DIFFUSION MAP PARTICLE SYSTEMS FOR GENERATIVE MODELING

Fengyi Li[✉]* and Youssef Marzouk[✉]

Massachusetts Institute of Technology, Cambridge, MA 02139 USA

(Communicated by Kody J. H. Law)

Abstract. We propose a novel diffusion map particle system (DMPS) for generative modeling, based on diffusion maps and Laplacian-adjusted Wasserstein gradient descent (LAWGD). Diffusion maps are used to approximate the generator of the corresponding Langevin diffusion process from samples, and hence to learn the underlying data-generating manifold. On the other hand, LAWGD enables efficient sampling from the target distribution given a suitable choice of kernel, which we construct here via a spectral approximation of the generator, computed with diffusion maps. Our method requires no offline training and minimal tuning, and can outperform other approaches on data sets of moderate dimension.

1. **Introduction.** Generative modeling is a central task in fields such as computer vision [7, 21, 43, 48, 59, 41] and natural language processing [61, 34, 30, 29], and applications ranging from medical image analysis [60, 37, 24, 33] to protein design [3, 57]. Given samples from a probability distribution of interest, the goal of generative modeling is to generate additional samples from the same distribution, without knowledge of its unnormalized density. Despite their successes, popular generative models such as variational auto-encoders (VAE) [26, 42], generative adversarial networks (GAN) [16], and diffusion models or score-based generative models (SGM) [20, 50, 52, 58] typically need careful hyperparameter tuning [45, 51] and may involve long convergence times, e.g., for Langevin-type sampling [14]. The performance of such methods highly depends on the architecture and the choice of parameters of deep neural networks [46, 25], which, all too often, require expert knowledge.

In this paper, we propose a new nonparametric kernel-based approach to generative modeling, based on diffusion maps and interacting particle systems.

Diffusion maps [11, 10, 36, 35], along with many other graph-based methods [4, 53, 44], have mainly been used as a tool for nonlinear dimension reduction. The kernel matrix, constructed using pairwise distances between samples with proper normalization, approximates the generator of a Langevin diffusion process. This approximation becomes exact as the number of samples goes to infinity and the kernel bandwidth goes to zero. Construction of the kernel matrix using smooth kernels (e.g., Gaussians) enables one to compute the inverse of the eigenvalues and the gradients of the eigenfunctions analytically.

Separately, the notion of gradient flow [47, 2, 13] underlies a very active field of research and offers a unifying perspective on sampling and optimization [22, 56], with numerous connections to partial differential equations and differential geometry [55, 1]. Many common sampling algorithms approximate the gradient flow (of some functional) on a space of probability measures. The unadjusted Langevin algorithm (ULA) is a canonical example, and follows from the time discretization of a Langevin SDE. But, many other particle systems, particularly interacting particle systems, approximate gradient flows: examples include Stein variational gradient descent (SVGD) [31, 32], affine-invariant interacting Langevin dynamics (ALDI) [15], and Laplacian-adjusted Wasserstein gradient descent (LAWGD) [9]. ULA, SVGD, and other algorithms access the target distribution via the gradient of its log-density, and hence many of these methods have become popular for Bayesian inference.

Our approach combines these two ideas by using diffusion maps to directly approximate (the gradient of the inverse of) the generator of the Langevin diffusion process from samples, and in turn using this approximation within LAWGD to produce more samples efficiently. The only input is a set of samples from the target distribution, and thus our approach enables generative modeling. Compared to other generative modeling methods, our approach has several advantages. First, the use of diffusion maps facilitates accurate sampling from distributions supported on manifolds, particularly when the dimension of the manifold is lower than that of the ambient space. Second, we demonstrate accurate sampling from distributions with (a priori unknown) bounded support. Both of these features are in contrast with methods driven only by approximations of the local score: we conjecture that such methods are less able to detect the overall geometry of the target distribution, whereas our approach harnesses graph-based methods that are widely used for non-linear dimension reduction to approximate the generator as a whole. Finally, our method is quite simple and computationally efficient comparing to training a neural network (e.g., as in score-based generative modeling) [50] or normalizing flows [8, 19]: the only parameter that needs to be tuned is the kernel bandwidth, and no offline training is required.

2. **Notation and preliminaries.** We use π to denote the target distribution, i.e., the distribution we would like to sample from, and assume it is supported on a compact manifold $\mathcal{D}$ embedded in $\mathbb{R}^d$. We let $V : \mathcal{D} \to \mathbb{R}$ denote the potential associated with this distribution, such that $\pi \propto \exp(-V)$. All distributions are assumed to have densities with respect to Lebesgue measure on $\mathbb{R}^d$, and we abuse notation by using the same symbol to denote a measure and its density. The kernel $K(\cdot, \cdot) : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ is assumed to be differentiable with respect to both arguments, and we use $\nabla_1 K(\cdot, \cdot)$, $\nabla_2 K(\cdot, \cdot)$ to denote the (Euclidean) gradient of the kernel with respect to its first and second arguments, respectively. We assume sufficient regularity to exchange the order of integration and differentiation (Leibniz integral rule) throughout.

3. **A review of Wasserstein gradient flow and LAWGD.** We first review the basics of Wasserstein gradient flow and recall the LAWGD algorithm of [9].

3.1. **Gradient flow on Wasserstein space.** Let $F(\mu)$ be a functional over the space of probability measures, i.e., $F : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$, where $\mathcal{P}_2$ is the space of probability measures that have finite second moment. We seek to steer the measure μ_t (at time t) in the direction of steepest descent, defined by F and a chosen metric.

That is, $\frac{\partial \mu_t}{\partial t} = -\nabla_{W_2} F(\mu_t)$, where ∇_{W_2} denotes the general gradient in Wasserstein metric. Under some smoothness assumptions, we can write this as

$$\frac{\partial \mu_t}{\partial t} = \text{div}(\mu_t \nabla \delta F(\mu_t)), \tag{1}$$

where $\delta F(\mu)$ is the first variation of F evaluated at μ [54]. If we choose the functional F to be the Kullback–Leibler (KL) divergence, $F(\mu) = \mathrm{D}_{\mathrm{KL}}(\mu || \pi)$, then (1) becomes

$$\frac{\partial \mu_t}{\partial t} = \text{div}\left(\mu_t \nabla \log \frac{d\mu_t}{d\pi}\right),$$

which is the Fokker–Planck equation [22]. The measure μ_t can be approximated by particles evolving according to the following dynamic:

$$\dot{x} = -\nabla \log \frac{d\mu_t}{d\pi}(x).$$

Forward Euler discretization with stepsize h then yields the following numerical scheme:

$$x_{t+1} = x_t - h\nabla \log \frac{d\mu_t}{d\pi}(x_t).$$

3.2. **LAWGD algorithm.** One challenge with the scheme above is that the measure μ_t is intractable at time t. To solve this problem, SVGD [32] implements the following kernelized dynamics (in the continuum limit):

$$\dot{x} = -\int K(x, y) \nabla \log \frac{d\mu_t}{d\pi}(y) d\mu_t(y).$$

The expression above can be equivalently written as

$$\dot{x} = -\int K(x, y) \nabla \frac{d\mu_t}{d\pi}(y) d\pi(y). \tag{2}$$

Define

$$\mathcal{K}_\pi f(x) := \int K(x, y) f(y) d\pi(y).$$

Then, we write (2) as

$$\dot{x} = -\mathcal{K}_\pi \nabla \frac{d\mu_t}{d\pi}(x),$$

and, under SVGD, the density evolves according to

$$\partial_t \mu_t = \text{div}\left(\mu_t \mathcal{K}_\pi \nabla \frac{d\mu_t}{d\pi}\right).$$

On the other hand, LAWGD makes the JKO scheme implementable by considering the kernelization

$$\dot{x} = -\nabla \mathcal{K}_\pi \frac{d\mu_t}{d\pi}(x),$$

and expanding it to obtain

$$\dot{x} = -\int \nabla_1 K(x, y) \frac{d\mu_t}{d\pi}(y) d\pi(y) = -\int \nabla_1 K(x, y) d\mu_t(y). \tag{3}$$

The kernel $K(\cdot, \cdot)$ is specifically chosen in such a way that $\mathcal{K}_\pi = \mathscr{L}^{-1}$, where $-\mathscr{L} = \nabla^2 - \langle \nabla V, \nabla \cdot \rangle$ is generator of the Langevin diffusion process, and ∇^2 denotes

the Laplacian operator. Here, we assume $\mathscr{L}$ has discrete spectrum (see Appendix A). This choice is motivated by the rate of change of KL divergence,

$$\partial_t \, \mathrm{D_{KL}}(\mu_t || \pi) = -\mathbb{E}_\pi \left[\frac{d\mu_t}{d\pi} \mathscr{L} \mathcal{K}_\pi \frac{d\mu_t}{d\pi} \right]. \tag{4}$$

Indeed, such a choice yields

$$\partial_t \, \mathrm{D_{KL}}(\mu_t || \pi) = -\mathbb{E}_\pi \left[\left(\frac{d\mu_t}{d\pi} - 1 \right)^2 \right] = -\chi^2(\mu_t || \pi). \tag{5}$$

The evolution of the density under LAWGD thus follows

$$\partial_t \mu_t = \mathrm{div} \left(\mu_t \nabla \mathscr{L}^{-1} \frac{d\mu_t}{d\pi} \right).$$

Here, $\mathscr{L}^{-1}$ is the pseudo-inverse of the generator of the Langevin diffusion process. In practice, we write $\mathscr{L}$ using its spectral decomposition and define $\mathscr{L}^{-1}$ using all of the eigenpairs of $\mathscr{L}$ except for the one corresponding to the zero eigenvalue. See Appendix A for more details. Now suppose we initialize $\{x_0^i\}_{i=1}^M \sim \mu_0$. We then obtain a discrete algorithm from (3), where the update step reads

$$x_{t+1}^i = x_t^i - \frac{h}{M} \sum_{j=1}^M \nabla_1 K_{\mathscr{L}^{-1}}(x_t^i, x_t^j). \tag{6}$$

Here, $K_{\mathscr{L}^{-1}}$ can be understood as a kernelized version of $\mathscr{L}^{-1}$, satisfying $\mathscr{L}^{-1} f(x) = \int K_{\mathscr{L}^{-1}}(x,y) f(y) d\pi(y)$. In particular, setting $f = \frac{d\mu_t}{d\pi}$, we have $\mathscr{L}^{-1} \frac{d\mu_t}{d\pi}(x) = \int K_{\mathscr{L}^{-1}}(x,y) \frac{d\mu_t}{d\pi}(y) d\pi(y) = \int K_{\mathscr{L}^{-1}}(x,y) d\mu_t(y)$. More details about LAWGD can be found in [9].

