Large-Scale Wasserstein Gradient Flows

Anonymous Author(s)
Affiliation
Address
email

Abstract

Wasserstein gradient flows provide a powerful means of understanding and solving many diffusion equations. Specifically, Fokker-Planck equations, which model the diffusion of probability measures, can be understood as gradient descent over entropy functionals in Wasserstein space. This equivalence, introduced by Jordan, Kinderlehrer and Otto, inspired the so-called JKO scheme to approximate these diffusion processes via an implicit discretization of the gradient flow in Wasserstein space. Solving the optimization problem associated to each JKO step, however, presents serious computational challenges. We introduce a scalable method to approximate Wasserstein gradient flows, targeted to machine learning applications. Our approach relies on input-convex neural networks (ICNNs) to discretize the JKO steps, which can be optimized by stochastic gradient descent. Unlike previous work, our method does not require domain discretization or particle simulation. As a result, we can sample from the measure at each time step of the diffusion and compute its probability density. We demonstrate our algorithm's performance by computing diffusions following the Fokker-Planck equation and apply it to unnormalized density sampling as well as nonlinear filtering.

1 Introduction

Stochastic differential equations (SDEs) are used to model the evolution of random diffusion processes across time, with applications in physics [58], finance [19, 48], and population dynamics [31]. In machine learning, diffusion processes also arise in applications filtering [18, 30] and unnormalized posterior sampling via a discretization of the Langevin diffusion [65].

The time-evolving probability density ρ_t of these diffusion processes is governed by the Fokker-Planck equation. Jordan, Kinderlehrer, and Otto [28] showed that the Fokker-Planck equation is equivalent to following the gradient flow of an entropy functional in Wasserstein space, i.e., the space of probability measures with finite second order moment endowed with the Wasserstein distance. This inspired a simple minimization scheme called JKO scheme, which consists an implicit Euler discretization of the Wasserstein gradient flow. However, each step of the JKO scheme is costly as it requires solving a minimization problem involving the Wasserstein distance.

One way to compute the diffusion is to use a fixed discretization of the domain and apply standard numerical integration methods [13, 14, 15, 36, 45] to get ρ_t. For example, [46] proposes a method to approximate the diffusion based on JKO stepping and entropy-regularized optimal transport. However, these methods are limited to small dimensions since the discretization of space grows exponentially.

An alternative to domain discretization is particle simulation. It involves drawing random samples (particles) from the initial distribution and simulating their evolution via standard methods such as Euler-Maruyama scheme [32, §9.2]. After convergence, the particles are approximately distributed according to the stationary distribution, but no density estimate is readily available.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Another way to avoid discretization is to parameterize the density of ρ_t. Most methods approximate only the first and second moments ρ_t, e.g., via Gaussian approximation. Kalman filtering approaches can then compute the dynamics [29, 30, 35, 56]. More advanced Gaussian mixture approximations [1, 60] or more general parametric families have also been studied [59, 64]. In [44], variational methods are used to minimize the divergence between the predictive and the true density.

Recently, [21] introduced a parametric method to compute JKO steps via entropy-regularized optimal transport. The authors regularize the Wasserstein distance in the JKO step to ensure strict convexity and solve the unconstrained dual problem via stochastic program on a finite linear subset of basis functions. The method yields *unnormalized* probability density without direct sample access.

Recent works propose scalable continuous optimal transport solvers, parametrizing the solutions by reproducing kernels [9], fully-connected neural networks [57], or Input Convex Neural Networks (ICNNs) [33, 34, 40]. In particular, ICNNs gained attention for Wasserstein-2 transport since their gradients $\nabla\psi_\theta : \mathbb{R}^D \to \mathbb{R}^D$ can represent OT maps for the quadratic cost. These continuous solvers scale better to high dimension without discretizing the input measures, but they are too computationally expensive to be applied directly to JKO steps.

Contributions. We propose a scalable parametric method to approximate Wasserstein gradient flows via JKO stepping using input-convex neural networks (ICNNs) [5]. Specifically, we leverage Brenier's theorem to bypass the costly computation of the Wasserstein distance, and parametrize the optimal transport map as the gradient of an ICNN. Given sample access to the initial measure ρ_0, we use stochastic gradient descent (SGD) to sequentially learn time-discretized JKO dynamics of ρ_t. The trained model can sample from a continuous approximation of ρ_t and compute its density $\frac{d\rho_t}{dx}(x)$. We compute gradient flows for the Fokker-Planck free energy functional $\mathcal{F}_{\text{FP}}$ given by (5), but our method generalizes to other cases. We demonstrate performance by computing diffusion following the Fokker-Planck equation and applying it to unnormalized density sampling as well as nonlinear filtering.

Notation. $\mathcal{P}_2(\mathbb{R}^D)$ denotes the set of Borel probability measures on $\mathbb{R}^D$ with finite second moment. $\mathcal{P}_{2,ac}(\mathbb{R}^D)$ denotes its subset of probability measures absolutely continuous with respect to Lebesgue measure. For $\rho \in \mathcal{P}_{2,ac}(\mathbb{R}^D)$, we denote by $\frac{d\rho}{dx}(x)$ its density with respect to the Lebesgue measure. $\Pi(\mu, \nu)$ denotes the set of probability measures on $\mathbb{R}^D \times \mathbb{R}^D$ with marginals μ and ν. For measurable $T : \mathbb{R}^D \to \mathbb{R}^D$, we denote by $T\sharp$ the associated push-forward operator between measures.

2 Background on Wasserstein Gradient Flows

We consider gradient flows in Wasserstein space $(\mathcal{P}_2(\mathbb{R}^D), \mathcal{W}_2)$, the space of probability measures with finite second moment on $\mathbb{R}^D$ endowed with the Wasserstein-2 metric $\mathcal{W}_2$.

Wasserstein-2 distance. The (squared) Wasserstein-2 metric $\mathcal{W}_2$ between $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^D)$ is

$$\mathcal{W}_2^2(\mu, \nu) \stackrel{\text{def}}{=} \min_{\pi \in \Pi(\mu,\nu)} \int_{\mathbb{R}^D \times \mathbb{R}^D} \|x - y\|_2^2 \, d\pi(x, y), \tag{1}$$

where the minimum is over measures π on $\mathbb{R}^D \times \mathbb{R}^D$ with marginals μ and ν respectively [63].

