\boldsymbol{x}\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\right]} - \boldsymbol{z}\right).\end{aligned} \tag{11}$$

The score for $q(\boldsymbol{z};\sigma)$ is derived in the same way for $\boldsymbol{z} = \boldsymbol{y} + \sigma\boldsymbol{\epsilon}$, with $\boldsymbol{y} \sim q$. This leads to the following expression for the score difference:

$$\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z};\sigma) - \nabla_{\boldsymbol{z}} \log q(\boldsymbol{z};\sigma) = \frac{1}{\sigma^2}\left(\frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_\sigma(\boldsymbol{z},\boldsymbol{x})\boldsymbol{x}\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_\sigma(\boldsymbol{z},\boldsymbol{x})\right]} - \frac{\mathbb{E}_{\boldsymbol{y}\sim q}\left[K_\sigma(\boldsymbol{z},\boldsymbol{y})\boldsymbol{y}\right]}{\mathbb{E}_{\boldsymbol{y}\sim q}\left[K_\sigma(\boldsymbol{z},\boldsymbol{y})\right]}\right), \tag{12}$$

where $K_\sigma(\boldsymbol{z},\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{z}-\boldsymbol{x}\|^2}{2\sigma^2}\right)$ is the Gaussian kernel.[3]

If we set the noise level according to the schedule $\sigma^2 = \gamma(t)$, then the variance term cancels from equation 4, leading to the flow

$$\mathrm{d}\boldsymbol{z} = \frac{1}{2}\left[\frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_\sigma(\boldsymbol{z},\boldsymbol{x})\boldsymbol{x}\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_\sigma(\boldsymbol{z},\boldsymbol{x})\right]} - \frac{\mathbb{E}_{\boldsymbol{y}\sim q}\left[K_\sigma(\boldsymbol{z},\boldsymbol{y})\boldsymbol{y}\right]}{\mathbb{E}_{\boldsymbol{y}\sim q}\left[K_\sigma(\boldsymbol{z},\boldsymbol{y})\right]}\right]\mathrm{d}t. \tag{13}$$

Since $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{z} = \mathrm{d}\boldsymbol{y}/\mathrm{d}\boldsymbol{z} = \boldsymbol{I}$, the prescribed dynamics for the clean data are the same as for the corrupted data, and we only need to keep track of the sign implied by the forward or reverse process. If we are moving a point $\boldsymbol{y} \sim q$ toward p, then $\mathrm{d}\boldsymbol{y} = \mathrm{d}\boldsymbol{z}$, while if we were moving a point $\boldsymbol{x} \sim p$ toward q, then $\mathrm{d}\boldsymbol{x} = -\mathrm{d}\boldsymbol{z}$. We test this formulation of SD flow in Appendix E.

---

[2] We will suppress the time index on q here for convenience.

[3] This substitution is a useful simplification because the normalization constant of the normal distribution cancels from the numerator and denominator. It also makes the later comparison to MMD gradient flow (Appendix D) more straightforward.

3.2 Alternative Formulations of SD Flow

In the limit of infinite data, equation 12 is exact. But applying this formulation in a large-data setting can be computationally expensive, and estimates using smaller batches may suffer from low accuracy, especially in high dimensions. It is useful to consider each term of equation 12 as a *module* that can be swapped out for another method, depending on the problem setting. We provide several options in Algorithms 1 and 2.

As one example, we can rewrite equation 12 as

$$\begin{aligned}\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z};\sigma) - \nabla_{\boldsymbol{z}} \log q(\boldsymbol{z};\sigma) &= \frac{1}{\sigma^2}\left[\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}] - \mathbb{E}[\boldsymbol{y}|\boldsymbol{z}]\right] && (14)\\ &= \frac{1}{\sigma^2}\left[D_p^*(\boldsymbol{z};\sigma) - D_{q_t}^*(\boldsymbol{z};\sigma)\right], && (15)\end{aligned}$$

where $D_p^*(\boldsymbol{z};\sigma)$ and $D_{q_t}^*(\boldsymbol{z};\sigma)$ are the *optimal* denoising models for the distributions p and q_t, respectively, when corrupted by Gaussian noise at level σ. A simple derivation of this result is possible by rearranging Tweedie's formula (Efron, 2011),[4] but we provide a separate proof of optimality in Appendix B.

The denoiser corresponding to the target data would need to be trained only once, while the denoiser for the generative distribution would, at least in principle, need to be retrained after each step along the flow. However, since we actually observe $\boldsymbol{y} \sim q_t$ before corrupting it to form $\boldsymbol{z} = \boldsymbol{y} + \sigma\boldsymbol{\epsilon}$, we can just replace $D_{q_t}^*(\boldsymbol{z};\sigma) = \mathbb{E}[\boldsymbol{y}|\boldsymbol{z}]$ with $\boldsymbol{y}$ in equation 15,[5] leading to the update

$$\boldsymbol{y} \leftarrow (1-\rho)\boldsymbol{y} + \rho D_p^*(\boldsymbol{z};\sigma) \tag{16}$$

for some small step size ρ, which, as $\rho \to 0$, corresponds to the continuous dynamics $\mathrm{d}\boldsymbol{y} = D_p^*(\boldsymbol{z};\sigma)\mathrm{d}t$. In Section 5, we show that this formulation is equivalent to the reverse process in denoising diffusion models.

4 Flows in Generative Adversarial Networks

4.1 Decomposing Generative Modeling into Sub-problems

When training a generative model $g_{\boldsymbol{\theta}}$, we define a loss $\mathcal{L}$, which is a scalar function of the generator output that quantifies a discrepancy between the current model output and the target distribution. We treat this loss as a function of the parameters $\boldsymbol{\theta} \in \mathbb{R}^n$ and then optimize $\boldsymbol{\theta}$ to minimize $\mathcal{L}$ via gradient[6] descent at some learning rate $\eta > 0$:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}\right)^\top. \tag{17}$$

However, the loss $\mathcal{L}$ is also a function of the generated data $\tilde{\boldsymbol{x}} = g_{\boldsymbol{\theta}}(\boldsymbol{\xi}) \in \mathbb{R}^d$, which is *itself* a function of either $\boldsymbol{\xi} \in \mathbb{R}^l$ or $\boldsymbol{\theta} \in \mathbb{R}^n$, depending on our perspective. This perspective can be made explicit by decomposing the derivative of the loss via the multivariate chain rule,

$$\underbrace{\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}}_{1\times n} = \underbrace{\frac{\partial\mathcal{L}}{\partial\tilde{\boldsymbol{x}}}}_{1\times d}\underbrace{\frac{\partial\tilde{\boldsymbol{x}}}{\partial\boldsymbol{\theta}}}_{d\times n}. \tag{18}$$

This allows us to consider each component of the decomposition as corresponding to its own sub-problem.

In the **first sub-problem**, we perturb the generated data $\tilde{\boldsymbol{x}}$ in the direction of the negative gradient,

$$\tilde{\boldsymbol{x}}' = \tilde{\boldsymbol{x}} - \lambda_1\left(\frac{\partial\mathcal{L}}{\partial\tilde{\boldsymbol{x}}}\right)^\top, \tag{19}$$

---

[4] Tweedie's formula states that $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}] = \boldsymbol{z} + \sigma^2\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z};\sigma)$,

[5] Note that this is an approximation, since $D_{q_t}^*(\boldsymbol{z};\sigma) = \mathbb{E}[\boldsymbol{y}|\boldsymbol{z}]$ will not necessarily equal $\boldsymbol{y}$.

[6] We treat the gradient as the transpose of the derivative.

where $\lambda_1 > 0$ is some small step size. Intuitively, the perturbed data $\tilde{\boldsymbol{x}}'$ corresponds to a potential output of the generator that would have a lower loss, but we can also interpret it as resulting from a gradient flow on the synthetic data.