4. Diffusion map and kernel construction.

4.1. Kernel approximation of $\mathscr{L}$.
It remains to see how to implement (6) in the generative modeling setting. Given a finite collection of samples $\{z^i\}_{i=1}^n \sim \pi$, our goal is to approximate $\nabla \mathscr{L}^{-1} f(x)$. Diffusion maps [11, 10, 36, 35] provide a natural framework for approximating $\mathscr{L}$ using kernels. Consider the Gaussian kernel $K_\epsilon(x,y) = e^{-\frac{\|x-y\|^2}{2\epsilon}}$ under some normalization

$$M_\epsilon(x,y) := \frac{K_\epsilon(x,y)}{\sqrt{\int K_\epsilon(x,y) d\pi(x)} \sqrt{\int K_\epsilon(x,y) d\pi(y)}}.$$

We construct the two kernels

$$P_\epsilon^f(x,y) := \frac{M_\epsilon(x,y)}{\int M_\epsilon(x,y) d\pi(x)},$$

$$P_\epsilon^b(x,y) := \frac{M_\epsilon(x,y)}{\int M_\epsilon(x,y) d\pi(y)},$$

by normalizing with respect to the first or second argument. Here, f and b stand for 'forward' and 'backward', respectively. Their actions on a function g are defined as

$$T_\epsilon^{f,b} g(\cdot) = \int P_\epsilon^{f,b}(\cdot, y) g(y) d\pi(y).$$

We also define the associated operators

$$L_\epsilon^{f,b} := \frac{\mathrm{Id} - T_\epsilon^{f,b}}{\epsilon}.$$

As studied in [35], both the forward and backward operators converge to the generator in an appropriate sense,

$$\lim_{\epsilon \to 0} L_\epsilon^f = \lim_{\epsilon \to 0} L_\epsilon^b = \mathscr{L}.$$

Combining the previous results, we have

$$\lim_{\epsilon \to 0} \frac{g(\cdot) - \int P_\epsilon^{f,b}(\cdot, y)g(y)d\pi(y)}{\epsilon} = \mathscr{L}g(\cdot). \tag{7}$$

Note that this approximation holds only when data lie on a compact manifold [35, 18, 49]. In practice, however, this assumption can be relaxed. In addition, although $L_\epsilon^{f,b}$ converges to a symmetric operator in the continuum limit, neither L_ϵ^f nor L_ϵ^b are symmetric. However, they satisfy $P_\epsilon^f(x,y) = P_\epsilon^b(y,x)$ and $L_\epsilon^f = (L_\epsilon^b)^*$. Therefore, one way to get a symmetric operator is to take the average of the two:

$$L_\epsilon = \frac{1}{2}(L_\epsilon^f + L_\epsilon^b).$$

L_ϵ inherits all the properties of the forward and the backward kernel, hence converging to $\mathscr{L}$ in the limit. We now consider a finite sample approximation of the operator $\mathscr{L}$. Given samples $\{z^i\}_{i=1}^N \sim \pi$, the above construction can be approximated by samples, namely, by replacing the integral with its empirical average (see Appendix B). We add another subscript N to these kernels, i.e., $M_{\epsilon,N}, P_{\epsilon,N}^{f,b}, T_{\epsilon,N}^{f,b}$, and $L_{\epsilon,N}^{f,b}$, to denote their counterparts resulting from finite sample approximation, where

$$M_{\epsilon,N}(x,y) = \frac{K_\epsilon(x,y)}{\sqrt{\sum_{i=1}^N K_\epsilon(z^i,y)}\sqrt{\sum_{i=1}^N K_\epsilon(x,z^i)}},$$

$$P_{\epsilon,N}^f(x,y) = \frac{M_{\epsilon,N}(x,y)}{\sum_{i=1}^N M_{\epsilon,N}(z^i,y)},$$

$$P_{\epsilon,N}^b(x,y) = \frac{M_{\epsilon,N}(x,y)}{\sum_{i=1}^N M_{\epsilon,N}(x,z^i)},$$

and

$$P_{\epsilon,N}(x,y) = \frac{1}{2}\left(P_{\epsilon,N}^f(x,y) + P_{\epsilon,N}^b(x,y)\right). \tag{8}$$

4.2. **Spectral approximation of $\nabla \mathscr{L}^{-1}f(x)$.** We now exploit the spectral properties of the kernel. Recall that the operator $\mathscr{L}$ admits the kernel approximation

$$\mathscr{L} = \lim_{\epsilon \to 0} \frac{\mathrm{Id} - T_\epsilon}{\epsilon},$$

where $T_\epsilon g(\cdot) = \int P_\epsilon(\cdot, y)g(y)\pi(y)$. Note that $P_\epsilon = \frac{1}{2}(P_\epsilon^f + P_\epsilon^b)$ is a symmetric positive definite kernel (see Appendix C for a proof). The classic tool for studying such a kernel is Mercer's theorem.

Theorem 4.1 (Mercer). *Let T_ϵ and $P_\epsilon(x,y)$ be defined as in the previous discussion. Then there is a sequence of non-negative eigenvalues $\{\lambda_i\}_{i\in\mathbb{N}}$ and an orthonormal basis of eigenfunctions $\{\phi_i\}_{i\in\mathbb{N}}$ of T_ϵ, i.e.,*

$$\int P_\epsilon(x,y)\phi_m(y)d\pi(y) = \lambda_m\phi_m(x),$$

$$\int \phi_m(x)\phi_n(x)d\pi(y) = \delta_{m,n},$$

such that

$$P_\epsilon(x,y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(y). \tag{9}$$

An immediate corollary we can see is the connection between the spectra of T_ϵ and L_ϵ.

Corollary 4.2. *Let T_ϵ and L_ϵ be defined as in the previous discussion, and denote the eigenvalues and eigenfunctions of T_ϵ by $\{\lambda_i, \phi_i\}$, with $1 = \lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \cdots$. Then the set of eigenvalues and eigenfunctions of L_ϵ is $\left\{\frac{1-\lambda_i}{\epsilon}, \phi_i\right\}$. In particular $\mathscr{L}$ can be written as the limit of the integral operator,*

$$\mathscr{L}f(x) = \lim_{\epsilon \to 0} \int K_{\mathscr{L},\epsilon}(x,y)f(y)d\pi(y),$$

where $K_{\mathscr{L},\epsilon}(x,y) = \sum_{i=1}^{\infty} \frac{1-\lambda_i}{\epsilon}\phi_i(x)\phi_i(y)$.

The interchange of the order between limit and the integral is guaranteed by the dominated convergence theorem. Using Mercer's theorem and its corollary, we can write $K_{\mathscr{L},\epsilon}$ and its inverse using the eigendecomposition:

$$K_{\mathscr{L},\epsilon}(x,y) = \sum_{i=1}^{\infty} \frac{1-\lambda_i}{\epsilon}\phi_i(x)\phi_i(y),$$

$$K_{\mathscr{L}^{-1},\epsilon}(x,y) = \sum_{i=1}^{\infty} \frac{\epsilon}{1-\lambda_i}\phi_i(x)\phi_i(y).$$

$\mathscr{L}^{-1}$ admits the kernel expression

$$\mathscr{L}^{-1}f(x) = \lim_{\epsilon \to 0} \int K_{\mathscr{L}^{-1},\epsilon}(x,y)f(y)d\pi(y).$$

For the case where $\nabla_1 K_{\mathscr{L},\epsilon}(x,y)$ exists, we have

$$\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x,y) = \sum_{i=1}^{\infty} \frac{\epsilon}{1-\lambda_i}\nabla\phi_i(x)\phi_i(y).$$

Therefore,

$$\nabla L_\epsilon^{-1}f(x) = \int \nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x,y)f(y)d\pi(y).$$

Under regularity assumptions (see Appendix D), $\lim_{\epsilon \to 0} \nabla L_\epsilon f(x) = \nabla\mathscr{L}f(x)$ and $\lim_{\epsilon \to 0} \nabla L_\epsilon^{-1}f(x) = \nabla\mathscr{L}^{-1}f(x)$.

5. The generative model.

5.1. Computing $\nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(x,y)$ for arbitrary points x, y.
We can now write the update step (6) using an ϵ kernel approximation

$$x_{t+1}^i = x_t^i - \frac{h}{M}\sum_{j=1}^{M} \nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x_t^i, x_t^j),$$

or using its empirical counterparts,

$$x_{t+1}^i = x_t^i - \frac{h}{M} \sum_{j=1}^{M} \nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(x_t^i, x_t^j).$$

Note that the kernel $K_{\mathscr{L}^{-1},\epsilon,N}$ is constructed only at the locations of the training samples $\{z^i\}_{i=1}^N \sim \pi$. In order to obtain an implementable algorithm, we need to be able to compute $\nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(\cdot,\cdot)$ for arbitrary points x^*, y^*. One way is to interpolate the eigenfunctions ϕ and their gradients $\nabla\phi$ at x^* and y^*. However, this is restricted by the number of training samples for learning the kernel, as well as the interpolation method. To overcome this problem, we propose yet another natural way of computing $\nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(\cdot,\cdot)$ by taking advantage of the eigendecomposition of the kernel, avoiding interpolation of eigenfunctions. We illustrate this idea using the ϵ kernel approximation $\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(\cdot,\cdot)$; its empirical counterpart follows directly. Set $\sigma_i = \frac{1-\lambda_i}{\epsilon}$, and recall from (9) that $P_\epsilon(x,y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(y)$. Consider the following eigendecomposition of the kernel:

$$\nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x^*, y^*)$$

$$= \int_{\mathcal{Z}} \int_{\mathcal{W}} \left(\sum_{k=1}^{\infty} \lambda_k \nabla\phi_k(x^*)\phi_k(w) \right)$$

$$\left(\sum_{j=1}^{\infty} \lambda_j^{-1}\sigma_j^{-1}\lambda_j^{-1}\phi_j(w)\phi_j(z) \right) \left(\sum_{i=1}^{\infty} \lambda_i \phi_i(z)\phi_i(y^*) \right) d\pi(w)\, d\pi(z)$$

$$= \int_{\mathcal{Z}} \int_{\mathcal{W}} \nabla_1 P_\epsilon(x^*, w) \left(\sum_{j=1}^{\infty} \lambda_j^{-1}\sigma_j^{-1}\lambda_j^{-1}\phi_j(w)\phi_j(z) \right) P_\epsilon(z, y^*) d\pi(w)\, d\pi(z). \quad (10)$$

where we have used the orthogonality of the eigenfunctions.