For $\mu \in \mathcal{P}_{2,ac}(\mathbb{R}^D)$, there exists a μ-unique map $\nabla\psi^* : \mathbb{R}^D \to \mathbb{R}^D$ that is the gradient of a convex function $\psi^* : \mathbb{R}^D \to \mathbb{R} \sqcup \{\infty\}$ satisfying $\nabla\psi^*\sharp\mu = \nu$ [42]. From Brenier's theorem [12], it follows that $\pi^* = [\text{id}_{\mathbb{R}^D}, \nabla\psi^*]\sharp\mu$ is the unique minimizer of (1), i.e.,

$$\mathcal{W}_2^2(\mu, \nu) = \int_{\mathbb{R}^D} \|x - \nabla\psi^*(x)\|_2^2 \, d\mu(x).$$

Wasserstein Gradient Flows. In the Euclidean case, gradient flows along a function $f : \mathbb{R} \to \mathbb{R}$ follow the steepest descent direction and are defined through the ODE $\frac{dx_t}{dt} = -\boldsymbol{\nabla} f(x_t)$. Discretization of this flow leads to the gradient descent minimization algorithm. When functionals are defined over the space of measures equipped with the Wasserstein-2 metric, the equivalent flow is called the Wasserstein gradient flow. The idea is similar: the flow follows the steepest descent direction, but this time the notion of gradient is more complex. We refer the reader to [3] for exposition of gradient flows in metric spaces, or [54, Chapter 8] for an accessible introduction.

A curve of measures $\{\rho_t\}_{t\in\mathbb{R}_+}$ following the Wasserstein gradient flow of a functional $\mathcal{F}$ solves the continuity equation

$$\frac{\partial \rho_t}{\partial t} = \text{div}(\rho_t \nabla_x \mathcal{F}'(\rho_t)), \qquad \text{s.t. } \rho_0 = \rho^0, \tag{2}$$

where $\mathcal{F}'(\cdot)$ is the first variation of $\mathcal{F}$ [3, Theorem 8.3.1]. The term on the right can be understood as the gradient of $\mathcal{F}$ in Wasserstein space, a vector field perturbatively rearranging the mass in ρ_t to yield the steepest possible local change of $\mathcal{F}$.

Wasserstein gradient flows are used in various applied tasks. For example, gradient flows are applied in training [7, 22, 39] or refinement [6] of implicit generative models. In reinforcement learning, gradient flows facilitate policy optimization [50, 67]. Other tasks include crowd motion modelling [41, 46, 53], dataset optimization [2], and in-between animation [23].

Many applications come from the connection between Wasserstein gradient flows and SDEs. Consider an $\mathbb{R}^D$-valued stochastic process $\{X_t\}_{t\in\mathbb{R}_+}$ governed by the following Itô SDE:

$$dX_t = -\nabla\Phi(X_t)dt + \sqrt{2\beta^{-1}}dW_t, \qquad \text{s.t. } X_0 \sim \rho^0 \tag{3}$$

where $\Phi : \mathbb{R}^D \to \mathbb{R}$ is the potential function, W_t is the standard Wiener process, and $\beta > 0$ is the magnitude. The solution of (3) is called an *advection-diffusion* process. The marginal measure ρ_t of X_t at each time satisfies the *Fokker-Planck equation* with fixed diffusion coefficient:

$$\frac{\partial \rho_t}{\partial t} = \text{div}(\nabla\Phi(x)\rho_t) + \beta^{-1}\Delta\rho_t, \qquad \text{s.t. } \rho_0 = \rho^0. \tag{4}$$

Equation (4) is the Wasserstein gradient flow (2) for $\mathcal{F}$ given by the Fokker-Planck free energy functional [28]

$$\mathcal{F}_{\text{FP}}(\rho) = \mathcal{U}(\rho) - \beta^{-1}\mathcal{E}(\rho), \tag{5}$$

where $\mathcal{U}(\rho) = \int_{\mathbb{R}^D} \Phi(x)d\rho(x)$ is the *potential energy* and $\mathcal{E}(\rho) = -\int_{\mathbb{R}^D} \log \frac{d\rho}{dx}(x)d\rho(x)$ is the *entropy*. As the result, to solve the SDE (3), one may compute the Wasserstein gradient flow of the Fokker-Planck equation with the free-energy functional $\mathcal{F}_{\text{FP}}$ given by (5).

JKO Scheme. Computing Wasserstein gradient flows is challenging. The closed form solution is typically unknown, necessitating numerical approximation techniques. Jordan, Kinderlehrer, and Otto proposed a method—later abbreviated as *JKO integration*—to approximate the dynamics of ρ_t in (2) [28]. It consists of a time-discretization update of the continuous flow given by:

$$\rho^{(k)} \leftarrow \underset{\rho\in\mathcal{P}_2(\mathbb{R}^n)}{\arg\min} \left[\mathcal{F}(\rho) + \frac{1}{2h}\mathcal{W}_2^2(\rho^{(k-1)}, \rho)\right] \tag{6}$$

where $\rho^{(0)} = \rho^0$ is the initial condition and $h > 0$ is the time-discretization step size. The discrete time gradient flow converges to the continuous one as $h \to 0$, i.e., $\rho^{(k)} \approx \rho_{kh}$. The method was further developed in [3, 55], but performing JKO iterations remains challenging thanks to the minimization with respect to $\mathcal{W}_2$.

A common approach to perform JKO steps is to discretize the spatial domain. For support size $\lessapprox 10^6$, (6) can be solved by standard optimal transport algorithms [47]. In dimensions $D \geq 3$, discrete supports can hardly approximate continuous distributions and hence the dynamics of gradient flows. To tackle this issue, [21] propose a stochastic parametric method to approximate the density of ρ_t. Their method uses entropy-regularized optimal transport (OT), which is biased.

3 Computing Wasserstein Gradient Flows with ICNNs

We now describe our approach to compute Wasserstein gradient flows via JKO stepping with ICNNs.

3.1 JKO Reformulation via Optimal Push-forwards Maps

Our key idea is to replace the optimization (6) over probability measures by an optimization over convex functions, an idea inspired by [10]. Thanks to Brenier's theorem, for any $\rho \in \mathcal{P}_{2,ac}$ there

exists a unique $\rho^{(k-1)}$-measurable gradient $\nabla\psi : \mathbb{R}^D \to \mathbb{R}^D$ of a convex function ψ satisfying $\rho = \nabla\psi\sharp\rho^{(k-1)}$. We set $\rho = \nabla\psi\sharp\rho^{(k-1)}$ and rewrite (6) as an optimization over convex ψ:

$$\psi^{(k)} \leftarrow \underset{\text{Convex } \psi}{\arg\min} \left[\mathcal{F}(\nabla\psi\sharp\rho^{(k-1)}) + \frac{1}{2h}\mathcal{W}_2^2(\rho^{(k-1)}, \nabla\psi\sharp\rho^{(k-1)}) \right]. \tag{7}$$

To proceed to the next step of JKO scheme, we define $\rho^{(k)} \stackrel{\text{def}}{=} \nabla\psi^{(k)}\sharp\rho^{(k-1)}$.