In the **second sub-problem**, we update the generator parameters $\boldsymbol{\theta}$ by regressing the new, perturbed target $\tilde{\boldsymbol{x}}'$ on the original generator input $\boldsymbol{\xi}$ via the least-squares loss

$$\mathcal{J} = \frac{1}{2}\|g_{\boldsymbol{\theta}}(\boldsymbol{\xi}) - \tilde{\boldsymbol{x}}'\|^2 = \frac{1}{2}\|\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}'\|^2. \tag{20}$$

Putting the pieces together leads to

$$\begin{aligned}
\boldsymbol{\theta}' &= \boldsymbol{\theta} - \lambda_2 \left(\frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}\right)^{\top} &(21)\\
&= \boldsymbol{\theta} - \lambda_2 \left(\frac{\partial g_{\boldsymbol{\theta}}(\boldsymbol{\xi})}{\partial \boldsymbol{\theta}}\right)^{\top} (g_{\boldsymbol{\theta}}(\boldsymbol{\xi}) - \tilde{\boldsymbol{x}}') &(22)\\
&= \boldsymbol{\theta} - \lambda_2 \left(\frac{\partial \tilde{\boldsymbol{x}}}{\partial \boldsymbol{\theta}}\right)^{\top} (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}') &(23)\\
&= \boldsymbol{\theta} - \lambda_1\lambda_2 \left(\frac{\partial \tilde{\boldsymbol{x}}}{\partial \boldsymbol{\theta}}\right)^{\top} \left(\frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{x}}}\right)^{\top} &(24)\\
&= \boldsymbol{\theta} - \lambda_1\lambda_2 \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)^{\top}, &(25)
\end{aligned}$$

which is is equal to the standard gradient update of $\boldsymbol{\theta}$ under the original loss $\mathcal{L}$ (equation 17) with step size $\eta = \lambda_1\lambda_2$. Here equation 24 follows from equation 23 by rearranging and substituting equation 19, while equation 25 follows from equation 24 via equation 18.

Although this decomposition is a direct consequence of gradient descent, it shows that hidden within generator training are two sub-problems with separate control options (their learning rates, for instance), each of which may be easier to conceptualize and handle than the original problem. In particular, we see that the model-optimization step of generator training is preceded, at least implicitly, by a data-optimization step that prescribes a flow in the ambient data space $\mathbb{R}^d$, regardless of the overall loss being optimized. This suggests that we can treat this data-optimization step as a *target-generation* module that can be swapped out in favor of other procedures. Furthermore, it means that a wide variety of generative models can be understood in terms of the dynamics imposed on the generated data during training.

4.2 GAN Dynamics in General

Since the negative gradient of the GAN loss defines a flow on the generated data $\tilde{\boldsymbol{x}}$, which the generator fits via regression, we can track the evolution of the synthetic distribution q_t within the context of *normalizing flows* (Rezende & Mohamed, 2015; Papamakarios et al., 2021). In particular, the dynamics induced by a generator loss constitute, in the limit of arbitrarily small steps, a *continuous normalizing flow* whose effect on the synthetic (generated) data distribution is governed by the *Liouville equation* (Ehrendorfer, 1994),

$$\frac{\partial q_t(\tilde{\boldsymbol{x}}_t)}{\partial t} = \nabla_{\tilde{\boldsymbol{x}}_t} \cdot \left[q_t(\tilde{\boldsymbol{x}}_t)\nabla_{\tilde{\boldsymbol{x}}}\mathcal{L}\right], \tag{26}$$

a continuity equation with solution

$$q_t(\tilde{\boldsymbol{x}}_t) = q_0(\tilde{\boldsymbol{x}}_0)\exp\left(\int_0^t \operatorname{Tr}\left[\boldsymbol{H}_{\mathcal{L}}(\tilde{\boldsymbol{x}}_\tau)\right] \mathrm{d}\tau\right) = q_0(\tilde{\boldsymbol{x}}_0)\exp\left(\int_0^t \nabla^2_{\tilde{x}_\tau}\mathcal{L}\,\mathrm{d}\tau\right), \tag{27}$$

where $\boldsymbol{H}_{\mathcal{L}}(\tilde{\boldsymbol{x}}_\tau)$ is the Hessian matrix of the loss $\mathcal{L}$ evaluated at $\tilde{\boldsymbol{x}}_\tau$, whose trace is the Laplacian $\nabla^2_{\tilde{x}}\mathcal{L} = \sum_i \partial^2/\partial\tilde{x}_i{}^2\mathcal{L}$.[7] Taking the logarithm of both sides of equation 27 yields the solution to the *instantaneous change of variables* differential equation (Chen et al., 2018; Grathwohl et al., 2018). Note that this evolution holds for any generator loss $\mathcal{L}$ and does not make any assumptions about an optimal discriminator.

---

[7] Many authors denote the Laplacian by Δ, but we have reserved its use for discrete-time differences.

4.3 SD Flow in GANs

Many GAN generators employ the widely used *non-saturating loss* (Goodfellow, 2016) given by

$$\mathcal{L}(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{\xi}\sim p_0}\left[\log f(g_{\boldsymbol{\theta}}(\boldsymbol{\xi}))\right], \tag{28}$$

where $\boldsymbol{\xi} \in \mathbb{R}^l$ is a random noise input to the generator drawn from a prior distribution p_0 and $f : \mathbb{R}^d \to (0,1)$ is a separately trained discriminator that estimates the probability that its argument is real data coming from a target distribution p (in which case $f \approx 1$), as opposed to fake data coming from the generator distribution q_t (in which case $f \approx 0$). Intuitively, the aim of the loss given by equation 28 is to tune the parameters $\boldsymbol{\theta}$ to *maximize* the discriminator's assessment of generated data as real.

It can be shown that, if the prior probabilities of coming from either p or q_t are equal, the *Bayes optimal* classifier f_t is given by

$$f_t(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{p(\boldsymbol{x}) + q_t(\boldsymbol{x})}, \tag{29}$$

where we have included the time subscript to indicate the optimal discriminator's dependence on the changing distribution q_t. An optimal discriminator is often assumed in the analysis of GANs but almost never holds in actual practice.

When implemented as a neural network, the discriminator f_t usually terminates with a *sigmoid* activation,

$$f_t(\boldsymbol{x}) = \frac{1}{1 + \exp[-h_t(\boldsymbol{x})]}, \tag{30}$$

where $h_t(\boldsymbol{x})$ is the *pre-activation* output of the discriminator f_t. Equating equation 29 and equation 30, we see that in the case of an optimal discriminator, $h_t(\boldsymbol{x}) = \log p(\boldsymbol{x}) - \log q_t(\boldsymbol{x})$, whose gradient is the SD flow,

$$\nabla_{\boldsymbol{x}} h_t(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x}). \tag{31}$$

When trained using the non-saturating loss (equation 28),

$$-\nabla_{\boldsymbol{x}}\mathcal{L} = [1 - f_t(\boldsymbol{x})]\nabla_{\boldsymbol{x}} h_t(\boldsymbol{x}).$$

Since $[1 - f_t(\boldsymbol{x})] > 0$ for all $\boldsymbol{x}$, taking the results of Section 4.1 into account we see that standard GAN training with an optimal discriminator consistently pushes the generated data toward the target data in a direction parallel to the score difference (equation 4). We can also consider an alternative to the non-saturating loss that focuses on the pre-activation output $h_t(\boldsymbol{x})$:

$$\mathcal{L}^{\text{alt}} = -\mathbb{E}_{\boldsymbol{\xi}\sim p_0}\left[h_t(g_{\boldsymbol{\theta}}(\boldsymbol{\xi}))\right] = -\mathbb{E}_{\tilde{\boldsymbol{x}}\sim q_t}[h_t(\tilde{\boldsymbol{x}})], \tag{32}$$

which induces a gradient flow on $\tilde{\boldsymbol{x}} \sim q_t$ exactly equal to the SD flow when the discriminator is optimal.