As noted previously, we use $\{z^i\}_{i=1}^N$ to represent the training samples and use $\{x_t^i\}_{i=1}^M$ to represent the generated samples at time t. Focusing on a single time step, we drop the dependence on t. Then the empirical approximation of (10) is as follows:

$$\nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(x^i, x^j)$$

$$= \sum_{k_1=1}^{N} \sum_{k_2=1}^{N} \nabla_1 P_{\epsilon,N}(x^i, z^{k_1}) \left(\sum_{k_3=1}^{N} \phi_{k_3}(z^{k_1})\lambda_{k_3}^{-1}\sigma_{k_3}^{-1}\lambda_{k_3}^{-1}\phi_{k_3}(z^{k_2}) \right) P_{\epsilon,N}(z^{k_2}, x^j).$$
$$\tag{11}$$

In the matrix representation, the three matrices (from left to right) on the right-hand side above are of size $M \times N$, $N \times N$, and $N \times M$. The ingredients for computing the expression above are $P_{\epsilon,N}(\cdot,\cdot), \nabla_1 P_{\epsilon,N}(\cdot,\cdot), \lambda_i, \sigma_i, \phi_i$. Since $P_{\epsilon,N}$ is constructed using Gaussian kernels, its derivative with respect to the first argument $\nabla_1 P_{\epsilon,N}(\cdot,\cdot)$ can be computed in closed form (see Appendix B.1). We therefore obtain an implementable algorithm.

5.2. **Algorithm for generative modeling.** Algorithm 1 summarizes our proposed scheme, called a diffusion map particle system (DMPS). We offer several comments. The classical analysis of diffusion maps requires the underlying distribution to have bounded support, but we find that this algorithm works well even

---

Algorithm 1: Diffusion map particle system (DMPS)

Input: Training samples $\{z^i\}_{i=1}^N \sim \pi$ and initial particles $\{x_0^i\}_{i=1}^M$, tolerance tol, bandwidth ϵ, step size h
Output: $\{x_T^i\}_{i=1}^M$
Construct $P_{\epsilon,N}$ using $\{z^i\}_{i=1}^N$ as in (8).
Compute the eigenpairs $\{\lambda_i, \phi_i\}$ such that $P_{\epsilon,N}(x,y) = \sum_{i=1}^N \lambda_i \phi_i(x)\phi_i(y)$,
i.e., by performing singular value decomposition on the kernel matrix $P_{\epsilon,N}$.
while tol not met **do**
    for $i = 1, \cdots, M$ **do**
      $x_{t+1}^i = x_t^i - \frac{h}{M}\sum_{j=1}^M \nabla_1 K_{\mathscr{L}^{-1},\epsilon,N}(x_t^i, x_t^j)$ as in (11).
    end for
end while

---

when the support of π is (in principle) unbounded. We suggest initializing the samples $\{x_0^i\}_{i=1}^M$ inside the support of π. Even though initializing samples outside the support would work because of the finite bandwidth ϵ, starting the samples inside the support generally makes the algorithm more stable. In practice, we choose the bandwidth as $\epsilon = \text{med}^2/(2\log N)$, following the heuristics proposed in [31], where med denotes the median of the pairwise distances between training samples. A small ϵ results in particles not diffusing sufficiently, while a large ϵ causes particles to diffuse excessively; this heuristic seeks to balance such extremes. The tolerance terminating Algorithm 1 is met if the Euclidean distance between successive positions of a particle, averaged over all particles, falls below a specified threshold for two consecutive iterations.

6. **Convergence analysis.** We comment on the convergence rate of our scheme at the population level. From (4), we see that if the kernel is exact, then the rate of change of the KL divergence is $-\mathbb{E}_\pi\left[\frac{d\mu_t}{d\pi}\mathscr{L}\mathcal{K}_\pi\frac{d\mu_t}{d\pi}\right]$. If we replace $\mathcal{K}_\pi$ by its kernel approximation L_ϵ^{-1}, then the resulting rate of change is

$$\partial_t \, \mathrm{D}_{\mathrm{KL}}(\hat{\mu}_t||\pi) = -\mathbb{E}_\pi\left[\frac{d\hat{\mu}_t}{d\pi}\mathscr{L}L_\epsilon^{-1}\frac{d\hat{\mu}_t}{d\pi}\right],$$

where $\hat{\mu}_t$ is the distribution at time t obtained from the following evolution

$$\dot{x} = \int \nabla_1 K_{\mathscr{L}^{-1},\epsilon}(x,y)d\hat{\mu}_t(y).$$

Classical results from diffusion map literature [18, 49] reveal that the bias $|\mathscr{L}f(x) - L_\epsilon f(x)| \sim \mathcal{O}(\epsilon)$ if data lie on a compact manifold. Using the same assumptions, we state the following theorem.

Theorem 6.1. *Suppose the target distribution π is supported on a compact manifold. Let μ_0 be the initial distribution of the particles and $\hat{\mu}_t$ be the distribution of the generated process at time t, and assume that $\frac{d\hat{\mu}_t}{d\pi}$ exists and is twice differentiable for all t. Then we have*

$$\mathrm{D}_{\mathrm{KL}}(\hat{\mu}_t||\pi) \leq \big(\mathcal{O}(\epsilon) + \mathrm{D}_{\mathrm{KL}}(\mu_0||\pi)\big)e^{-t} + \mathcal{O}(\epsilon).$$

Proof. From the diffusion map approximation, we have $\left| \mathscr{L} \frac{d\hat{\mu}_t}{d\pi}(x) - L_\epsilon \frac{d\hat{\mu}_t}{d\pi}(x) \right| \sim \mathcal{O}(\epsilon)$. Then we obtain that

$$\mathscr{L} \frac{d\hat{\mu}_t}{d\pi}(x) = (L_\epsilon + \mathcal{O}(\epsilon)) \frac{d\hat{\mu}_t}{d\pi}(x)$$

by factoring out $\frac{d\hat{\mu}_t}{d\pi}(x)$. Then, using the Neumann series or binomial expansion,

$$\mathscr{L}^{-1} \frac{d\hat{\mu}_t}{d\pi}(x) = (L_\epsilon + \mathcal{O}(\epsilon))^{-1} \frac{d\hat{\mu}_t}{d\pi}(x)$$

$$= \left(L_\epsilon^{-1} - \mathcal{O}(\epsilon) L_\epsilon^{-2} + \mathcal{O}(\epsilon)^2\right) \frac{d\hat{\mu}_t}{d\pi}(x)$$

$$= \left(L_\epsilon^{-1} + \mathcal{O}(\epsilon)\right) \frac{d\hat{\mu}_t}{d\pi}(x).$$

This follows from the fact that the inverse L_ϵ^{-1} is bounded. We then have

$$\mathscr{L}^{-1} = L_\epsilon^{-1} + \mathcal{O}(\epsilon).$$

We can then write

$$\mathbb{E}_\pi \left[\frac{d\hat{\mu}_t}{d\pi} \mathscr{L} L_\epsilon^{-1} \frac{d\hat{\mu}_t}{d\pi} \right]$$

$$= \mathbb{E}_\pi \left[\frac{d\hat{\mu}_t}{d\pi} \mathscr{L} \left(\mathscr{L}^{-1} \frac{d\hat{\mu}_t}{d\pi} + \mathcal{O}(\epsilon) \right) \right]$$

$$= \mathbb{E}_\pi \left[\left(\frac{d\hat{\mu}_t}{d\pi} - 1 \right)^2 \right] + \mathcal{O}(\epsilon),$$

and, consequently, by (5),

$$\partial_t \, \mathrm{D_{KL}}(\hat{\mu}_t || \pi) = -\chi^2(\hat{\mu}_t || \pi) + \mathcal{O}(\epsilon).$$

Then, using the results (Theorem 1) in [9] and Gronwall's inequality, we have

$$\mathrm{D_{KL}}(\hat{\mu}_t || \pi) \leq (\mathcal{O}(\epsilon) + \mathrm{D_{KL}}(\mu_0 || \pi)) \, e^{-t} + \mathcal{O}(\epsilon).$$

$\square$

We see that for a fixed ϵ, as $t \to \infty$, the KL divergence between the target distribution and the initial distribution vanishes exponentially, and the error is dominated by the error arising from the diffusion map approximation.

7. **Related work.** Several recent studies have explicitly considered the problem of generative modeling on manifolds. [5] construct a score-based generative model on a Riemannian manifold by coupling an Euler–Maruyama discretization on the tangent space with the exponential map to move along geodesics. Unlike the present method, however, this approach requires the underlying manifold structure to be prescribed. [6] use ideas from GANs, VAEs, and normalizing flows to learn the manifold and the density on that manifold simultaneously; [8] propose a "rectangular" normalizing flow for a similar setting, where the underlying manifold is unknown. These neural network based methods are quite complex and require significant tuning, with performance depending on the richness of the approximation families (e.g., width and depth of the networks, architectural choices in the normalizing flows). Our method is comparatively much simpler and nonparametric: the expressivity of the approximation grows with the sample size, and the only parameter that requires tuning is the kernel bandwidth. The role of manifold structure in

score-based generative modeling has also been explored recently by [39], who shows that if the learned score is sufficiently accurate, samples generated by the SGM lie exactly on the underlying manifold. Our numerical experiments below will show that SGMs can indeed successfully identify the underlying manifold structure, but are comparatively more expensive and less accurate.