Since ρ is the pushforward of $\rho^{(k-1)}$ by the gradient of a convex function $\nabla\psi$, the $\mathcal{W}_2^2$ term in (7) can be evaluated explicitly, simplifying the Wasserstein-2 distance term in (7):

$$\psi^{(k)} \leftarrow \underset{\text{Convex } \psi}{\arg\min} \left[\mathcal{F}(\nabla\psi\sharp\rho^{(k-1)}) + \frac{1}{2h}\int_{\mathbb{R}^D} \|x - \nabla\psi(x)\|_2^2 d\rho^{(k-1)}(x) \right]. \tag{8}$$

This formulation avoids the difficulty of computing Wasserstein-2 distances. An additional advantage is that we can *sample* from $\rho^{(k)}$. Since $\rho^{(k)} = [\nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}]\sharp\rho^0$, one may sample $x_0 \sim \rho^{(0)}$, and then $\nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}(x_0)$ gives a sample from $\rho^{(k)}$. Moreover, if functions $\psi^{(\cdot)}$ are strictly convex, then gradients $\nabla\psi^{(\cdot)}$ are invertible. In this case, the *density* $\frac{d\rho^{(k)}}{dx}$ of $\rho^{(k)} = \nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}\sharp\rho^0$ is computable by the change of variables formula (assuming $\psi^{(\cdot)}$ are twice differentiable)

$$\frac{d\rho^{(k)}}{dx}(x_k) = [\det \nabla^2\psi^{(k)}(x_{k-1})]^{-1} \cdots [\det \nabla^2\psi^{(1)}(x_0)]^{-1} \cdot \frac{d\rho^{(0)}}{dx}(x_0), \tag{9}$$

where $x_i = \nabla\psi^{(i)}(x_{i-1})$ for $i = 1, \dots, k$ and $\frac{d\rho^{(0)}}{dx}$ is the density of $\rho^{(0)}$.

3.2 Stochastic Optimization for JKO via ICNNs

In general, the solution $\psi^{(k)}$ of (8) is intractable since it requires optimization over all convex functions. To tackle this issue, [10] discretizes the space of convex function. The approach also requires discretization of measures $\rho^{(k)}$ limiting this method to small dimensions.

We propose to parametrize the search space using input convex neural networks (ICNNs) [5] satisfying a universal approximation property among convex functions [17]. ICNNs are parametric models of the form $\psi_\theta : \mathbb{R}^D \to \mathbb{R}$ with ψ_θ convex w.r.t. the input. ICNNs are constructed from neural network layers, with restrictions on the weights and activation functions to preserve the input-convexity, see [5, §3.1] or [33, §B.2]. The parameters are optimized via deep learning optimization techniques such as SGD.

The JKO step then becomes finding the optimal parameters θ^* for ψ_θ:

$$\theta^* \leftarrow \underset{\theta}{\arg\min} \left[\mathcal{F}(\nabla\psi_\theta\sharp\rho^{(k-1)}) + \frac{1}{2h}\int_{\mathbb{R}^D} \|x - \nabla\psi_\theta(x)\|_2^2 d\rho^{(k-1)}(x) \right]. \tag{10}$$

If the functional $\mathcal{F}$ can be estimated stochastically using random batches from $\rho^{(k-1)}$, then SGD can be used to optimize θ. $\mathcal{F}_{\text{FP}}$ given by (5) is an example of such a functional:

Theorem 1 (Estimator of $\mathcal{F}_{\text{FP}}$)**.** *Let $\rho \in \mathcal{P}_{2,ac}(\mathbb{R}^D)$ and $T : \mathbb{R}^D \to \mathbb{R}^D$ be a diffeomorphism. For a random batch $x_1, \dots, x_N \sim \rho$, the expression $[\widehat{\mathcal{U}_T}(x_1, \dots, x_N) - \beta^{-1}\widehat{\Delta\mathcal{E}_T}(x_1, \dots, x_N)]$, where*

$$\widehat{\mathcal{U}_T}(x_1, \dots, x_N) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{n=1}^{N} \Phi\big(T(x_n)\big) \text{ and}$$

$$\widehat{\Delta\mathcal{E}_T}(x_1, \dots, x_N) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{n=1}^{N} \log|\det \nabla T(x_n)|,$$

is an estimator of $\mathcal{F}_{FP}(T\sharp\rho)$ up to constant (w.r.t. T) shift given by $\beta^{-1}\mathcal{E}(\rho)$.

Proof. $\widehat{\mathcal{U}_T}$ is a straightforward unbiased estimator for $\mathcal{U}(T\sharp\rho)$. Let p and p_T be the densities of ρ and $T\sharp\rho$. Since T is a diffeomorphism, we have $p_T(y) = p(x) \cdot |\det \nabla T(x)|^{-1}$ where $x = T^{-1}(y)$. Using the change of variables formula, we write

$$\mathcal{E}(T\sharp\rho) = -\int_{\mathbb{R}^D} p_T(y) \log p_T(y) dy$$

$$
\begin{aligned}
&= -\int_{\mathbb{R}^D} p(x) \cdot |\det \nabla T(x)|^{-1} \log \left[p(x) \cdot |\det \nabla T(x)|^{-1} \right] \cdot |\det \nabla T(x)| dx \\
&= -\int_{\mathbb{R}^D} p(x) \log p(x) dx + \int_{\mathbb{R}^D} p(x) \log |\det \nabla T(x)| dx \\
&= \mathcal{E}(\rho) + \int_{\mathbb{R}^D} p(x) \log |\det \nabla T(x)| dx, \\
\implies \Delta\mathcal{E}_T(\rho) &\stackrel{\text{def}}{=} \mathcal{E}(T\sharp\rho) - \mathcal{E}(\rho) = \int_{\mathbb{R}^D} \log |\det \nabla T(x)| d\rho(x)
\end{aligned}
$$

which explains that $\widehat{\Delta\mathcal{E}_T}$ is an unbiased estimator of $\Delta\mathcal{E}_T(\rho)$. As the result, $\widehat{\mathcal{U}_T} - \beta^{-1}\widehat{\Delta\mathcal{E}_T}$ is an estimator for $\mathcal{F}_{\text{FP}}(T\sharp\rho) = \mathcal{U}(T\sharp\rho) - \beta^{-1}\mathcal{E}(T\sharp\rho)$ up to a shift of $\beta^{-1}\mathcal{E}(\rho)$. □

To apply Theorem 1 to our case, we take $T \leftarrow \nabla\psi_\theta$ and $\rho \leftarrow \rho^{(k-1)}$ to obtain a stochastic estimator for $\mathcal{F}_{\text{FP}}(\nabla\psi_\theta\sharp\rho^{(k-1)})$ in (10). Here, $\beta^{-1}\mathcal{E}(\rho^{(k-1)})$ is θ-independent and constant since $\rho^{(k-1)}$ is fixed, so the offset of the estimator plays no role in the optimization w.r.t. θ.