5 Relation to Denoising Diffusion Models

In diffusion modeling, data from the target distribution p is corrupted in a forward diffusion process by Gaussian noise under the scale and noise schedules α_t and σ_t, respectively. Then for $\boldsymbol{z}_0 = \boldsymbol{x} \sim p$, the conditional distribution at time t relative to that at time $s < t$ is given by

$$p(\boldsymbol{z}_t|\boldsymbol{z}_s) = \mathcal{N}(\alpha_{t|s}\boldsymbol{z}_s, \sigma_{t|s}^2\boldsymbol{I}),$$

where $\alpha_{t|s} = \alpha_t/\alpha_s$ and $\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2\sigma_s^2$ (Kingma et al., 2021).

The hard part is inferring the *reverse* diffusion process, $p(\boldsymbol{z}_s|\boldsymbol{z}_t)$, which is intractable unless also conditioned on $\boldsymbol{z}_0 = \boldsymbol{x}$: $p(\boldsymbol{z}_s|\boldsymbol{z}_t, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_{s|t}, \sigma_{s|t}^2)$, where $\boldsymbol{\mu}_{s|t} = (\alpha_{t|s}\sigma_s^2/\sigma_t^2)\boldsymbol{z}_t + (\alpha_s\sigma_{t|s}^2/\sigma_t^2)\boldsymbol{x}$ and $\sigma_{s|t}^2 = \sigma_{t|s}^2\sigma_s^2/\sigma_t^2$. In practice, $\boldsymbol{x}$ is replaced by $D(\boldsymbol{z}_t; \sigma_t)$, the output of a denoising model.[8]

---

[8] In alternative but equivalent implementations, the error between $\boldsymbol{x}$ and $\boldsymbol{z}_t$ is predicted by a parametric model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_t; t)$.

If we let $\alpha_s = \alpha_t = 1$ for all s, t, then

$$\begin{aligned} \boldsymbol{z}_s &= \frac{\sigma_s^2}{\sigma_t^2}\boldsymbol{z}_t + \left(1 - \frac{\sigma_s^2}{\sigma_t^2}\right) D(\boldsymbol{z}_t; \sigma_t) + \sigma_{s|t}\boldsymbol{\epsilon}_s \qquad (33) \\ &= (1-\rho)\boldsymbol{z}_t + \rho D(\boldsymbol{z}_t; \sigma_t) + \sqrt{\rho}\sigma_s\boldsymbol{\epsilon}_s \qquad (34) \end{aligned}$$

for $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\rho = 1 - \sigma_s^2/\sigma_t^2$. Recalling that in the SD flow framework, $\boldsymbol{z}_t = \boldsymbol{y}_t + \sigma_t\boldsymbol{\epsilon}_t \sim q_t(\boldsymbol{z}_t; \sigma_t)$ for all t, equation 34 becomes

$$\begin{aligned} \boldsymbol{z}_s &= (1-\rho)\boldsymbol{y}_t + \rho D(\boldsymbol{z}_t; \sigma_t) + \sqrt{\rho}\sigma_s\boldsymbol{\epsilon}_s + (1-\rho)\sigma_t\boldsymbol{\epsilon}_t \\ &= (1-\rho)\boldsymbol{y}_t + \rho D(\boldsymbol{z}_t; \sigma_t) + \sqrt{\rho\sigma_s^2 + (1-\rho)^2\sigma_t^2}\,\boldsymbol{\epsilon} \\ &= \underbrace{(1-\rho)\boldsymbol{y}_t + \rho D(\boldsymbol{z}_t; \sigma_t)}_{\boldsymbol{y}_s} + \underbrace{\sqrt{\left(1 - \frac{\sigma_s^2}{\sigma_t^2}\right)\sigma_s^2 + \left(\frac{\sigma_s^2}{\sigma_t^2}\right)^2\sigma_t^2}}_{\sigma_s}\,\boldsymbol{\epsilon} \qquad (35) \\ &= \boldsymbol{y}_s + \sigma_s\boldsymbol{\epsilon}, \end{aligned}$$

where $\boldsymbol{\epsilon}, \boldsymbol{\epsilon}_s, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{y}_s$ follows from equation 16. In other words, the updating process under SD flow is equivalent to the denoising diffusion reverse process under the substitution described in Section 3.2.

6 Discussion and Concluding Remarks

In this work, we introduced the *score-difference* (SD) flow and showed that it corresponds to an optimal trajectory for aligning a source (synthetic) distribution with a target distribution. We also showed that this alignment can be performed by working entirely with proxy distributions formed by smoothing the data with additive noise. As a result, while the SD flow defines a deterministic trajectory, its application to noise-injected points adds a stochastic component.

Unlike most other score-based methods, there are no restrictions on the choice of base or prior distributions. In this way, SD flow is capable of performing data-set interpolation in a manner similar to diffusion Schrödinger bridges (De Bortoli et al., 2021) while being more straightforward to apply.[9] Technical details aside, the score difference is an intuitive trajectory to follow: If our goal is to get from point A to point B as quickly as possible, we want to move not only toward B but also *away from* A.

As the difference in scores between the target and (current) synthetic distribution, the score difference is equal to the gradient of the logarithm of the *density ratio*. The density ratio is an important quantity in statistics that has a considerable literature dedicated to its estimation (Sugiyama et al., 2012; Rhodes et al., 2020; Choi et al., 2022, for example). Beyond the applications to implicit generative modeling that are the focus of this paper, we hope that this work inspires additional advances in the estimation and leveraging of this important quantity.

We have shown that SD flow emerges in both GANs and diffusion models under certain conditions. The conditions for GANs include the presence of an *optimal* discriminator, which is often unattainable when training with finite, categorically labeled data. By contrast, diffusion models have the comparatively easier task of learning to denoise an image, a task for which the ground truth is more readily represented in the training data. Modern neural denoising architectures that employ attention provide another edge, since they have shown themselves to be extraordinarily capable of capturing patterns in data due to their error-correction and pattern-retrieval characteristics reminiscent of Hopfield networks (Ramsauer et al., 2020).

SD flow supplies a link between IGM methods that collectively perform well on all three desiderata of the so-called *generative learning trilemma* (Xiao et al., 2021)—high sample quality, mode coverage, and fast sampling. We hope that this work provides a useful advance toward the development of "triple threat" models that produce high-quality, diverse output in a single inference step.

---

[9] See Appendix E.3.

References

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Michael Arbel, Anna Korba, Adil Salim, and Arthur Gretton. Maximum mean discrepancy gradient flow. *Advances in Neural Information Processing Systems*, 32, 2019.

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*, pp. 217–235. Springer, 1999.

Giovanni Bussi and Michele Parrinello. Accurate sampling using langevin dynamics. *Physical Review E*, 75 (5):056707, 2007.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pp. 2552–2573. PMLR, 2022.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34: 17695–17709, 2021.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106 (496):1602–1614, 2011.

Martin Ehrendorfer. The liouville equation and its potential usefulness for the prediction of forecast skill. part i: Theory. *Monthly Weather Review*, 122(4):703–713, 1994.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Jackson Gorham and Lester Mackey. Measuring sample quality with stein's method. *Advances in Neural Information Processing Systems*, 28, 2015.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

David JC MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

Henry P McKean Jr. A class of markov processes associated with nonlinear parabolic equations. *Proceedings of the National Academy of Sciences*, 56(6):1907–1911, 1966.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

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.