A similar idea has been used in [17], where diffusion maps are employed to approximate the score function. While both approaches utilize diffusion maps, the key difference is that [17] uses diffusion maps specifically to approximate the score function, and then adopts Langevin dynamics for *stochastic* generative modeling. In contrast, our approach directly uses the diffusion map approximation of the generator of the Langevin diffusion process, without specializing this approximation to the score; this approach enables the use of LAWGD and results in *deterministic* generative modeling.

8. **Numerical experiments.** In this section, we study four numerical examples, exploring the performance of the algorithm on connected and disconnected domains and on manifolds. To benchmark the performance of DMPS, we compare it with (i) SVGD and (ii) ULA, where the score ∇V required by both algorithms is replaced by its empirical approximation using the diffusion map, as well as with (iii) the score-based generative model (SGM) of [50]. We implement SGM using a lightweight notebook from [40, 39]. To make (i) and (ii) more precise, recall that the a diffusion map can be used as a tool for approximating the Langevin generator from samples. That is, $\mathscr{L} f = \nabla^2 f - \langle \nabla V, \nabla f \rangle$. Then note that by letting f be the identity, i.e., $f(x) = x$, we have that $\mathscr{L}(x) = \nabla \log \pi$. Therefore, we can use samples to approximate the gradient of the potential. For DMPS and SVGD, we run the algorithm until a prescribed tolerance is met and we run ULA and SGM for a fixed number of iterations.

To evaluate the quality of samples generated with each method, we compute the regularized optimal transport (OT) distance between the generated samples and reference samples from the target distribution. We compute this distance using the Sinkhorn–Knopp algorithm [12, 27]. The cost matrix is set to be the pairwise distance between the reference samples and generated particles, and each sample is assigned equal weight marginally. The number of reference samples is chosen to be large to mitigate error in the OT distance resulting from discretization of the target: 20000 for the first three examples and 50000 for the last one due to its higher dimension. The entropic regularization penalty $1/\lambda$ is set to be $O(10^{-2})$ in the Mickey mouse, two moons, arc, and hyper-semisphere examples, and $O(10^{-3})$ in the high energy physics example. For the Mickey mouse, two moons, arc, and hyper-semisphere examples, each experiment is repeated 10 times for reproducibility; this replication involves sampling new training data and repeating all steps of each algorithm. For the high energy physics example, the experiment is conducted for 30 different physical particles. For the Mickey mouse, two moons, arc, and high energy physics examples, the number of generated particles is varied over $\{100, 300, 900, 2700\}$; for the hyper-semisphere example, it is fixed to 300.

8.1. **Mickey mouse: Two-dimensional connected domain.** In this example, the target distribution is uniform over a compactly supported Mickey mouse-shaped domain. The generative process is initiated uniformly inside a circle. Results are obtained with both 1000 and 2000 training samples. In Figure 1, we show the initial particles, the generated particles, and the target distribution. Both methods
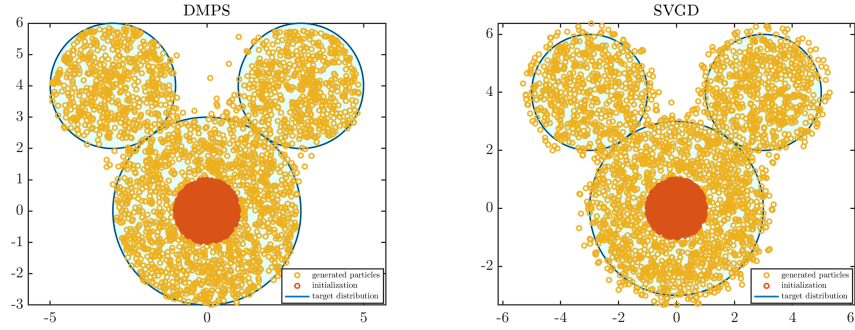
FIGURE 1. Mickey mouse: 2700 generated particles using DPMS
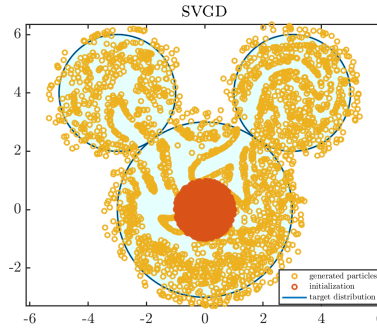and SVGD, with 2000 training samples

FIGURE 2. Mickey mouse: an instance of running the SVGD gener-
ative model shows strange non-uniform pattern with 1000 training
samples and 2700 generated particles

capture the shape relatively well. However, particles generated from SVGD move
out of the domain, while most of the particles generated using DMPS stay inside.
In some cases, the SVGD-generated particles exhibit a non-uniform pattern; see
Figure 2. Figure 3 shows quantitative comparisons of the error.

8.2. **Two moons: Two-dimensional disconnected domain.** In this exam-
ple, the target distribution is uniform and compactly supported on a two-moon-
shaped domain. In contrast with the previous example, the domain is disconnected.
Though the underlying distribution has zero density outside the support, the finite
kernel bandwidth enables the methods to be implementable in this case. Results are
obtained with 500 and 1000 training samples. We show the initial particles, target
distribution, and particles generated with DMPS, SVGD, and ULA in Figure 4, and
the regularized OT distance in Figure 5. As we can see in Figure 4, SVGD does not
explore the very end of the domain, and ULA has many samples that diffuse out of
the support. The error plot (Figure 5) shows that DMPS enjoys the smallest error
in terms of OT distance, and that this the error decreases with more generated
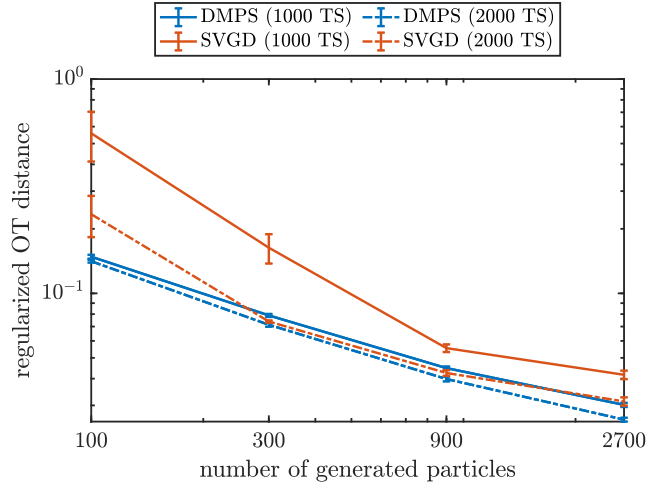
FIGURE 3. Mickey mouse: error comparison between DMPS, SVGD
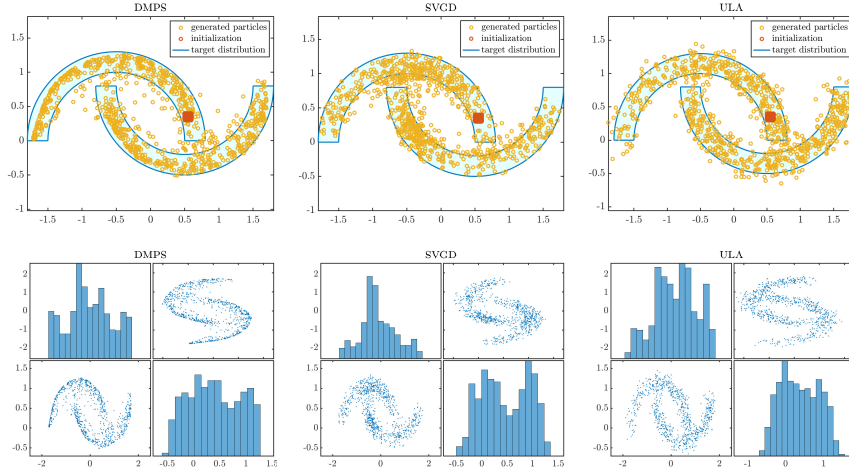

FIGURE 4. Two moons: 900 generated particles from DMPS, SVGD, and ULA with 500 training samples

particles. While ULA shows a similar convergence (with larger values of error), the error of SVGD fluctuates as more particles are included.

8.3. **The arc: One-dimensional manifold embedded in a three-dimensional space.** We now consider an example where the data lie on a manifold, in this case an arc of radius 1 embedded in $\mathbb{R}^3$. Training data are drawn uniformly from the arc and perturbed in the radial direction only, with $U(0, 10^{-2})$ noise. Results are obtained with both 100 and 1000 training samples. The initial particles and the target distribution are shown in Figure 6 (left). We then run DMPS, SVGD, ULA, and SGM for each batch of training and initialization samples, visualizing an instance in
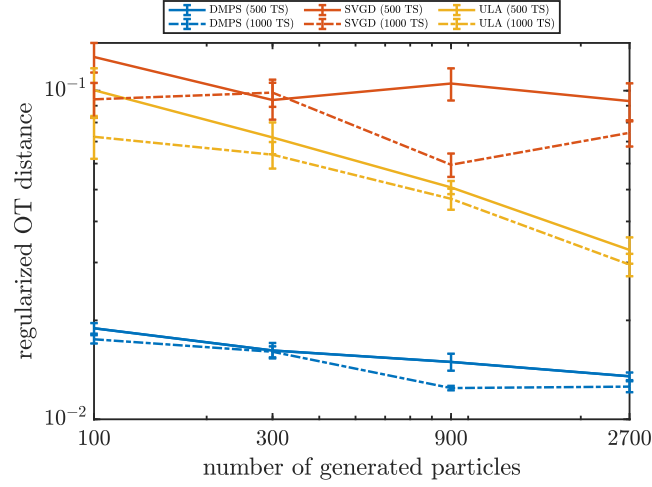
FIGURE 5. Two moons: error comparison between DMPS, SVGD, and ULA. Solid lines use 500 training samples, dashed lines use 1000.