Algorithm 1 details our stochastic JKO method for $\mathcal{F}_{\text{FP}}$. The training is done solely based on random samples from the initial measure ρ^0: its density is not needed.

Algorithm 1: Fokker-Planck JKO via ICNNs

Input : Initial measure ρ^0 accessible by samples;
JKO discretization step $h > 0$, number of JKO steps $K > 0$;
target potential $\Phi(x)$, diffusion process temperature β^{-1};
batch size N;

Output : trained ICNN models $\{\psi^{(k)}\}_{k=1}^K$ representing JKO steps

for $k = 1, 2, \ldots, K$ **do**
  $\psi_\theta \leftarrow$ basic ICNN model;
  for $i = 1, 2, \ldots$ **do**
    Sample batch $Z \sim \rho^0$ of size N;
    $X \leftarrow \boldsymbol{\nabla}\psi^{(k-1)} \circ \cdots \circ \boldsymbol{\nabla}\psi^{(1)}(Z)$;
    $\widehat{\mathcal{W}_2^2} \leftarrow \frac{1}{N} \sum_{x \in X} \|\boldsymbol{\nabla}\psi_\theta(x) - x\|_2^2$;
    $\widehat{\mathcal{U}} \leftarrow \frac{1}{N} \sum_{x \in X} \Phi\big(\boldsymbol{\nabla}\psi_\theta(x)\big)$;
    $\widehat{\Delta\mathcal{E}} \leftarrow \frac{1}{N} \sum_{x \in X} \log\det \nabla^2\psi_\theta(x)$;
    $\widehat{\mathcal{L}} \leftarrow \frac{1}{2h}\widehat{\mathcal{W}_2^2} + \widehat{\mathcal{U}} - \beta^{-1}\widehat{\Delta\mathcal{E}}$;
    Perform a gradient step over θ by using $\frac{\partial\widehat{\mathcal{L}}}{\partial\theta}$;
  $\psi^{(k)} \leftarrow \psi_\theta$

This algorithm assumes $\mathcal{F}$ is the Fokker-Planck diffusion energy functional. However, our method admits straightforward generalization to any $\mathcal{F}$ that can be stochastically estimated; studying such functionals is a promising avenue for future work.

3.3 Computing the Density of the Diffusion Process

Our algorithm provides a *computable density* for $\rho^{(k)}$. As discussed in §3.1, it is possible to sample from $\rho^{(k)}$ while simultaneously computing the density of the samples. However, this approach does not provide a direct way to evaluate $\frac{d\rho^{(k)}}{dx}(x_k)$ for arbitrary $x_k \in \mathbb{R}^D$. We resolve this issue below.

If a convex function is strongly convex, then its gradient is bijective on $\mathbb{R}^D$. By the change of variables formula for $x_k \in \mathbb{R}^D$, it holds $\frac{d\rho^{(k)}}{dx}(x_k) = \frac{d\rho^{(k-1)}}{dx}(x_{k-1}) \cdot [\det \nabla^2\psi^{(k)}(x_{k-1})]^{-1}$ where

$x_k = \nabla\psi^{(k)}(x_{k-1})$. To compute x_{k-1}, one needs to solve the convex optimization problem:

$$x_k = \nabla\psi^{(k)}(x_{k-1}) \qquad \Longleftrightarrow \qquad x_{k-1} = \arg\min_{x\in\mathbb{R}^D} \big[\langle x, x_k\rangle - \psi^{(k)}(x)\big]. \tag{11}$$

If we know the density of ρ^0, to compute the density of $\rho^{(k)}$ at x_k we solve k convex problems

$$x_{k-1} = \arg\min_{x\in\mathbb{R}^D} \big[\langle x, x_k\rangle - \psi^{(k)}(x)\big] \qquad \dots \qquad x_0 = \arg\min_{x\in\mathbb{R}^D} \big[\langle x, x_1\rangle - \psi^{(1)}(x)\big]$$

to obtain $x_{k-1}, \dots, x_0$ and then evaluate the density as

$$\frac{d\rho_k}{dx}(x_k) = \frac{d\rho^0}{dx}(x_0) \cdot \Big[\prod_{i=1}^{k} \det \nabla^2\psi^{(i)}(x_{i-1})\Big]^{-1}.$$

Note the steps above provide a general method for tracing back the position of a particle along the flow, and density computation is simply a byproduct.

4 Experiments

In this section, we evaluate our method on toy and real-world applications. Our code is written in PyTorch. The experiments are conducted on a GTX 1080Ti. In most cases, we performed several random restarts to obtain mean and variation of the considered metric. As the result, experiments require about 100-150 hours of computation. The details are given in Appendix A.

Neural network architectures. In all experiments, we use the DenseICNN [33, Appendix B.2] architecture for ψ_θ in Algorithm 1 with *SoftPlus* activations. The network ψ_θ is twice differentiable w.r.t. the input x and has bijective gradient $\nabla\psi_\theta : \mathbb{R}^D \to \mathbb{R}^D$ with positive semi-definite Hessian $\nabla^2\psi_\theta(x) \succeq 0$ at each x. We use automatic differentiation to compute $\nabla\psi_\theta$ and $\nabla^2\psi_\theta$.

Metric. To qualitatively compare measures, we use the symmetric Kullback-Leibler divergence

$$\text{SymKL}(\rho_1, \rho_2) \stackrel{\text{def}}{=} \text{KL}(\rho_1\|\rho_2) + \text{KL}(\rho_2\|\rho_1), \tag{12}$$

where $\text{KL}(\rho_1\|\rho_2) \stackrel{\text{def}}{=} \int_{\mathbb{R}^D} \log \frac{d\rho_1}{d\rho_2}(x) d\rho_1(x)$ is the Kullback-Leibler divergence. For particle-based methods, we obtain an approximation of the distribution by kernel density estimation.

4.1 Convergence to Stationary Solution

Starting from an arbitrary initial measure ρ^0, an advection-diffusion process (4) converges to the unique stationary solution ρ^* [51] with density

$$\frac{d\rho^*}{dx}(x) = Z^{-1}\exp(-\beta\Phi(x)), \tag{13}$$

where $Z = \int_{\mathbb{R}^D} \exp(-\beta\Phi(x))dx$ is the normalization constant. This property makes it possible to compute the symmetric KL between the distribution to which our method converges and the ground truth, provided Z is known.