Bharath Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Aapo Hyvärinen, and Revant Kumar. Density estimation in infinite dimensional exponential families. *Journal of Machine Learning Research*, 18, 2017.

Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12(7), 2011.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning.* Cambridge University Press, 2012.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib, and David Barber. Spread divergence. In *International Conference on Machine Learning*, pp. 11106–11116. PMLR, 2020.

A Guide to the Appendices

In Appendix B, we show that the score difference corresponds to the difference between the outputs of *optimal* denoisers corresponding to the target (p) and current synthetic (q_t) distributions. In Appendix C, we provide flexible pseudocode algorithms for applying SD flow in both the direct sampling (*data-optimization*) and parametric generative modeling (*model-optimization*) settings. In Appendix D, we draw a connection between the kernel definition of SD flow and maximum mean discrepancy (MMD) gradient flow (Arbel et al., 2019). In Appendix E, we report several experiments in both the data- and model-optimization settings and also demonstrate SD flow's ability to interpolate between arbitrary distributions.

B The Score Difference as the "Denoiser Difference"

We follow an approach similar to that found in Appendix B.3 of Karras et al. (2022) to derive the optimal denoiser for p. We define the *denoising loss* by

$$\begin{aligned}\mathcal{E}(D_p;\sigma) &= \mathbb{E}_{\boldsymbol{x}\sim p}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\|D_p(\boldsymbol{x}+\boldsymbol{\epsilon};\sigma)-\boldsymbol{x}\|^2\right]\\ &= \mathbb{E}_{\boldsymbol{x}\sim p}\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{x},\sigma^2\boldsymbol{I})}\left[\|D_p(\boldsymbol{z};\sigma)-\boldsymbol{x}\|^2\right]\\ &= \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{x},\sigma^2\boldsymbol{I})}\mathbb{E}_{\boldsymbol{x}\sim p}\left[\|D_p(\boldsymbol{z};\sigma)-\boldsymbol{x}\|^2\right]\\ &= \int_{\mathbb{R}^d}\underbrace{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\|D_p(\boldsymbol{z};\sigma)-\boldsymbol{x}\|^2\right]}_{\mathcal{E}(D_p;\boldsymbol{z},\sigma)}\mathrm{d}\boldsymbol{z}.\end{aligned} \tag{36}$$

We can optimize $\mathcal{E}(D_p;\sigma)$ by minimizing the integrand $\mathcal{E}(D_p;\boldsymbol{z},\sigma)$ pointwise. The optimal denoiser is then given by

$$D_p^*(\boldsymbol{z};\sigma) = \arg\min_{D_p(\boldsymbol{z};\sigma)}\mathcal{E}(D_p;\boldsymbol{z},\sigma), \tag{37}$$

which defines a convex optimization problem. We can then find the global minimum by setting $\nabla_{D_p(\boldsymbol{z};\sigma)}\mathcal{E}(D_p;\boldsymbol{z},\sigma)$ to zero, leading to

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\nabla_{D_p(\boldsymbol{z};\sigma)}\|D_p(\boldsymbol{z};\sigma)-\boldsymbol{x}\|^2\right] &= \boldsymbol{0}\\ 2D_p(\boldsymbol{z};\sigma)\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\right] &= 2\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\boldsymbol{x}\right]\\ D_p^*(\boldsymbol{z};\sigma) &= \frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\boldsymbol{x}\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[\mathcal{N}(\boldsymbol{z};\boldsymbol{x},\sigma^2\boldsymbol{I})\right]},\end{aligned} \tag{38}$$

the final term of which is equal to the first term inside the brackets in equation 13 when K_σ is the Gaussian kernel. The derivation for $D_{q_t}^*(\boldsymbol{z};\sigma)$ is identical, which leads to the result.

C Algorithms

Here we provide pseudocode algorithms for applying SD flow. Note that we present several interpretations of SD flow, providing the user with some flexibility to swap out one approach for another.

C.1 Data Optimization

Algorithm 1 Data optimization with SD flow

```
**Input:** Target data $\{\boldsymbol{x}_i\}_{i=1}^N \sim p$, base (prior) data $\{\boldsymbol{y}_j\}_{j=1}^M \sim q_0$, noise schedule $\sigma(t)$, step schedule $\eta(t)$
repeat
  Draw data batches $\boldsymbol{x} \sim p$ and $\boldsymbol{y} \sim q_t$
  Draw noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ and perturb data: $\boldsymbol{z} = \boldsymbol{y} + \sigma(t)\boldsymbol{\epsilon}$
  # Calculate SD (equation 13, 15, or 31).
  $\Delta\boldsymbol{z} \propto \frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{x})\boldsymbol{x}\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{x})\right]} - \frac{\mathbb{E}_{\boldsymbol{y}\sim q_t}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{y})\boldsymbol{y}\right]}{\mathbb{E}_{\boldsymbol{y}\sim q_t}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{y})\right]} = D_p^*(\boldsymbol{z};\sigma(t)) - D_{q_t}^*(\boldsymbol{z};\sigma(t)) \propto \nabla_{\boldsymbol{z}}h_t(\boldsymbol{z})$
  # Move (clean) data in SD direction.
  $\boldsymbol{y} \leftarrow \boldsymbol{y} + \eta(t)\Delta\boldsymbol{z}$
until Terminated
```

We first focus on the *data-optimization* application (Algorithm 1), in which a sample is generated by perturbing a single batch of "base data," much like one would via Langevin dynamics or Hamiltonian Monte Carlo. Here we interpret the base data as being drawn from q_0 and evolving to the distribution q_t through iterative perturbation.

Algorithm 2 Model optimization with SD flow

```
Input: Target data $\{\boldsymbol{x}_i\}_{i=1}^N \sim p$, noise input for generator $\boldsymbol{\xi} \sim p_0$, initialized generator $g_{\boldsymbol{\theta}}$ ($\boldsymbol{\theta} \in \mathbb{R}^n$), noise schedule $\sigma(t)$, step schedule $\eta(t)$, learning rate schedule $\lambda(t)$
repeat
    Draw data batches $\boldsymbol{x} \sim p$ and $\boldsymbol{\xi} \sim p_0$
    Generate sample $\boldsymbol{y} = g_\theta(\boldsymbol{\xi})$
    Draw noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and perturb data: $\boldsymbol{z} = \boldsymbol{y} + \sigma(t)\boldsymbol{\epsilon}$
    # Calculate SD (equation 13, 15, or 31).
    $\Delta \boldsymbol{z} \propto \frac{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{x})\boldsymbol{x}\right]}{\mathbb{E}_{\boldsymbol{x}\sim p}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{x})\right]} - \frac{\mathbb{E}_{\boldsymbol{y}\sim q_t}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{y})\boldsymbol{y}\right]}{\mathbb{E}_{\boldsymbol{y}\sim q_t}\left[K_{\sigma(t)}(\boldsymbol{z},\boldsymbol{y})\right]} = D_p^*(\boldsymbol{z};\sigma(t)) - D_{q_t}^*(\boldsymbol{z};\sigma(t)) \propto \nabla_{\boldsymbol{z}} h_t(\boldsymbol{z})$
    # Move (clean) data in SD direction.
    $\boldsymbol{y} \leftarrow \boldsymbol{y} + \eta(t)\Delta\boldsymbol{z}$
    # Update generator parameters via regression.
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\lambda(t)}{2}\nabla_{\boldsymbol{\theta}}\|g_\theta(\boldsymbol{\xi}) - \boldsymbol{y}\|^2$
until Terminated
```

C.2 Model Optimization

Ordinary regression problems involve paired training data $\{(\boldsymbol{\xi}, \boldsymbol{x})\}$ that implicitly define the function mapping $g : \boldsymbol{\xi} \mapsto \boldsymbol{x}$ to be learned. In the case of generative modeling, where the aim is to map data from a prior distribution to a target distribution, no *a priori* pairing exists. This requires the mapping to be learned indirectly. In the case of GANs, the sub-problem interpretation of Section 4.1 is that the discriminator provides this pairing by associating a noise input $\boldsymbol{\xi}$ with a perturbed output $\boldsymbol{x}'$ that serves as a regression target.