Figure 6. Particles generated by the two deterministic methods, DMPS and SVGD, lie only on the two-dimensional plane of the training data, but the particles generated using SVGD do not fully explore the target distribution. Particles generated by the two stochastic methods, ULA and SGM, span the full three-dimensional space due to the added noise. Errors are plotted in Figure 7, for both choices of training set size. We see that DMPS exhibits the smallest errors, and that this error decreases as we increase the number of generated particles. SGM shows a similar trend, but with larger errors. The performance of ULA does not seem to improve after using more training samples, which might be due to finite discretization timestep (although it was chosen small relative to the width of the target, $h = 5 \times 10^{-4}$). SVGD gives the largest errors, which do not seem to decrease with more particles.

8.4. **Hyper-semisphere: 2 to 14-dimensional manifolds.** We finally study an example where data are uniformly sampled on a half-sphere embedded in ambient dimensions $d \in \{3, 6, 9, 12, 15\}$; in each case, the manifold is thus of dimension $d-1$. For this problem, the number of training samples and the number of generated particles are fixed to 1000 and 300, respectively. A visualization for $d = 3$ can be seen in Figure 8. We show the error (in OT distance), and the standard error of the mean error over 10 trials, in Table 1. For all dimensions, DMPS enjoys the smallest error and the smallest standard error, followed by SGM and ULA, which also produce relatively small errors and stable results. SVGD has the largest error and does not produce stable results (large variability over the 10 trials).

8.5. **High energy physics: Gluon jet dataset.** We now study a real-world example from high energy physics, where the goal is to generate relative angular coordinates $\eta^{\mathrm{rel}}, \phi^{\mathrm{rel}}$ and relative transverse momenta p_T^{rel} of elementary physical particles produced in a gluon jet. Details on the dataset are in [23]. The dataset is of dimension $177252 \times 30 \times 4$, meaning that there are 177252 jets, 30 physical

(a) Initialization and target distribution

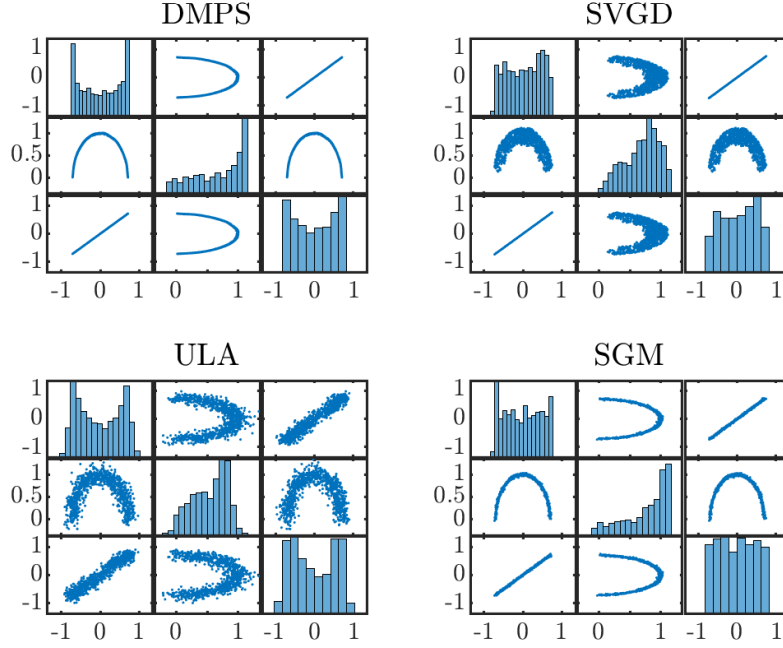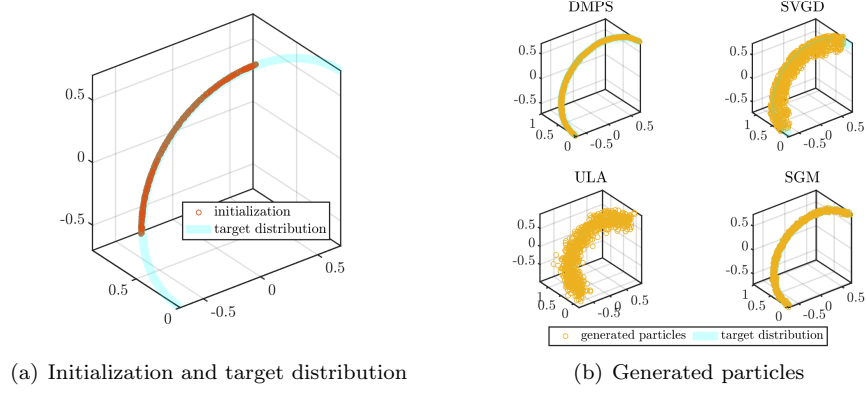(b) Generated particles

(c) Marginal distributions

FIGURE 6. The arc: 900 generated particles using DMPS, SVGD, ULA, and SGM with 1000 training samples.

particles per jet, and each particle is characterized by four distinct features: η^{rel}, ϕ^{rel}, p_T^{rel}, and a binary mask. Since the value of the last feature is either 0 or 1, we only use the first three features for the propose of generative modeling. Therefore, for each physical particle, there are 177252 available samples, and each sample is of 3 dimensions. We then train a generative model for each of the 30 physical particles. To train each model, we normalize the training data so that they have mean zero and marginal variances of one. We use 1000 samples for training, which
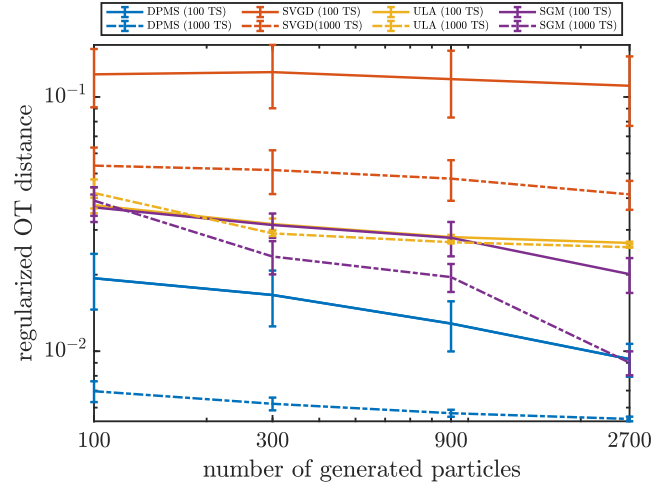
FIGURE 7. The arc: error comparison between DMPS, SVGD, ULA, and SGM. Solid lines use 100 training samples; dashed lines use 1000.
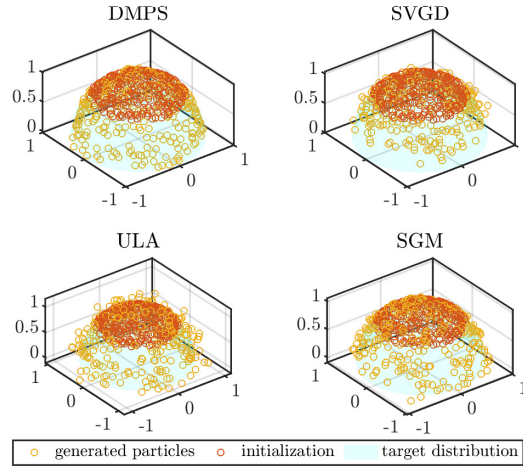

FIGURE 8. Hyper-semisphere: 300 generated samples using DMPS, SVGD, ULA, and SGM in three dimensions with 1000 training samples.

	DMPS	SVGD	ULA	SGM
$d = 3$	**0.018** $\pm$ 0.0003	0.146 ± 0.0229	0.033 ± 0.0006	0.032 ± 0.0024
$d = 6$	**0.142** $\pm$ 0.0003	0.267 ± 0.0171	0.185 ± 0.0012	0.170 ± 0.0022
$d = 9$	**0.303** $\pm$ 0.0003	0.361 ± 0.0077	0.378 ± 0.0011	0.348 ± 0.0025
$d = 12$	**0.441** $\pm$ 0.0002	0.811 ± 0.0783	0.555 ± 0.0018	0.496 ± 0.0041
$d = 15$	**0.564** $\pm$ 0.0009	0.986 ± 0.0055	0.713 ± 0.0025	0.608 ± 0.0024

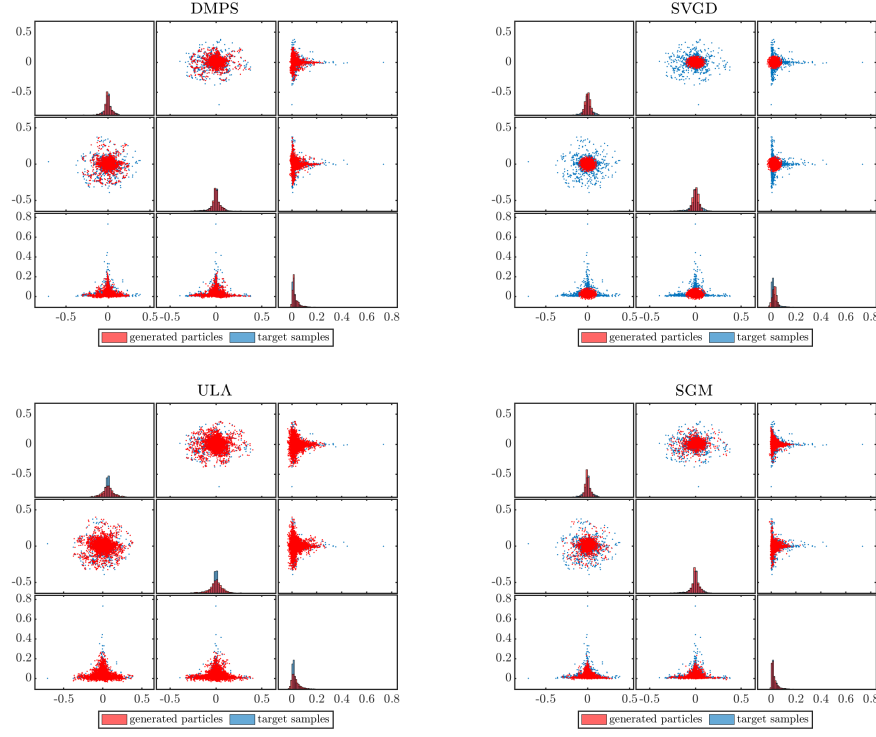TABLE 1. Hyper-semisphere: error comparison ($\pm$ standard error) between DMPS, SVGD, ULA, and SGM.