We use $\mathcal{N}(0, 16I_D)$ as the initial measure ρ^0 and a random Gaussian mixture as the stationary measure ρ^*. In our method, we perform $K = 40$ JKO steps with step size $h = 0.1$. We compare with a particle simulation method (with $10^3, 10^4, 10^5$ particles) based on the Euler-Maruyama $\lfloor$EM$\rceil$ approximation [32, §9.2]. We repeat the experiment 5 times and report the averaged results in Figure 1.

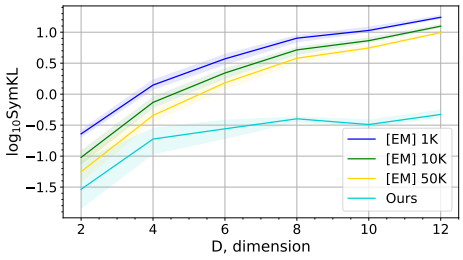
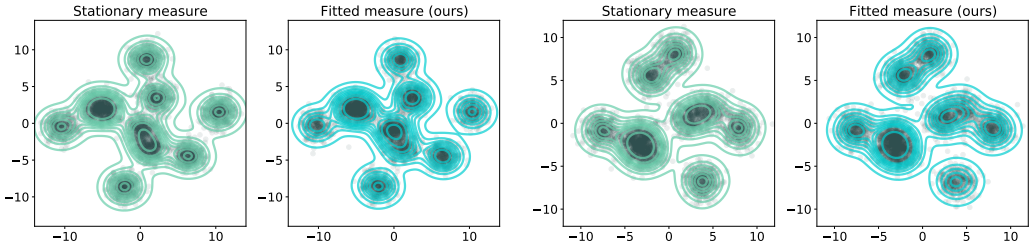


Figure 1: SymKL between the computed and the stationary measure in $D = 2, 4, \dots 12$

In Figure 2, we present qualitative results of our method converging to the ground truth in $D = 13, 32$.



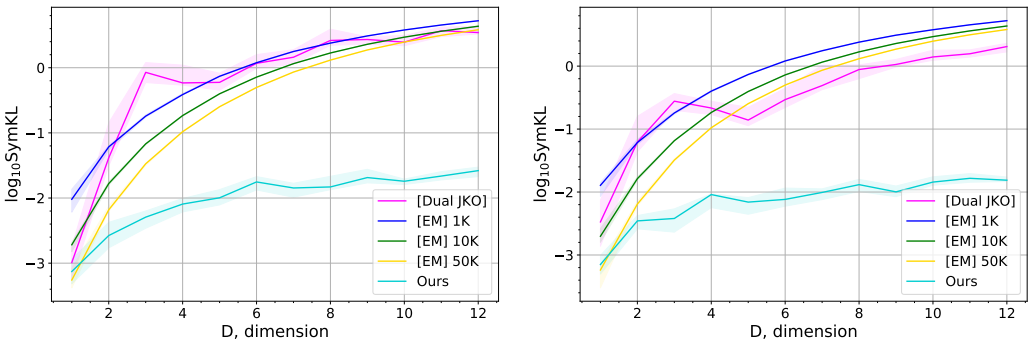
(a) Dimension $D = 13$ (b) Dimension $D = 32$

Figure 2: Projections to 2 first PCA components of the true stationary measure and the measure approximated by our method in dimensions $D = 13$ (on the left) and $D = 32$ (on the right).

4.2 Modeling Ornstein-Uhlenbeck Processes

Ornstein-Uhlenbeck processes are advection-diffusion processes (4) with $\Phi(x) = \frac{1}{2}(x-b)^T A(x-b)$ for symmetric positive definite $A \in \mathbb{R}^{D\times D}$ and $b \in \mathbb{R}^D$. They are among the few examples where we know ρ_t for any $t \in \mathbb{R}^+$ in closed form, when the initial measure ρ^0 is Gaussian [62]. This allows to quantitatively evaluate the computed dynamics of the process, not just the stationary measure.

We choose A, b at random and set ρ^0 to be the standard Gaussian measure $\mathcal{N}(0, I_D)$. We approximate the dynamics of the process by our method with JKO step $h = 0.05$ and compute SymKL between the true ρ_t and the approximate one at time $t = 0.5$ and $t = 0.9$. We repeat the experiment 15 times in dimensions $D = 1, 2 \ldots, 12$ and report the performance at in Figure 3. The baselines are $\lfloor$EM$\rceil$ with $10^3, 10^4, 5 \times 10^4$ particles and the parametric dual inference method [21] for JKO steps $\lfloor$Dual JKO$\rceil$.



(a) Time $t = 0.5$ (b) Time $t = 0.9$

Figure 3: SymKL values between the computed measure and the true measure ρ_t at $t = 0.5$ (on the left) and $t = 0.9$ (on the right) in dimensions $D = 1, 2, \ldots, 12$. Best viewed in color.

4.3 Unnormalized Posterior Sampling in Bayesian Logistic Regression

An important task in Bayesian machine learning to which our algorithm can be applied is sampling from an unnormalized posterior distribution. Given the model parameters $x \in \mathbb{R}^D$ with the prior distribution $p_0(x)$ as well as the conditional density $p(\mathcal{S}|x) = \prod_{m=1}^{M} p(s_m|x)$ of the data $\mathcal{S} = \{s_1, \ldots, s_M\}$, the posterior distribution is given by

$$p(x|\mathcal{S}) = \frac{p(\mathcal{S}|x)p_0(x)}{p(\mathcal{S})} \propto p(\mathcal{S}|x)p_0(x) = p_0(x) \cdot \prod_{m=1}^{M} p(s_m|x).$$

Computing the normalization constant $p(\mathcal{S})$ is in general intractable, underscoring the need for estimation methods that sample from $p(\mathcal{S}|x)$ given the density only up to a normalizing constant.

In our context, sampling from $p(x|\mathcal{S})$ can be solved similarly to the task in §4.1. From (13), it follows that the advection-diffusion process with temperature $\beta > 0$ and $\Phi(x) = -\frac{1}{\beta}\log\left[p_0(x)\cdot p(\mathcal{S}|x)\right]$ has $\frac{d\rho^*}{dx}(x) = p(x|\mathcal{S})$ as the stationary distribution. Thus, we can use our method to approximate the diffusion process and obtain a sampler for $p(x|\mathcal{S})$ as a result.

Dataset	Accuracy		Log-Likelihood	
	Ours	⌈SVGD⌋	Ours	⌈SVGD⌋
covtype	0.75	0.75	-0.515	-0.515
german	0.67	0.65	-0.6	-0.6
diabetis	0.775	0.78	-0.45	-0.46
twonorm	0.98	0.98	-0.059	-0.062
ringnorm	0.74	0.74	-0.5	-0.5
banana	0.55	0.54	-0.69	-0.69
splice	0.845	0.85	-0.36	-0.355
waveform	0.78	0.765	-0.485	-0.465
image	0.82	0.815	-0.43	-0.44

Table 1: Comparison of our method with ⌈SVGD⌋ [38] for Bayesian logistic regression.