The *model-optimization* application (Algorithm 2) trains a generator with unpaired data via regression on perturbed targets that move progressively closer to the target distribution. Here we interpret q_0 as the distribution of the output of the generator $g_{\boldsymbol{\theta}}$ before training. As the generator regresses on perturbed targets, its output distribution changes to q_t at time step t according to the evolution in equation 27. When the choice of SD flow is the gradient of the pre-activation discriminator output, $\nabla_{\boldsymbol{z}} h_t(\boldsymbol{z})$ (equation 31), this algorithm is equivalent to GAN training with the alternative loss described in equation 32.

D Relation to MMD Gradient Flow

In the study of reproducing kernel Hilbert spaces (RKHS), the Gaussian kernel $K_\sigma(\boldsymbol{z}, \boldsymbol{x})$ is known as a *characteristic* kernel (Sriperumbudur et al., 2011). This means that the mapping $\phi_p(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}\sim p}[K_\sigma(\boldsymbol{z}, \boldsymbol{x})]$ is *injective*, and $\phi_p(\boldsymbol{z}) = \phi_q(\boldsymbol{z})$ for all $\boldsymbol{z}$ if and only if $p = q$. This forms the basis of the *maximum mean discrepancy* (MMD), which is equal to the Hilbert space norm $\|\mathcal{W}_{p,q}\|_{\mathcal{H}}$, where

$$\mathcal{W}_{p,q}(\boldsymbol{z}) = \phi_q(\boldsymbol{z}) - \phi_p(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{y}\sim q}[K_\sigma(\boldsymbol{z}, \boldsymbol{y})] - \mathbb{E}_{\boldsymbol{x}\sim p}[K_\sigma(\boldsymbol{z}, \boldsymbol{x})] \tag{39}$$

is known as the *witness function* (Gretton et al., 2012; Arbel et al., 2019).

In the theory of optimal transport, we wish to efficiently transport "mass" from an initial distribution q_0 to a target distribution p, which we can do by defining a flow from q_0 to p via intermediate distributions q_t. One such flow is defined by the solution to

$$\frac{\partial q_t}{\partial t} = \nabla \cdot \left[q_t \nabla \mathcal{W}_{p,q_t}\right], \tag{40}$$

another instance of the Liouville equation (26) that defines a McKean-Vlasov process (McKean Jr, 1966) with dynamics

$$\begin{aligned} \mathrm{d}\boldsymbol{z}_t &= -\nabla_{\boldsymbol{z}_t}\mathcal{W}_{p,q_t}(\boldsymbol{z}_t)\,\mathrm{d}t \\ &= \left(\mathbb{E}_{\boldsymbol{x}\sim p}[\nabla_{\boldsymbol{z}_t}K_\sigma(\boldsymbol{z}_t, \boldsymbol{x})] - \mathbb{E}_{\boldsymbol{y}\sim q}[\nabla_{\boldsymbol{z}_t}K_\sigma(\boldsymbol{z}_t, \boldsymbol{y})]\right)\mathrm{d}t, \end{aligned} \tag{41}$$

where $\boldsymbol{z}_0 \sim q_0$. The results of Section 3.1 suggest that, in the limit of infinite data, this direction is proportional to $\nabla_{\boldsymbol{z}_t} p(\boldsymbol{z}_t; \sigma) - \nabla_{\boldsymbol{z}_t} q_t(\boldsymbol{z}_t; \sigma)$.

For the Gaussian kernel, we have

$$\nabla_{\boldsymbol{z}_t} K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}) = K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}) \left(\frac{\boldsymbol{x} - \boldsymbol{z}_t}{\sigma^2} \right),$$

and for discrete time and finite data we can write equation 41 as

$$\Delta \boldsymbol{z}_t = \frac{1}{N} \sum_{i=1}^{N} \left[K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}_i) \left(\frac{\boldsymbol{x}_i - \boldsymbol{z}_t}{\sigma^2} \right) \right] - \frac{1}{M} \sum_{j=1}^{M} \left[K_\sigma(\boldsymbol{z}_t, \boldsymbol{y}_j) \left(\frac{\boldsymbol{y}_j - \boldsymbol{z}_t}{\sigma^2} \right) \right] \tag{42}$$

$$= \sum_{i=1}^{N} w_i^{(p)} \boldsymbol{x}_i - \sum_{j=1}^{M} w_j^{(q_t)} \boldsymbol{y}_j + \left(\sum_{j=1}^{M} w_j^{(q_t)} - \sum_{i=1}^{N} w_i^{(p)} \right) \boldsymbol{z}_t, \tag{43}$$

where $w_i^{(p)} = K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}_i)/(N\sigma^2)$ and $w_j^{(q_t)} = K_\sigma(\boldsymbol{z}_t, \boldsymbol{y}_j)/(M\sigma^2)$. This process defines the *MMD gradient flow* (Arbel et al., 2019). The kernel version of SD flow (equation 13) can also be written in the form of equation 43 by setting $w_i^{(p)} = \frac{1}{2} K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}_i) / \sum_{i=1}^{N} K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}_i)$ and $w_j^{(q_t)} = \frac{1}{2} K_\sigma(\boldsymbol{z}_t, \boldsymbol{x}_i) / \sum_{j=1}^{M} K_\sigma(\boldsymbol{z}_t, \boldsymbol{y}_j)$, which causes the $\boldsymbol{z}_t$ term to vanish.

There are practical consequences of this difference in weighting schemes between methods, which put the MMD gradient flow at a disadvantage in some conditions. We discuss this issue further in Appendix E.

E Experiments

Here we provide several experimental results using toy data. We intentionally avoid experiments on high-dimensional image data and instead focus on experiments that can easily be run on one's CPU.[10] Beyond the considerable computational and data demands, there are many architectural design choices and training tricks that contribute to the current state of the art in image generation, which fall outside the scope of this work.

E.1 Data Optimization

We tested our algorithm for SD flow versus our own implementation of MMD gradient flow (Arbel et al., 2019) based on that paper's pseudocode description. We chose to use our own implementation to guarantee that all settings and hyperparameters between the methods were identical. We tested both batch-based and full-data versions of both algorithms. We performed two versions of the experiment that varied where the base data distribution was initialized, which, based on our analysis, we believed would reveal performance differences between the two methods.

Remark 1 *If p and q_t are far apart, for a point $\boldsymbol{z}_t = \boldsymbol{y} + \sigma\boldsymbol{\epsilon}$, with $\boldsymbol{y} \sim q_t$ and a small kernel bandwidth σ, equation 43 (Appendix D) shows that under the MMD framework $\sum_j w_j^{(q_t)} \approx 1/(M\sigma^2)$ and $\sum_i w_i^{(p)} \approx 0$. This suggests that under these conditions, the MMD gradient flow direction $\Delta \boldsymbol{z}_t^{MMD}$ will be nearly parallel to $\boldsymbol{z}_t - \boldsymbol{y} = \sigma\boldsymbol{\epsilon}$, which would have the effect of increasing the variance of q_t while not necessarily pushing it toward p. (See also Remark 2 below.) Normalizing the weights in equation 43 to sum to one resolves this issue, however, and MMD gradient flow and SD flow then become equivalent.*

To measure distribution alignment, we define a mean *characteristic function distance* (CFD),

$$\mathbb{D}_{\mathrm{CF}}(p \| q_t) = \frac{1}{K} \sum_{k=1}^{K} \left| \mathbb{E}_{\boldsymbol{x} \sim p} \left[\exp(\mathrm{i} \boldsymbol{\omega}_k^\top \boldsymbol{x}) \right] - \mathbb{E}_{\boldsymbol{y} \sim q_t} \left[\exp(\mathrm{i} \boldsymbol{\omega}_k^\top \boldsymbol{y}) \right] \right|,$$

the mean absolute difference between the empirical characteristic functions of p and q_t evaluated at K frequencies ($K = 256$ in our case) drawn from a normal distribution.