FIGURE 9. Gluon jet dataset: marginal distributions of the target samples and 2700 generated particles using DMPS, SVGD, ULA, and SGM with 1000 traning samples. Coordinates are η^{rel}, ϕ^{rel}, and p_T^{rel}, respectively.

are drawn randomly from the full set of 177252 samples, and initialize 100, 300, 900, or 2700 samples from $U(-1, 1)^3$ for the generative process. We also compare SVGD, ULA, and SGM with the same setup: 1000 training samples and increasing numbers of generated samples. Errors ($\pm$ standard error) for all four methods are shown in Table 2. These results are averaged over all 30 different physical particles. We see that DMPS consistently outperforms the other methods across all cases in this problem. In Figure 9, we also show the marginal distributions of 2700 generated samples (in red) using different methods and plot target samples (in blue) as a reference, for the first physical particle. As we can see, samples generated using DMPS, ULA, and SGM resemble those from the target distribution. Samples generated using SVGD, however, qualitatively fail to capture the target distribution, consistent with the larger errors in Table 2.

8.6. **Remarks on the experiments.** In these experiments, we see that for all methods the error decreases with more training samples, and that the error of DMPS, ULA, and SGM decreases with more generated particles. We also observe that DMPS has the best performance in terms of the regularized OT metric. The reason might be that the kernel method for approximating the generator relies on diffusion maps, or more broadly, the graph Laplacian, which is widely used for

# particles	DMPS	SVGD	ULA	SGM
100	**0.0020**	0.0062	0.0041	0.0029
	$\pm 5.03 \times 10^{-5}$	$\pm\, 4.00 \times 10^{-4}$	$\pm\, 2.64 \times 10^{-4}$	$\pm\, 1.21 \times 10^{-4}$
300	**0.0014**	0.0059	0.0032	0.0021
	$\pm 1.65 \times 10^{-5}$	$\pm\, 3.75 \times 10^{-4}$	$\pm\, 1.73 \times 10^{-4}$	$\pm\, 0.55 \times 10^{-4}$
900	**0.0012**	0.0058	0.0029	0.0018
	$\pm 1.61 \times 10^{-5}$	$\pm\, 3.65 \times 10^{-4}$	$\pm\, 1.53 \times 10^{-4}$	$\pm\, 0.46 \times 10^{-4}$
2700	**0.0012**	0.0057	0.0027	0.0018
	$\pm 1.59 \times 10^{-5}$	$\pm\, 3.61 \times 10^{-4}$	$\pm\, 1.30 \times 10^{-4}$	$\pm\, 0.44 \times 10^{-4}$

TABLE 2. Gluon jet dataset: error comparison ($\pm$ standard error) between DMPS, SVGD, ULA, and SGM, for different numbers of generated particles.

manifold learning due to its flexibility in detecting the underlying geometry. The fact that the generator is approximated as a whole distinguishes it from other score-based methods.

Although the gradient of the potential term is approximated in the same fashion in our SVGD approach, performance might be affected by the interaction between two kernels: one from the diffusion map approximation and the other being the kernel of the SVGD algorithm itself, as well as the choice of the kernel bandwidths. Performance is more stable for the two non-deterministic methods, ULA and SGM, compared to SVGD. For ULA, since Gaussian noise is added at every step, particles are able to "leave" the manifold for any finite stepsize. On the other hand, SGM produces generally good results. However, it does suffer from having longer training times. For the arc problem, with 1000 training samples and 100 initial particles, DMPS took 6.21 seconds while SGM (trained using 5000 epochs) took 1633 seconds (27 minutes).

9. **Conclusion.** We introduced DMPS as a *simple-to-implement* and *computationally efficient* kernel method for generative modeling. Our approach combines diffusion maps with the LAWGD approach to construct a generative particle system that adapts to the geometry of the underlying distribution. Our method compares favorably with other competing schemes (SVGD and ULA with learned scores, and diffusion-based generative models) on synthetic datasets, consistently achieving the smallest errors in terms of regularized OT distance. While the examples presented here are of moderate dimension (up to $d = 15$ for the example in Section 8.4), we expect that more sophisticated kernel methods [28] can slot naturally into the DMPS framework and help extend the method to higher-dimensional problems. Future research will also study the convergence rate of the method in discrete time and with finite samples, and consider more complex geometric domains.

Appendix A. Spectral properties of $\mathscr{L}$. Suppose $\mathscr{L} = -\nabla^2 + \langle \nabla V, \nabla \cdot \rangle$ is an operator defined on a domain $\mathfrak{D} \subset L^2(\pi)$ with discrete spectrum $\lambda_0 = 0 < \lambda_1 \leq \lambda_2 \leq \cdots$. Let ϕ_i be the corresponding eigenfunctions. The action of $\mathscr{L} : \mathfrak{D} \to L^2(\pi)$

reads

$$\mathscr{L}f(x) = \sum_{i=1}^{\infty} \lambda_i \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i(x),$$

where, since $\lambda_0 = 0$, the summation above could equivalently start at $i = 0$ or at $i = 1$. We define $\mathscr{L}^{-1}$ via

$$\mathscr{L}^{-1}f(x) := \sum_{i=1}^{\infty} \lambda_i^{-1} \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i(x).$$

Note that we exclude the eigenfunction corresponding to $\lambda_0 = 0$. Therefore, if we write

$$f = \langle \phi_0, f \rangle_{L^2(\pi)} \phi_0 + \sum_{i=1}^{\infty} \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i = \int f d\pi + \sum_{i=1}^{\infty} \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i,$$

then

$$\mathscr{L}^{-1}\left(f - \int f d\pi\right) = \sum_{i=1}^{\infty} \lambda_i^{-1} \langle \phi_i, f \rangle_{L^2(\pi)} \phi_i = \mathscr{L}^{-1}f,$$

where we have used the fact that $\phi_0 = 1$ is the eigenfunction corresponding to $\lambda_0 = 0$. Note that $\mathscr{L}^{-1}\mathscr{L}f = f - \int f d\pi$. Therefore, if $\int f d\pi = 0$, $\mathscr{L}^{-1}\mathscr{L}f = f$, and $\mathscr{L}^{-1}$ can be understood as the inverse of $\mathscr{L}$ on the space orthogonal to the constant functions.

Appendix B. Finite sample approximation of the operator. In this section, we introduce the finite sample counterpart to the approximations in Section 4.1. We consider the approximation of the generator of the Langevin diffusion process $\mathscr{L}$ from finite samples $\{z^i\}_{i=1}^{N} \sim \pi$. In this case, we have

$$M_{\epsilon,N}(x, y) = \frac{K_\epsilon(x, y)}{\sqrt{\sum_{i=1}^{N} K_\epsilon(z^i, y)} \sqrt{\sum_{i=1}^{N} K_\epsilon(x, z^i)}},$$

and the corresponding $L^f_{\epsilon,N}$ and $L^b_{\epsilon,N}$ can be written as

$$P^f_{\epsilon,N}(x, y) := \frac{M_{\epsilon,N}(x, y)}{\sum_{i=1}^{N} M_{\epsilon,N}(z^i, y)},$$

$$P^b_{\epsilon,N}(x, y) := \frac{M_{\epsilon,N}(x, y)}{\sum_{i=1}^{N} M_{\epsilon,N}(x, z^i)},$$

and we set

$$P_{\epsilon,N}(x, y) = \frac{1}{2}\left(P^f_{\epsilon,N}(x, y) + P^b_{\epsilon,N}(x, y)\right). \tag{12}$$

Similarly, its action on a function g writes

$$T^{f,b}_{\epsilon,N}g(x) = \sum_{i=1}^{N} P^{f,b}_{\epsilon,N}(x, z^i)g(z^i).$$

Let

$$L^{f,b}_{\epsilon,N} := \frac{\mathrm{Id} - T^{f,b}_{\epsilon,N}}{\epsilon}.$$

Similar to their spatial continuum limit, we have

$$\lim_{\epsilon \to 0, N \to \infty} L^f_{\epsilon,N} = \lim_{\epsilon \to 0, N \to \infty} L^b_{\epsilon,N} = \mathscr{L}.$$

Then, similar to (7), we have that

$$\lim_{\epsilon \to 0, N \to \infty} \frac{g(x) - \sum_{i=1}^{N} P^{f,b}_{\epsilon,N}(x, z^i) g(z^i)}{\epsilon} = \mathscr{L}g(x),$$

and

$$L_{\epsilon,N} = \frac{1}{2}(L^f_{\epsilon,N} + L^b_{\epsilon,N})$$

is the symmetric kernel.

B.1. Computing $\nabla_1 P_\epsilon(x, y)$. Recall that $P_\epsilon = \frac{1}{2}(P^f_\epsilon + P^b_\epsilon)$. We then compute $\nabla_1 P^f_\epsilon(x, y)$ and $\nabla_1 P^b_\epsilon(x, y)$ separately.