The potential energy $\mathcal{U}(\rho) = \int_{\mathbb{R}^D}\Phi(x)d\rho(x)$ can be estimated efficiently by using a trick similar to the ones in stochastic gradient Langevin dynamics [65], which consists in resampling samples in $\mathcal{S}$ uniformly. For evaluation, we consider the Bayesian linear regression setup of [38]. We use the 8 datasets from [43]. The number of features ranges from 2 to 60 and the dataset size from 700 to 7400 data points. We also use the Covertype dataset[1] with 500K data points and 54 features. The prior on regression weights w is given by $p_0(w|\alpha) = \mathcal{N}(w|0,\alpha^{-1})$ with $p_0(\alpha) = \text{Gamma}(\alpha|1, 0.01)$, so the prior on parameters $x = [w, \alpha]$ of the model is given by $p_0(x) = p_0(w,\alpha) = p_0(w|\alpha)\cdot p_0(\alpha)$. We randomly split each dataset into train $\mathcal{S}_{\text{train}}$ and test $\mathcal{S}_{\text{test}}$ ones with ratio 4:1 and apply the inference on the posterior $p(x|\mathcal{S}_{\text{train}})$. In Table 1, we report accuracy and log-likelihood of the predictive distribution on $\mathcal{S}_{\text{test}}$. As the baseline, we use particle-based Stein Variational Gradient Descent [38]. We use the author's implementation with the default hyper-parameters.

4.4 Nonlinear Filtering

We demonstrate the application of our method to filtering a nonlinear diffusion. In this task, we consider a diffusion process X_t governed by the Fokker-Planck equation (4). At times $t_1 < t_2 < \cdots < t_K$ we obtain noisy observations of the process $Y_k = X_{t_k} + v_k$ with $v_k \sim \mathcal{N}(0,\sigma)$. The goal is to compute the predictive distribution $p_{t,X}(x|Y_{1:K})$ for $t \geq t_K$ given observations $Y_{1:K}$.

For each k and $t \geq t_k$ predictive distribution $p_{t,X}(x|Y_{1:k})$ follows the diffusion process on time interval $[t_k, t]$ with initial distribution $p_{t_k,X}(x|Y_{1:k})$. If $t_k = t$ then

$$p_{t_k,X}(x|Y_{1:k}) \propto p(Y_k|X_{t_k} = x)\cdot p_{t_k,X}(x|Y_{1:k-1}). \tag{14}$$

For $k = 1,\ldots,K$, we sequentially obtain the predictive distribution $p_{t_k,X}(x|Y_{1:k})$ by using the previous predictive distribution $p_{t_{k-1},X}(x|Y_{1:k-1})$. First, given access to $p_{t_{k-1},X}(x|Y_{1:k-1})$, we approximate the diffusion on interval $[t_{k-1}, t_k]$ with initial distribution $p_{t_{k-1},X}(x|Y_{1:k-1})$ by our Algorithm 1 to get access to $p_{t_k,X}(x|Y_{1:k-1})$. Next, we use (14) to get unnormalized density and Metropolis-Hastings algorithm [52] to sample from $p_{t_k,X}(x|Y_{1:k})$. We give details in Appendix B.

For evaluation, we consider the experimental setup of [21, §6.3]. We assume that the 1-dimensional diffusion process X_t has potential function $\Phi(x) = \frac{1}{\pi}\sin(2\pi x) + \frac{1}{4}x^2$ which makes the process highly nonlinear. We simulate nonlinear filtering on the time interval $t_{\text{start}} = 0$ sec., $t_{\text{fin}} = 5$ sec. and take the noise observations each 0.5 sec. The noise variance is $\sigma^2 = 1$ and $p(X_0) = \mathcal{N}(X_0|0,1)$.

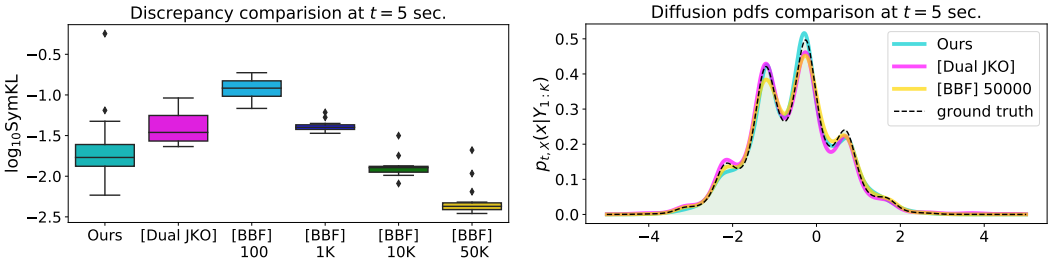
We predict the conditional density $p_{t_{\text{final}},X}(x|Y_{1:9})$ and compare the prediction with ground truth obtained with numerical integration method by Chang and Cooper [16], who use a fine discrete grid. As the baselines, we use ⌊Dual JKO⌉ [21] as well as the Bayesian Bootstrap filter ⌊BBF⌉ [24], which combines particle simulation with bootstrap resampling at observation times.

We repeat the experiment 15 times. In Figure 4a, we report the SymKL between predicted density and true $p(X_{t_{\text{fin}}}|Y_{1:9})$. We visually compare the fitted and true conditional distributions in Figure 4b.

5 Discussion

Complexity of training and sampling. Let T be the number of operations required to evaluate ICNN $\psi_\theta(x)$, and assume that the evaluation of $\Phi(x)$ in the potential energy $\mathcal{U}$ takes $O(1)$ time.

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html



(a) SymKL values.

(b) Visualized probability density functions.

Figure 4: Comparison of the predicted conditional density and true $p(X_{t_{\text{fin}}}|Y_{1:9})$.

Recall that computing the gradient is a small constant factor harder than computing the function itself [37]. Thus, evaluation of $\nabla\psi_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D$ requires $O(T)$ operations and evaluating the Hessian $\nabla^2\psi_\theta(x) : \mathbb{R}^D \to \mathbb{R}^{D\times D}$ takes $O(DT)$ time. To compute $\log\det\nabla^2\psi_\theta(x)$, we need $O(D^3)$ extra operations. Sampling from $\rho^{(k-1)} = \nabla\psi^{(k-1)} \circ \cdots \circ \nabla\psi^{(1)}\sharp\rho_0$ involves pushing $x_0 \sim \rho^0$ forward by a sequence of ICNNs $\psi^{(\cdot)}$ of length $k-1$, requiring $O\big((k-1)T\big)$ operations. The forward pass to evaluate the JKO step objective $\widehat{\mathcal{L}}$ in Algorithm 1 requires $O(DT + D^3)$ operations, as does the backward pass to compute the gradient $\frac{\partial\widehat{\mathcal{L}}}{\partial\theta}$ w.r.t. θ.