---

[10] All experiments reported here were carried out in MATLAB (R2022b) on a MacBook Pro (2019).

E.1.1 Overlapping Base and Target Data

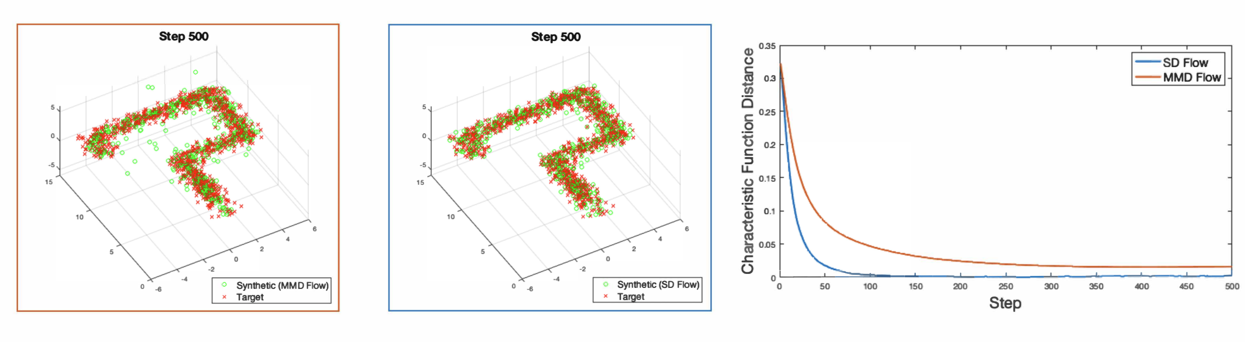

Figure 1: Relative performance on fitting the "mystery distribution," a mixture of 30 Gaussians in $\mathbb{R}^3$. The 500th step of MMD gradient flow (Arbel et al., 2019) is on the left, while the 500th step of SD flow (ours) is in the center. The right panel shows the algorithms' performance in terms of the mean characteristic function distance (CFD).

In this experiment, our target "mystery distribution" p is a mixture of 30 Gaussians in $\mathbb{R}^3$ arranged in the shape of a question mark. We used a modified cosine noise schedule to interpolate between a maximum and minimum variance $\gamma(0) = \sigma^2_{\text{max}} = 4$ and $\gamma(t_{\text{max}}) = \sigma^2_{\text{min}} = 0.5$ over $T = 500$ steps according to $\gamma(t) = \sigma^2_{\text{max}} \cos(\pi t/2)$ for $t \in [0, t_{\text{max}}]$,[11] with $t_{\text{max}} = 2/\pi \cos^{-1}(\sigma^2_{\text{min}}/\sigma^2_{\text{max}})$. Many noise schedules are possible, even *constant* noise schedules (see Appendix E.4), but the cosine schedule is a popular choice in diffusion modeling (Nichol & Dhariwal, 2021).

Both algorithms were initialized with the same base data sample of 500 points and were fit to the same sample of 1024 points from the target distribution. The step size for each algorithm was set automatically so that the first perturbation to the base data would have a norm of 0.5. These experiments correspond to the SD flow procedure described in Algorithm 1.

We set the base data to overlap the target distribution by initializing it as a spherical, unit-variance Gaussian centered at the target distribution's mean. With this initialization, both flows successfully yielded samples plausibly drawn from the target distribution, although the MMD flow produced more outlying points and had a consistently higher CFD value. (See Figure 1.)

E.1.2 Non-overlapping Base and Target Data

In Remark 1, our analysis suggested that the default weighting of terms in the MMD gradient flow could lead to pathological cases in which synthetic points from the base distribution would be perturbed in such a way to make the variance increase without aligning the synthetic and target distributions. In particular, we argued that this could occur if the kernel bandwidth in the MMD flow (equivalent to the noise level in SD flow) was small relative to the separation of the two distributions' data points.

We tested this scenario by initializing the base distribution at some distance from the target distribution. The target distribution and all other settings from the experiment described in Section E.1.1 were maintained, although this time we also tested the batch-based version of both algorithms with a batch size of 128. The results of these experiments are shown in Figure 2.

MMD gradient flow maintains the spherical shape of the base distribution for a large subset of points while increasing their variance, exactly in the manner predicted in Remark 1. SD flow, on the other hand, still successfully fits the target distribution. Note also that SD flow's performance is consistent when using a batch-based approach relative to using the full data in every step, while MMD gradient flow plateaus at a higher CFD in the batch-based setting.

---

[11] This was set as T linearly spaced points between 0 and t_{max} using MATLAB's `linspace` function.

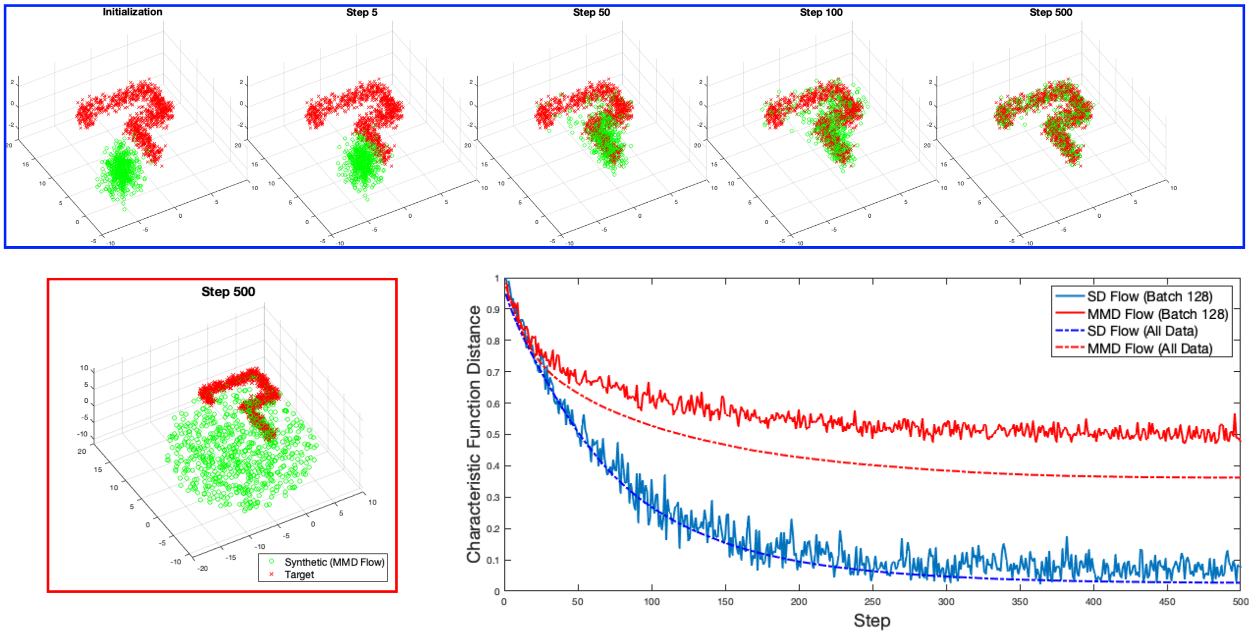

Figure 2: Top: Evolution of synthetic data points from an offset base distribution toward the target distribution over 500 steps of SD flow. Bottom left: Final step of MMD gradient flow from the same base distribution initialization. Bottom right: Characteristic function distance (CFD) for both flows under batch (solid lines) or full-data (broken lines) conditions.