B.1.1. Computing $\nabla_1 P^f_\epsilon(x, y)$. Recall

$$P^f_\epsilon(x, y) := \frac{M_\epsilon(x, y)}{\int_X M_\epsilon(x, y) d\pi(x)},$$

where

$$M_\epsilon(x, y) := \frac{K_\epsilon(x, y)}{\sqrt{\int_{\mathcal{X}} K_\epsilon(x, y) d\pi(x)} \sqrt{\int_{\mathcal{Y}} K_\epsilon(x, y) d\pi(y)}}. \tag{13}$$

Then

$$\nabla_1 P^f_\epsilon(x, y) = \frac{\nabla_1 M_\epsilon(x, y)}{\int_X M_\epsilon(x, y) d\pi(x)}.$$

We then compute $\nabla_1 M_\epsilon(x, y)$. Let $d_\epsilon(x) = \int_{\mathcal{Y}} K_\epsilon(x, y) d\pi(y)$ and $d_\epsilon(y) = \int_{\mathcal{X}} K_\epsilon(x, y) d\pi(x)$. Then

$$\nabla_1 M_\epsilon(x, y) = \frac{\nabla_1 K_\epsilon(x, y) \sqrt{d_\epsilon(y)} \sqrt{d_\epsilon(x)} - \left(\frac{\partial}{\partial x} \sqrt{d_\epsilon(x)}\right) \sqrt{d_\epsilon(y)} K_\epsilon(x, y)}{d_\epsilon(x) d_\epsilon(y)},$$

where

$$\nabla_1 K_\epsilon(x, y) = -\left(\frac{x - y}{\epsilon}\right) e^{-\frac{\|x - y\|^2}{2\epsilon}}, \tag{14}$$

$$\frac{\partial}{\partial x}\left(\sqrt{d_\epsilon(x)}\right) = \frac{1}{2} \int_{\mathcal{Y}} \left(-\left(\frac{x - y}{\epsilon}\right) e^{-\frac{\|x - y\|^2}{2\epsilon}}\right)^{-1/2} d\pi(y). \tag{15}$$

B.1.2. Computing $\nabla_1 P^b_\epsilon(x, y)$. On the other hand, we have

$$P^b_\epsilon(x, y) := \frac{M_\epsilon(x, y)}{\int_{\mathcal{Y}} M_\epsilon(x, y) d\pi(y)},$$

and

$$\nabla_1 P^b_\epsilon(x, y) = \frac{\nabla_1 M_\epsilon(x, y) \int_{\mathcal{Y}} M_\epsilon(x, y) d\pi(y) - \int_{\mathcal{Y}} \nabla_1 M_\epsilon(x, y) d\pi(y) M_\epsilon(x, y)}{\left(\int_{\mathcal{Y}} M_\epsilon(x, y) d\pi(y)\right)^2},$$

where all the ingredients are computable from (13), (14), and (15).

B.2. **Computing $\nabla_1 P_{\epsilon,N}(x,y)$.** The discrete version is obtained by replacing the integral with its empirical average. Similarly, let $\{z^i\}_{i=1}^N \sim \pi$, and define $d_{\epsilon,N}(x) = \sum_{i=1}^N K_\epsilon(x, z^i)$ and $d_{\epsilon,N}(y) = \sum_{i=1}^N K_\epsilon(z^i, y)$. Recall that

$$M_{\epsilon,N}(x,y) = \frac{K_\epsilon(x,y)}{\sqrt{\sum_{i=1}^N K_\epsilon(z^i, y)}\sqrt{\sum_{i=1}^N K_\epsilon(x, z^i)}}.$$

Then

$$\nabla_1 M_{\epsilon,N}(x,y)$$
$$= \frac{\nabla_1 K_{\epsilon,N}(x,y)\sqrt{d_{\epsilon,N}(y)}\sqrt{d_{\epsilon,N}(x)} - \left(\frac{\partial}{\partial x}\sqrt{d_{\epsilon,N}(x)}\right)\sqrt{d_{\epsilon,N}(y)}K_\epsilon(x,y)}{d_{\epsilon,N}(x)d_{\epsilon,N}(y)},$$

and

$$\frac{\partial}{\partial x}\left(\sqrt{d_{\epsilon,N}(x)}\right) = \frac{1}{2}\sum_{i=1}^N \left(-\left(\frac{x-z^i}{\epsilon}\right)e^{-\frac{\|x-z^i\|^2}{2\epsilon}}\right)^{-1/2}.$$

Finally, similar to the previous section, we have that

$$\nabla_1 P_{\epsilon,N}^f(x,y) = \frac{\nabla_1 M_{\epsilon,N}(x,y)}{\sum_{i=1}^N M_{\epsilon,N}(z^i, y)},$$

and

$$\nabla_1 P_{\epsilon,N}^b(x,y)$$
$$= \frac{\nabla_1 M_{\epsilon,N}(x,y)\left(\sum_{i=1}^N M_{\epsilon,N}(x,z^i)\right) - \left(\sum_{i=1}^N \nabla_1 M_{\epsilon,N}(x,z^i)\right)M_{\epsilon,N}(x,y)}{\left(\sum_{i=1}^N M_{\epsilon,N}(x,z^i)\right)^2}.$$

Appendix C. Symmetric positive-definiteness of P_ϵ. In this section we show that $P_\epsilon = \frac{1}{2}(P_\epsilon^f + P_\epsilon^b)$ is symmetric positive definite.

Theorem C.1. $P_\epsilon : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ *is a positive definite kernel.*

Proof. Recall that

$$P_\epsilon^f(x,y) := \frac{M_\epsilon(x,y)}{\int M_\epsilon(x,y)d\pi(x)},$$

$$P_\epsilon^b(x,y) := \frac{M_\epsilon(x,y)}{\int M_\epsilon(x,y)d\pi(y)},$$

where

$$M_\epsilon(x,y) := \frac{K_\epsilon(x,y)}{\sqrt{\int K_\epsilon(x,y)d\pi(x)}\sqrt{\int K_\epsilon(x,y)d\pi(y)}},$$

and K_ϵ is the Gaussian kernel, which is symmetric positive definite. Therefore, we have

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j K_\epsilon(x^i, x^j) > 0,$$

for all $n \in \mathbb{N}$, $x^1, \ldots, x^n \in \mathcal{D}$, and $c_1, \ldots, c_n \in \mathbb{R}$. This is equivalent to saying that K, with $K_{ij} = K_\epsilon(x^i, x^j)$, has positive eigenvalues for all choices of $n \in \mathbb{N}$, $x^1, \ldots, x^n \in \mathcal{D}$.

Then we see immediately that $M_\epsilon(x^i, x^j)$ has positive eigenvalues, as we can write $M = DKD$, where $M, K, D \in \mathbb{R}^{n \times n}$, $M_{ij} = M_\epsilon(x^i, x^j)$, and D is a diagonal matrix with positive diagonal entries $D_{ii} = 1/\sqrt{\int K_\epsilon(x^i, z) d\pi(z)}$. Using the same argument, we have that P_ϵ^f is also positive definite as $P^f = MQ^f$, where $P^f, Q^f \in \mathbb{R}^{n \times n}$ and Q^f is a diagonal matrix with $Q_{ii}^f = 1/\sqrt{\int M_\epsilon(z, x^i) d\pi(z)}$. Q^f has positive diagonal entries, as $M_\epsilon(x, y)$ is positive for all $x, y \in \mathcal{D}$. Similarly, P^b also has positive eigenvalues. Therefore, we conclude that P_ϵ is symmetric positive definite. $\qquad\square$

Appendix D. Regularity assumptions. We state here the regularity assumptions needed for the gradient to converge. The statement and the proof are adapted from [38].

Theorem D.1. *Suppose $L_\epsilon f(x)$ is a family of bounded differentiable functions from $\mathcal{D}$ to $\mathbb{R}$ converging pointwise to $\mathscr{L} f(x)$ as $\epsilon \to 0$. Furthermore, suppose $\nabla L_\epsilon f(x)$ is a family of uniformly equicontinuous functions. Then, $\mathscr{L} f(x)$ is differentiable on $\mathcal{D}$ and $\nabla L_\epsilon f(x)$ converges to $\nabla \mathscr{L} f(x)$ uniformly.*

Proof. We first choose a countable set of ϵ, say, $\epsilon = \{1/n\}_{i=1}^\infty$, and we use L_n to denote $L_{\epsilon=1/n}$ for convenience. Since $L_n f(x)$ are uniformly bounded and $\nabla L_n f(x)$ are uniformly equicontinuous, $\nabla L_n f(x)$ are uniformly bounded. Then, $\nabla L_n f(x)$ has a subsequence $\nabla L_{n(k)} f(x)$ that converges uniformly to some function $g \in C(\mathcal{D})$ by the Arzela-Ascoli theorem. We then show that $g = \nabla \mathscr{L} f(x)$ by contradiction. Suppose $\nabla L_n f(x)$ does not converge uniformly to $\nabla \mathscr{L} f(x)$. Then there exists $\epsilon > 0$ and another subsequence $\nabla L_{m(k)} f(x)$ of $\nabla L_n f(x)$ such that $\|\nabla L_{m(k)} f - \nabla \mathscr{L} f\|_\infty > \epsilon$ for all k. But, by the Arzela-Ascoli theorem, $\nabla L_{m(k)}$ has a subsequence converging uniformly to $\nabla \mathscr{L} f(x)$, a contradiction. Therefore, $\nabla L_n f(x)$ converges to $\nabla \mathscr{L} f(x)$ uniformly on $\mathcal{D}$. $\qquad\square$

REFERENCES

[1] S. Adams, N. Dirr, M. Peletier and J. Zimmer, Large deviations and gradient flows, *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, **371** (2013), 20120341, 17 pp.

[2] L. Ambrosio, N. Gigli and G. Savare, *Gradient Flows in Metric Spaces and in The Space Of Probability Measures*, Birkhäuser Verlag, Basel, 2008.

[3] N. Anand and P.-S. Huang, Generative modeling for protein structures, in *Advances in Neural Information Processing Systems* (eds. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett), vol. 31, Curran Associates, Inc., 2018, https://proceedings.neurips.cc/paper/2018/file/afa299a4d1d8c52e75dd8a24c3ce534f-Paper.pdf.

[4] M. Belkin and P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Computation*, **15** (2003), 1373-1396.

[5] V. D. Bortoli, E. Mathieu, M. Hutchinson, J. Thornton, Y. W. Teh and A. Doucet, Riemannian score-based generative modeling, *ArXiv*, arXiv:2202.02763.

[6] J. Brehmer and K. Cranmer, Flows for simultaneous manifold learning and density estimation, *ArXiv*, arXiv:2003.13913.