Operation	Time Complexity
Eval. ψ_θ,$\nabla\psi_\theta$,$\nabla^2\psi_\theta$	T, $O(T)$, $O(DT)$
Eval. $\log\det\nabla^2\psi_\theta$	$O(DT{+}D^3)$
Sample $x \sim \rho^{(k)}$	$O\big((k{-}1)T\big)$
Eval. $\widehat{\mathcal{L}}$ on $x \sim \rho^{(k)}$	$O(DT + D^3)$
Eval. $\frac{\partial\widehat{\mathcal{L}}}{\partial\theta}$ on $x \sim \rho^{(k)}$	$O(DT{+}D^3)$
Sample $x \sim \rho^{(k)}$ and Eval. $\frac{d\rho^{(k)}}{dx}(x)$	$O\big((k{-}1)(TD{+}D^3)\big)$

Table 2: Complexity of operations in our method for computing JKO steps via ICNNs.

The *memory complexity* is more difficult to characterize, since it depends on the autodiff implementation. It does not exceed the time complexity and is linear in the number of JKO steps k.

Wall-clock times. All particle-based methods considered in §4 and ⌊Dual JKO⌉ require from several seconds to several minutes CPU computation time. Our method requires from several minutes to few hours on GPU, the time is explained by the necessity to train a new network at each step.

Advantages. Due to using continuous approximation, our method scales well to high dimensions, as we show in §4.1 and §4.2. After training, we can produce infinitely many samples $x_k \sim \rho^{(k)}$, together with their trajectories $x_{k-1}, x_{k-2}, \ldots, x_0$ along the gradient flow. Moreover, the densities of samples in the flow $\frac{d\rho^{(k)}}{dx}(x_k), \frac{d\rho^{(k-1)}}{dx}(x_{k-1}), \ldots, \frac{d\rho^{(0)}}{dx}(x_0)$ can be evaluated immediately.

In contrast, particle-based and domain discretization methods do not scale well with the dimension (Figure 3) and provide no density. Interestingly, despite its parametric approximation, ⌊Dual JKO⌉ performs comparably to particle simulation and worse than ours (see additionally [21, Figure 3]).

Limitations. To train k JKO steps, our method requires time proportional to k^2 due to the increased complexity of sampling $x \sim \rho^{(k)}$. This may be disadvantageous for training long diffusions. In addition, for very high dimensions D, exact evaluation of $\log\det\nabla^2\psi_\theta(x)$ is time-consuming.

Future work. To reduce the computational complexity of sampling from $\rho^{(k)}$, at step k one may regress an invertible network $H : \mathbb{R}^D \to \mathbb{R}^D$ [8, 27] to satisfy $H(x_0) \approx \nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}(x_0)$ and use $H\sharp\rho_0 \to \rho^{(k)}$ to simplify sampling. An alternative is to use variational inference [11, 49, 66] to approximate $\rho^{(k)}$. To mitigate the computational complexity of computing $\log\det\nabla\psi_\theta(x)$, fast approximation can be used [25, 61]. More broadly, developing ICNNs with easily-computable exact Hessians is a critical avenue for further research as ICNNs continue to gain attention in machine learning [4, 20, 26, 33, 34, 40].

Potential impact. Diffusion processes appear in numerous scientific and industrial applications, including machine learning, finances, physics, and population dynamics. Our method will improve models in these areas, providing better scalability. Performance, however, might depend on the expressiveness of the ICNNs, pointing to theoretical convergence analysis as a key topic for future study to reinforce confidence in our model.

References

[1] Juha Ala-Luhtala, Simo Särkkä, and Robert Piché. Gaussian filtering and variational approximations for Bayesian smoothing in continuous-discrete stochastic dynamic systems. *Signal Processing*, 111:124–136, 2015.

[2] David Alvarez-Melis and Nicolò Fusi. Gradient flows in dataset space. *arXiv preprint arXiv:2010.12760*, 2020.

[3] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[4] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[5] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[6] Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via Wasserstein gradient flows. *arXiv preprint arXiv:2012.00780*, 2020.

[7] Michael Arbel, Anna Korba, Adil Salim, and Arthur Gretton. Maximum mean discrepancy gradient flow. *arXiv preprint arXiv:1906.04370*, 2019.

[8] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

[9] Genevay Aude, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic optimization for large-scale optimal transport. *arXiv preprint arXiv:1605.08527*, 2016.

[10] Jean-David Benamou, Guillaume Carlier, Quentin Mérigot, and Edouard Oudet. Discretization of functionals involving the Monge–Ampère operator. *Numerische mathematik*, 134(3):611–636, 2016.

[11] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[12] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[13] Martin Burger, José A Carrillo, and Marie-Therese Wolfram. A mixed finite element method for nonlinear diffusion equations. *Kinetic & Related Models*, 3(1):59, 2010.

[14] José A Carrillo, Alina Chertock, and Yanghong Huang. A finite-volume method for nonlinear nonlocal equations with a gradient flow structure. *Communications in Computational Physics*, 17(1):233–258, 2015.

[15] JS Chang and G Cooper. A practical difference scheme for fokker-planck equations. *Journal of Computational Physics*, 6(1):1–16, 1970.

[16] J.S Chang and G Cooper. A practical difference scheme for fokker-planck equations. *Journal of Computational Physics*, 6(1):1–16, 1970.

[17] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835*, 2018.

[18] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[19] Nicole El Karoui, Shige Peng, and Marie Claire Quenez. Backward stochastic differential equations in finance. *Mathematical finance*, 7(1):1–71, 1997.

[20] Jiaojiao Fan, Amirhossein Taghvaei, and Yongxin Chen. Scalable computations of wasserstein barycenter via input convex neural networks. *arXiv preprint arXiv:2007.04462*, 2020.

[21] Charlie Frogner and Tomaso Poggio. Approximate inference with Wasserstein gradient flows. In *International Conference on Artificial Intelligence and Statistics*, pages 2581–2590. PMLR, 2020.

[22] Yuan Gao, Yuling Jiao, Yang Wang, Yao Wang, Can Yang, and Shunkang Zhang. Deep generative learning via variational gradient flow. In *International Conference on Machine Learning*, pages 2093–2101. PMLR, 2019.

[23] Yuan Gao, Guangzhen Jin, and Jian-Guo Liu. Inbetweening auto-animation via fokker-planck dynamics and thresholding. *arXiv preprint arXiv:2005.08858*, 2020.