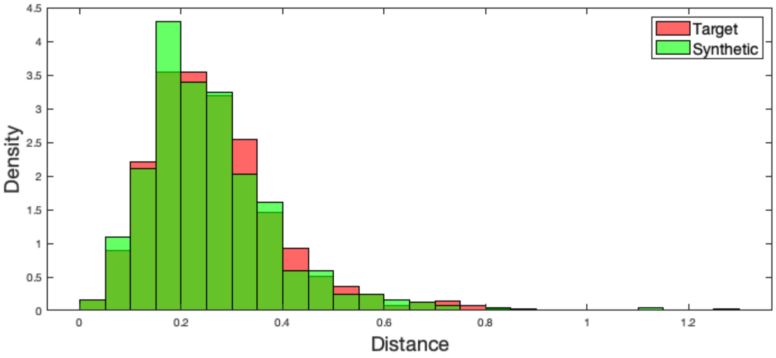

Figure 3: Distribution of distances from synthetic (green) and target (red) data points to their first nearest neighbors in the target distribution.

It is important to note that SD flow does not cause the synthetic data to collapse to nearest neighbors in the target distribution. In Figure 3, we show the distribution of distances from points in the synthetic distribution to their first nearest neighbors in the target distribution (shaded in green). Note the overlap with the distribution of distances between target data points and their first nearest neighbors (excluding themselves) in the target distribution. Overfitting to the target data would result in a large concentration of mass near zero for the synthetic data.

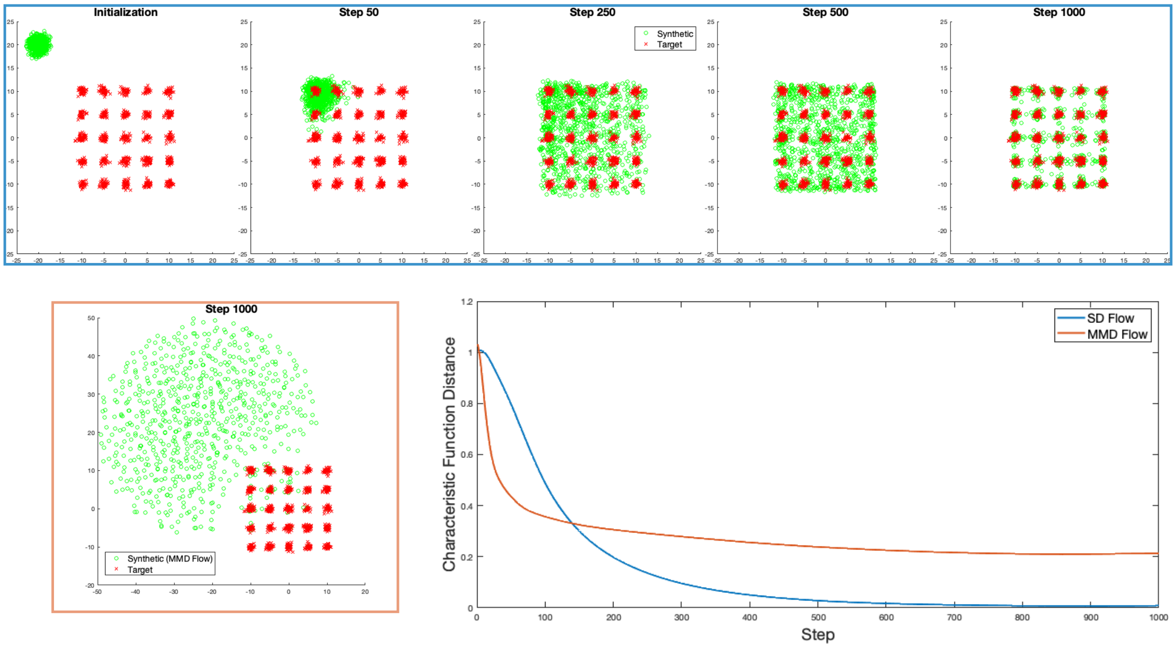

Figure 4: Top: Evolution of synthetic data points from an offset base distribution toward the target distribution of 25 Gaussians over 1000 steps of SD flow. Bottom left: Final step of MMD gradient flow from the same base distribution initialization. Bottom right: Characteristic function distance (CFD) for both flows.

E.2 Fitting a 25-Gaussian Grid

We then investigated whether SD flow would get stuck in local modes when initialized near them. To test this, we created a target distribution of 1024 points drawn from a mixture of 25 spherical Gaussians arranged on a grid in $\mathbb{R}^2$ and initialized 1024 points from a spherical Gaussian base distribution at a large distance from the target distribution but closest to its top-left component. (See Figure 4.)

We once again compared performance of SD flow relative to MMD gradient flow. Both algorithms were run for 1000 steps using a cosine noise schedule (described above) with $\sigma^2_{\text{max}} = 10$ and $\sigma^2_{\text{min}} = 0.5$. We tested only the full-data (as opposed to batch-based) setting in this experiment.

While MMD gradient flow showed an initially steeper reduction in the CFD relative to SD flow, it quickly plateaued while once again demonstrating an explosion of variance within the subset of points that did not converge to the target distribution. SD flow, on the other hand, continued to reduce the CFD while fitting all components of the target. Once again, the analysis of nearest neighbors showed no overfitting to the target data, with nearest-neighbor distances distributed in a manner consistent with that of the target distribution.

Remark 2 *Our method prescribes that we inject noise at a level equal to the kernel bandwidth in order to sample from the noise-smoothed proxy distribution. In contrast, with MMD gradient flow the noise level is a separate parameter that essentially controls a regularization effect. In that case, this added noise is typically at a level far greater than that of the kernel bandwidth, which remains fixed during training. We further note that evidence of the variance-exploding effect that we predicted in Remark 1 and observe experimentally is also apparent in the authors' original paper (e.g. Appendix G.2 therein).*

E.3 Data-Set Interpolation

Standard score-based generative modeling is designed such that the end of the forward process is a Gaussian distribution. While this has the advantage of defining a prior that is easy to sample from for the reverse,