[7] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely and B. Hariharan, Learning gradient fields for shape generation, in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III* (eds. A. Vedaldi,

H. Bischof, T. Brox and J. Frahm), vol. 12348 of Lecture Notes in Computer Science, Springer, 2020, 364-381.

[8] A. L. Caterini, G. Loaiza-Ganem, G. Pleiss and J. P. Cunningham, Rectangular flows for manifold learning, in *Neural Information Processing Systems*, 2021.

[9] S. Chewi, T. L. Gouic, C. Lu, T. Maunu and P. Rigollet, Svgd as a kernelized wasserstein gradient flow of the chi-squared divergence, *ArXiv*, `arXiv:2006.02509`.

[10] R. R. Coifman and S. Lafon, Diffusion maps, *Applied and Computational Harmonic Analysis*, **21** (2006), 5-30, Special Issue: Diffusion Maps and Wavelets.

[11] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner and S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, *Proceedings of the National Academy of Sciences*, **102** (2005), 7426-7431.

[12] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, in *Advances in Neural Information Processing Systems* (eds. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger), vol. 26, Curran Associates, Inc., 2013, `https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf`.

[13] S. Daneri and G. Savaré, Lecture notes on gradient flows and optimal transport, 2010, `https://arxiv.org/abs/1009.3737`.

[14] G. Franzese, S. Rossi, L. Yang, A. Finamore, D. Rossi, M. Filippone and P. Michiardi, How much is enough? a study on diffusion times in score-based generative models, *Entropy*, **25** (2023), Paper No. 633, 30 pp.`https://arxiv.org/abs/2206.05173`.

[15] A. Garbuno-Iñigo, N. Nüsken and S. Reich, Affine invariant interacting langevin dynamics for bayesian inference, *SIAM J. Appl. Dyn. Syst.*, **19** (2019), 1633-1658.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems* (eds. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K. Weinberger), vol. 27, Curran Associates, Inc., 2014, `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[17] G. A. Gottwald, F. Li, Y. Marzouk and S. Reich, Stable generative modeling using diffusion maps, 2024, `https://arxiv.org/abs/2401.04372`.

[18] M. Hein, J.-Y. Audibert and U. von Luxburg, From graphs to manifolds - weak and strong pointwise consistency of graph laplacians, in *Annual Conference Computational Learning Theory*, 2005.

[19] J. Ho, X. Chen, A. Srinivas, Y. Duan and P. Abbeel, Flow++: Improving flow-based generative models with variational dequantization and architecture design, *ArXiv*, `arXiv:1902.00275`.

[20] J. Ho, A. Jain and P. Abbeel, Denoising diffusion probabilistic models, in *Advances in Neural Information Processing Systems* (eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin), vol. 33, Curran Associates, Inc., 2020, 6840-6851, `https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

[21] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi and T. Salimans, Cascaded diffusion models for high fidelity image generation, *J. Mach. Learn. Res.*, **23** (2021), Paper No. [47], 33 pp.

[22] R. Jordan, D. Kinderlehrer and F. Otto, The variational formulation of the fokker–planck equation, *SIAM Journal on Mathematical Analysis*, **29** (1998), 1-17.

[23] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, D. Gunopulos et al., Particle cloud generation with message passing generative adversarial networks, *Advances in Neural Information Processing Systems*, **34** (2021), 23858-23871.

[24] S. Kazeminia, C. Baur, A. Kuijper, B. van Ginneken, N. Navab, S. Albarqouni and A. Mukhopadhyay, Gans for medical image analysis, *Artificial Intelligence in Medicine*, **109** (2020), 101938.

[25] A. Khandelwal and R. G. Krishnan, Fine-tuning generative models, 2019.

[26] D. P. Kingma and M. Welling, Auto-encoding variational bayes, *CoRR*, `arXiv:1312.6114`.

[27] P. A. Knight, The sinkhorn-knopp algorithm: Convergence and applications, *SIAM J. Matrix Anal. Appl.*, **30** (2008), 261-275.

[28] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang and B. Póczos, MMD GAN: Towards deeper understanding of moment matching network, in *NIPS*, 2017.

[29] X. L. Li, J. Thickstun, I. Gulrajani, P. Liang and T. B. Hashimoto, Diffusion-lm improves controllable text generation, 2022, `https://arxiv.org/abs/2205.14217`.

[30] Y. Li, Q. Pan, S. Wang, T. Yang and E. Cambria, A generative model for category text generation, *Information Sciences*, **450** (2018), 301-315.

[31] Q. Liu, Stein variational gradient descent as gradient flow, in *Advances in Neural Information Processing Systems* (eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett), vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper/2017/file/17ed8abedc255908be746d245e50263a-Paper.pdf.

[32] Q. Liu and D. Wang, Stein variational gradient descent: A general purpose bayesian inference algorithm, in *NIPS*, 2016.

[33] D. Mahapatra, B. Bozorgtabar and R. Garnavi, Image super-resolution using progressive generative adversarial networks for medical image analysis, *Computerized Medical Imaging and Graphics*, **71** (2019), 30-39.

[34] Y. Miao and P. Blunsom, Language as a latent variable: Discrete generative models for sentence compression, In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (2016), 319-328.

[35] B. Nadler, S. Lafon, R. R. Coifman and I. G. Kevrekidis, Diffusion maps, spectral clustering and reaction coordinates of dynamical systems, *Applied and Computational Harmonic Analysis*, **21** (2006), 113-127, Special Issue: Diffusion Maps and Wavelets.

[36] B. Nadler, S. Lafon, R. R. Coifman and I. G. Kevrekidis, Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators, in *Advances in Neural Information Processing Systems* (eds. Y. Weiss, B. Schölkopf and J. Platt), vol. 18, MIT Press, 2005, https://proceedings.neurips.cc/paper/2005/file/2a0f97f81755e2878b264adf39cba68e-Paper.pdf.

[37] D. Nie, R. Trullo, J. Lian, C. Petitjean, S. Ruan, Q. Wang and D. Shen, Medical image synthesis with context-aware generative adversarial networks, in *Medical Image Computing and Computer Assisted Intervention (MICCAI 2017)* (eds. M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins and S. Duchesne), Springer International Publishing, Cham, 2017, 417-425.

[38] Y. Peres, Conditions for convergence of a derivative, given the function itself is convergent, *Mathematics Stack Exchange*, https://math.stackexchange.com/q/4517640, (version: 2022-08-24).

[39] J. Pidstrigach, Score-based generative models detect manifolds, *ArXiv*, arXiv:2206.01018.

[40] J. Pidstrigach, Score-based generative models introduction, https://jakiw.com/sgm_intro, 2022, Accessed: 2024-01-24.

[41] M. Ranzato, J. Susskind, V. Mnih and G. Hinton, On deep generative models with applications to recognition, in *CVPR 2011*, 2011, 2857-2864.

[42] D. J. Rezende, S. Mohamed and D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in *Proceedings of the 31st International Conference on Machine Learning* (eds. E. P. Xing and T. Jebara), vol. 32 of Proceedings of Machine Learning Research, PMLR, Bejing, China, 2014, 1278-1286, https://proceedings.mlr.press/v32/rezende14.html.

[43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, High-resolution image synthesis with latent diffusion models, 2021, https://arxiv.org/abs/2112.10752.

[44] S. T. Roweis and L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, *Science*, **290** (2000), 2323-2326.

[45] L. Ruthotto and E. Haber, An introduction to deep generative modeling, *GAMM-Mitteilungen*, **44** (2021), e202100008, 24 pp.

[46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, Improved techniques for training gans, in *Advances in Neural Information Processing Systems* (eds. D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett), vol. 29, Curran Associates, Inc., 2016, https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[47] F. Santambrogio, {Euclidean, Metric, and Wasserstein} gradient flows: An overview, *Bulletin of Mathematical Sciences*, **7** (2017), 87-154.

[48] T. R. Shaham, T. Dekel and T. Michaeli, Singan: Learning a generative model from a single natural image, *IEEE/CVF International Conference on Computer Vision (ICCV)*, (2019) 4569-4579.

[49] A. Singer, From graph to manifold laplacian: The convergence rate, *Applied and Computational Harmonic Analysis*, **21** (2006), 128-134.

[50] Y. Song and S. Ermon, Generative modeling by estimating gradients of the data distribution, in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.

[51] Y. Song and S. Ermon, Improved techniques for training score-based generative models, in *Advances in Neural Information Processing Systems* (eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin), vol. 33, Curran Associates, Inc., 2020, 12438-12448, https://proceedings.neurips.cc/paper/2020/file/92c3b916311a5517d9290576e3ea37ad-Paper.pdf.

[52] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, Score-based generative modeling through stochastic differential equations, *ArXiv*, arXiv:2011.13456.

[53] J. B. Tenenbaum, V. de Silva and J. C. Langford, Reduction a global geometric framework for nonlinear dimensionality, 2011.

[54] C. Villani, *Topics in Optimal Transportation*, *Grad. Stud. Math.*, 58, American Mathematical Society, Providence, RI, 2003.

[55] C. Villani, *Optimal Transport: Old and New*, *Grundlehren Math. Wiss.*, 338, Springer-Verlag, Berlin, 2009.

[56] A. Wibisono, Sampling as optimization in the space of measures: The langevin dynamics as a composite optimization problem, in *Annual Conference Computational Learning Theory*, 2018.

[57] Z. Wu, K. E. Johnston, F. H. Arnold and K. K. Yang, Protein sequence design with deep generative models, *Current Opinion in Chemical Biology*, **65** (2021), 18-27.

[58] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui and M.-H. Yang, Diffusion models: A comprehensive survey of methods and applications, 2022, https://arxiv.org/abs/2209.00796.

[59] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson and M. N. Do, Semantic image inpainting with deep generative models, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, Los Alamitos, CA, USA, 2017, 6882-6890.

[60] X. Yi, E. Walia and P. Babyn, Generative adversarial network in medical imaging: A review, *Medical Image Analysis*, **58** (2019), 101552.

[61] D. Yogatama, C. Dyer, W. Ling and P. Blunsom, Generative and discriminative text classification with recurrent neural networks, *ArXiv*, arXiv:1703.01898.