[24] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. 1993.

[25] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic chebyshev expansions. In *International Conference on Machine Learning*, pages 908–917. PMLR, 2015.

[26] Chin-Wei Huang, Ricky TQ Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *arXiv preprint arXiv:2012.05942*, 2020.

[27] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[28] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

[29] Simon J Julier, Jeffrey K Uhlmann, and Hugh F Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 3, pages 1628–1632. IEEE, 1995.

[30] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. 1961.

[31] Søren Klim, Stig Bousgaard Mortensen, Niels Rode Kristensen, Rune Viig Overgaard, and Henrik Madsen. Population stochastic modelling (psm)—an r package for mixed-effects models based on stochastic differential equations. *Computer methods and programs in biomedicine*, 94(3):279–289, 2009.

[32] Peter E. Kloeden. *Numerical solution of stochastic differential equations / Peter E. Kloeden, Eckhard Platen*. Applications of mathematics; v. 23. Springer, Berlin, 1992.

[33] Alexander Korotin, Vage Egiazarian, Arip Asadulaev, Alexander Safin, and Evgeny Burnaev. Wasserstein-2 generative networks. In *International Conference on Learning Representations*, 2021.

[34] Alexander Korotin, Lingxiao Li, Justin Solomon, and Evgeny Burnaev. Continuous wasserstein-2 barycenter estimation without minimax optimization. In *International Conference on Learning Representations*, 2021.

[35] Harold Kushner. Approximations to optimal nonlinear filters. *IEEE Transactions on Automatic Control*, 12(5):546–556, 1967.

[36] Hugo Lavenant, Sebastian Claici, Edward Chien, and Justin Solomon. Dynamical optimal transport on discrete surfaces. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[37] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[38] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *arXiv preprint arXiv:1608.04471*, 2016.

[39] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pages 4104–4113. PMLR, 2019.

[40] Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020.

[41] Bertrand Maury, Aude Roudneff-Chupin, and Filippo Santambrogio. A macroscopic crowd motion model of gradient flow type. *Mathematical Models and Methods in Applied Sciences*, 20(10):1787–1821, 2010.

[42] Robert J McCann et al. Existence and uniqueness of monotone measure-preserving maps. *Duke Mathematical Journal*, 80(2):309–324, 1995.

[43] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pages 41–48. Ieee, 1999.

[44] Manfred Opper. Variational inference for stochastic differential equations. *Annalen der Physik*, 531(3):1800233, 2019.

[45] Lorenzo Pareschi and Mattia Zanella. Structure preserving schemes for nonlinear fokker–planck equations and applications. *Journal of Scientific Computing*, 74(3):1575–1600, 2018.

[46] Gabriel Peyré. Entropic approximation of Wasserstein gradient flows. *SIAM Journal on Imaging Sciences*, 8(4):2323–2351, 2015.

[47] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[48] Eckhard Platen and Nicola Bruti-Liberati. *Numerical solution of stochastic differential equations with jumps in finance*, volume 64. Springer Science & Business Media, 2010.

[49] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[50] Pierre H Richemond and Brendan Maginnis. On Wasserstein reinforcement learning and the Fokker-Planck equation. *arXiv preprint arXiv:1712.07185*, 2017.

[51] Hannes. Risken. *The Fokker-Planck Equation: Methods of Solution and Applications (Springer Series in Synergetics).* Springer,, 1996.

[52] Christian P Robert and George Casella. The Metropolis—Hastings algorithm. In *Monte Carlo Statistical Methods*, pages 231–283. Springer, 1999.

[53] Filippo Santambrogio. Gradient flows in Wasserstein spaces and applications to crowd movement. *Séminaire Équations aux dérivées partielles (Polytechnique)*, pages 1–16, 2010.

[54] Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

[55] Filippo Santambrogio. Euclidean, Metric, and Wasserstein gradient flows: an overview, 2016.

[56] Simo Sarkka. On unscented kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Transactions on automatic control*, 52(9):1631–1641, 2007.

[57] Vivien Seguy, Bharath Bhushan Damodaran, Rémi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. Large-scale optimal transport and mapping estimation. *arXiv preprint arXiv:1711.02283*, 2017.

[58] Kazimierz Sobczyk. *Stochastic differential equations: with applications to physics and engineering*, volume 40. Springer Science & Business Media, 2013.

[59] Tobias Sutter, Arnab Ganguly, and Heinz Koeppl. A variational approach to path estimation and parameter inference of hidden diffusion processes. *The Journal of Machine Learning Research*, 17(1):6544–6580, 2016.

[60] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D Scott. A novel gaussian sum filter method for accurate solution to the nonlinear filtering problem. In *2008 11th International Conference on Information Fusion*, pages 1–8. IEEE, 2008.

[61] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

[62] P Vatiwutipong and N Phewchean. Alternative way to derive the distribution of the multivariate ornstein–uhlenbeck process. *Advances in Difference Equations*, 2019(1):1–7, 2019.

[63] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[64] Michail D Vrettas, Manfred Opper, and Dan Cornford. Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations. *Physical Review E*, 91(1):012148, 2015.

[65] 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)*, pages 681–688. Citeseer, 2011.

[66] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[67] Ruiyi Zhang, Changyou Chen, Chunyuan Li, and Lawrence Carin. Policy optimization as Wasserstein gradient flows. In *International Conference on Machine Learning*, pages 5737–5746. PMLR, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   [Yes] *The abstract accurately reflects the paper's contributions and scope.*
   (b) Did you describe the limitations of your work?
   [Yes] *See §5.*
   (c) Did you discuss any potential negative societal impacts of your work?
   [Yes] *See §5.*
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them?
   [Yes] *We have read ethics review guidelines and ensured that our paper conforms to them.*
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results?
   [Yes] *See Theorem 1.*
   (b) Did you include complete proofs of all theoretical results?
   [Yes] *The proof of Theorem 1 is included right after the theorem statement.*
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?
   [Yes] *All the code, links to data and instructions are included in the supplementary material.*
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?
   [Yes] *See Appendix A.*

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)?
[Yes] *We report most of the results with mean and variance, see §4.*

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)?
[Yes] *See §4.*

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators?
[Yes] *We use existing datasets and cite their creators in §4.3.*

(b) Did you mention the license of the assets?
[No] *We refer to the dataset page for specific details.*

(c) Did you include any new assets either in the supplemental material or as a URL?
[No] *We refer to the dataset page for specific details.*

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating?
[No] *We refer to the dataset page for specific details.*

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content?
[No] *We refer to the dataset page for specific details.*

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable?
[N/A] *We did not use crowdsourcing or conducted research with human subjects.*

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable?
[N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation?
[N/A]