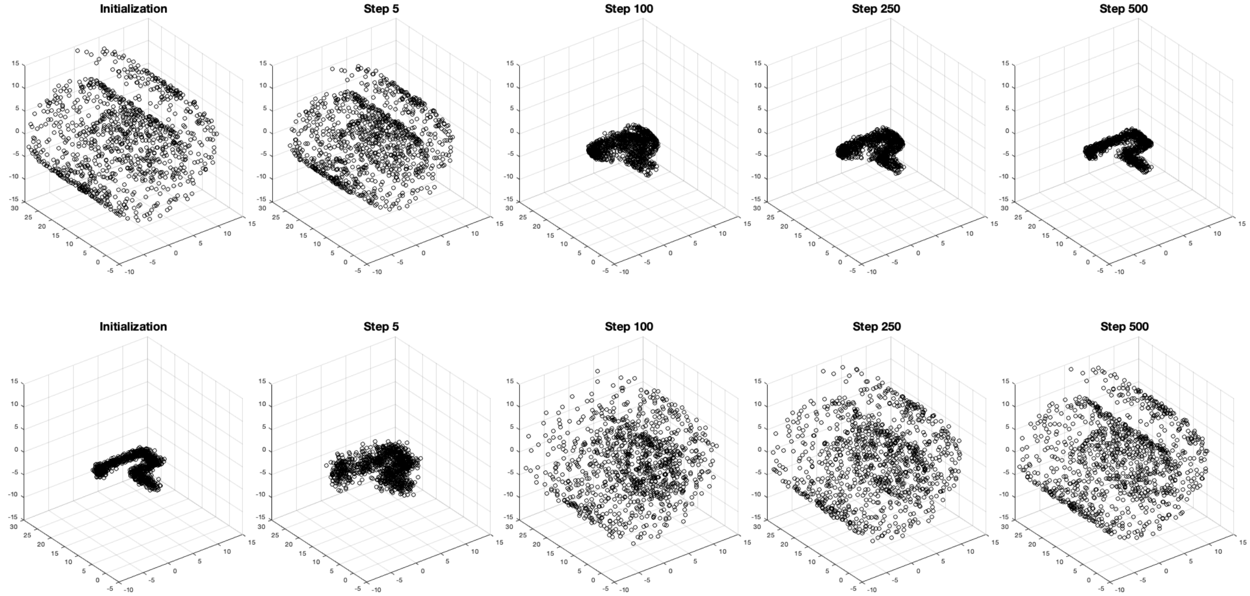

Figure 5: Top: Data-set interpolation via evolution of 1024 points from the "Swiss roll" distribution to the "mystery" distribution in $\mathbb{R}^3$. Bottom: The reverse interpolation, from the "mystery" distribution to the "Swiss roll" distribution.

generative process, it limits the flexibility of the method. Recent work on approximating a *Schrödinger bridge* between source and target distributions (De Bortoli et al., 2021) relaxes this limitation, but the method itself is relatively complicated.

SD flow, on the other hand, is (in our opinion) a simpler and more intuitive method that also has no restriction on the distributions p and q. It is therefore also capable of performing interpolation between arbitrary data sets. The results of one such interpolation experiment are shown in Figure 5. The figure actually shows *two* interpolation experiments: The first evolves 1024 points of the "Swiss roll" data toward the "mystery" distribution (Section E.1.1) in $\mathbb{R}^3$, while the second evolves from the "mystery" distribution to the "Swiss roll." The same variance schedule as in Section E.1.1 was employed. In each case, the entire process took roughly 8.8 seconds on the CPU of a 2019 MacBook Pro.

E.4 Model Optimization

Although we do not run any experiments on high-dimensional image data for the reasons described above, we report here an experiment using the model-optimization application (Algorithm 2) on "high"-dimensional data in $\mathbb{R}^{50}$. Here the scare quotes acknowledge that this dimensionality is far lower than the thousands to millions of dimensions typical in high-resolution image data, but it is high enough to exhibit the problematic, intuition-challenging characteristics of high-dimensional data in general.

Specifically, it is well known that as data dimensionality grows, the ratio of the distance of a point to its *farthest* neighbor, $D_{\max}$, and the distance to its *nearest* neighbor, $D_{\min}$, tends toward unity. The ratio $D_{\max}/D_{\min}$ drops precipitously in lower dimensions before leveling off at around 30 dimensions and very slowly approaching an asymptote of one afterward (Beyer et al., 1999). We therefore chose $\mathbb{R}^{50}$ as a reasonable setting to challenge our approach in high dimensions.

We generated a ground-truth target distribution by randomly populating a 50×25 matrix $\boldsymbol{B}$ with values drawn from $\mathcal{N}(0, 0.25\boldsymbol{I})$ and a 50-vector $\boldsymbol{\mu}$ with values drawn from $\mathcal{N}(10, \boldsymbol{I})$. Target data samples from

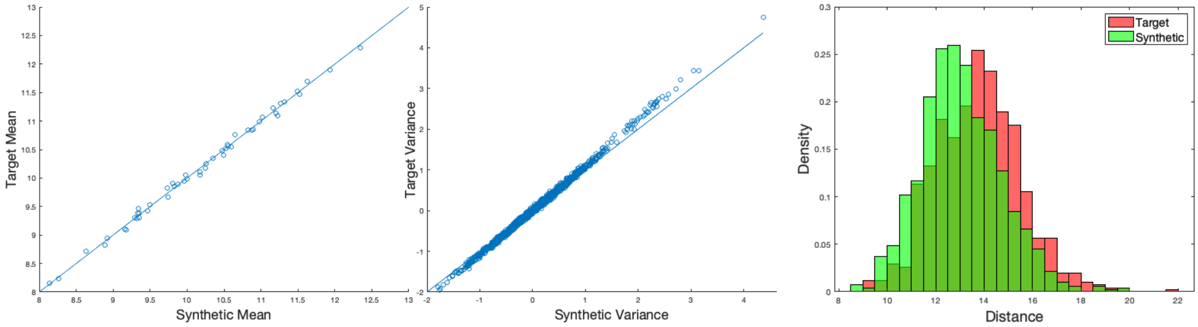

Figure 6: Model optimization results in $\mathbb{R}^{50}$ using a constant noise schedule. SD flow allows a parametric model to be learned that very closely matches the target mean ($\boldsymbol{\mu}$ versus $\hat{\boldsymbol{\mu}}$, left panel) and the elements of the covariance matrix ($\boldsymbol{B}\boldsymbol{B}^\top$ vs $\hat{\boldsymbol{B}}\hat{\boldsymbol{B}}^\top$, center panel). Diagonals are included for reference. Nearest-neighbor analysis showed no overfitting of the data (right panel) but showed a slightly lower average distance to nearest neighbors in the target set than exhibited by the target data relative to itself.

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{B}\boldsymbol{B}^\top)$ were then generated[12] by drawing samples $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \in \mathbb{R}^{25}$ and forming $\boldsymbol{x} = \boldsymbol{B}\xi + \boldsymbol{\mu}$. The model parameters to be learned were a 50×25 matrix $\hat{\boldsymbol{B}}$, initialized from $\mathcal{N}(0, 0.01\boldsymbol{I})$, and a 50-vector $\hat{\boldsymbol{\mu}}$, initialized to all zeros. This model can be interpreted as a single-layer linear neural network, but it is most important to note that it exactly matches the capacity of the data-generating model.

If we retained the input vectors $\boldsymbol{\xi} \in \boldsymbol{\Xi}$ for the outputs $\boldsymbol{x} \in \boldsymbol{X}$, then the task of learning the parameters would be fairly straightforward in the context of a regression problem on paired data $\{(\boldsymbol{\xi}, \boldsymbol{x})\}$. But in the general IGM problem, we have only *unpaired* data to work with, so we assume that all information about the target data inputs is unavailable.

We performed 1000 steps of SD flow using Algorithm 2 with a *constant* noise schedule of 10 times the average distance of the initial synthetic (base) distribution to first nearest neighbors in the target distribution (corresponding to $\sigma^2 > 700$),[13] with a batch size of 1024, an SD flow step size of $\eta = 1$, and a regression learning rate of $\lambda = 10^{-3}$. The training on the CPU of a 2019 MacBook Pro was completed in 24.72 seconds. Other than brief experimentation to set reasonable values, no effort was made to optimize these hyperparameters.

The results of this experiment are shown in Figure 6. Despite (or perhaps *because of*) a massive and constant injection of noise, SD flow successfully fit the target distribution. Analysis of nearest neighbors once again showed that SD flow did not overfit to the target distribution, although there was a very slight shift toward lower distances between synthetic data and their nearest neighbors in the target distribution as compared with the target data's nearest-neighbor distances relative to itself.

---

[12] Technically, this is not completely well defined as a normal distribution, since $\boldsymbol{B}\boldsymbol{B}^\top$ is not of full rank.

[13] We found that using a higher amount of noise somewhat improved the convergence profile of the algorithm.