arXiv:2112.02424v2 [cs.LG] 28 Jan 2022

Variational Wasserstein gradient flow

Jiaojiao Fan [1] **Qinsheng Zhang** [1] **Amirhossein Taghvaei** [2] **Yongxin Chen** [1]

Abstract

Wasserstein gradient flow has emerged as a promising approach to solve optimization problems over the space of probability distributions. A recent trend is to use the well-known JKO scheme in combination with input convex neural networks to numerically implement the proximal step. The most challenging step, in this setup, is to evaluate functions involving density explicitly, such as entropy, in terms of samples. This paper builds on the recent works with a slight but crucial difference: we propose to utilize a variational formulation of the objective function formulated as maximization over a parametric class of functions. Theoretically, the proposed variational formulation allows the construction of gradient flows directly for empirical distributions with a well-defined and meaningful objective function. Computationally, this approach replaces the computationally expensive step in existing methods, to handle objective functions involving density, with inner loop updates that only require a small batch of samples and scale well with the dimension. The performance and scalability of the proposed method are illustrated with the aid of several numerical experiments involving high-dimensional synthetic and real datasets.

1. Introduction

The Wasserstein gradient flow models the gradient dynamics on the space of probability densities with respect to the Wasserstein metric. It was first discovered by Jordan, Kinderlehrer, and Otto (JKO) in their seminal work (Jordan et al., 1998). They pointed out that the Fokker-Planck equation is in fact the Wasserstein gradient flow of the free energy, bringing tremendous physical insights to this type of partial differential equations (PDEs). Since then, the Wasserstein gradient flow has played an important role in optimal transport, PDEs, physics, machine learning, and many other areas (Ambrosio et al., 2008; Otto, 2001; Adams et al., 2011; Santambrogio, 2017; Carlier et al., 2017; Frogner & Poggio, 2020). Despite the abundant theoretical results on the Wasserstein gradient flow established over the past decades (Ambrosio et al., 2008; Santambrogio, 2017), the computation of it remains a challenge. Most existing methods are either based on a finite difference method applied to the underlying PDEs or based on a finite dimensional optimization; both require discretization of the underlying space (Peyré, 2015; Benamou et al., 2016; Carlier et al., 2017; Li et al., 2020; Carrillo et al., 2021). The computational complexity of these methods scales exponentially with the problem dimension, making them unsuitable for the cases with probability densities over high dimensional space.

This shortcoming motivated recent line of interesting works to develop scalable algorithms utilizing neural networks (Mokrov et al., 2021; Alvarez-Melis et al., 2021; Yang et al., 2020; Bunne et al., 2021; Bonet et al., 2021). A central theme, in most of these works, is the application of the JKO scheme in combination with input convex neural networks (ICNN) (Amos et al., 2017). The JKO scheme, which is essentially a backward Euler method, is used to discretize the continuous flow in time. At each time-step, one needs to find a probability distribution that minimizes a weighted sum of squared Wasserstein distance, with respect to the distribution at the previous time-step, and the objective function. The probability distribution is then parametrized as push-forward of the optimal transport map from the previous probability distribution. The optimal transport map is represented with gradient of an ICNN utilizing the knowledge that optimal transport maps are gradient of convex functions when the transportation cost is quadratic. The problem is finally cast as stochastic optimization problem which only requires samples from the distribution.

Our paper builds on these recent works but with a crucial difference. We propose to use a variational form of the objective function, leveraging f-divergences, which has been employed in multiple machine learning applications, such as generative models (Nowozin et al., 2016), and Bayesian inference (Wan et al., 2020). The variational problem is formulated as maximization over a parametrized class of functions. The variational form allows the evaluation of the

[1]Georgia Institute of Technology [2]University of Washington, Seattle. Correspondence to: Jiaojiao Fan <jiaojiaofan@gatech.edu>.

Preprint.

objective in terms of samples, without the need for density estimation or approximating the logarithm of the determinant of the Hessian of ICNNs which appears in (Mokrov et al., 2021; Alvarez-Melis et al., 2021). Moreover, the variational form, even when restricted to a finite-dimensional class of functions, admits nice geometrical properties of its own leading to a meaningful objective function to minimize.

At the end of the algorithm, a sequence of transport maps connecting the initial distribution with the terminal distribution along the gradient flow dynamics are obtained. One can then sample from the distributions along the flow by sampling from the initial distribution (often Gaussian) and then propagating these samples through the sequence of transport maps. When the transport map is modeled by the gradient of an input convex neural network, one can also evaluate the densities at every point.

Our contributions are summarized as follows.
i) We develop a numerical algorithm to implement the Wasserstein gradient flow that is based on a variational representation of the objective functions. The algorithm does not require spatial discretization, density estimation, or approximating logarithm of determinant of Hessians.
ii) We numerically demonstrate the performance of our algorithm on several representative problems including sampling from high-dimensional Gaussian mixtures, porous medium equation, and learning generative models on MNIST and CIFAR10 datasets. We illustrate the computational advantage of our proposed method in comparison with (Mokrov et al., 2021; Alvarez-Melis et al., 2021), in terms of computational time and scalibity with the problem dimension.
iii) We establish some preliminary theoretical results regarding the proposed variational objective function. In particular, we provide conditions under which the variational objective satisfies a moment matching property and an embedding inequality with respect to a certain integral probability metric (see Prop. 4.1).

Related works: Most existing methods to compute Wasserstein gradient flow are finite difference based (Peyré, 2015; Benamou et al., 2016; Carlier et al., 2017; Li et al., 2020; Carrillo et al., 2021). These methods require spatial discretization and are thus not scalable to high dimensional settings. There is a line of research that uses particle-based method to estimate the Wasserstein gradient flow (Carrillo et al., 2019a; Frogner & Poggio, 2020). In these algorithms, the current density value is often estimated using kernel method whose complexity scales at least quadratically with the number of particles. More recently, several interesting neural network based methods (Mokrov et al., 2021; Alvarez-Melis et al., 2021; Yang et al., 2020; Bunne et al., 2021; Bonet et al., 2021) were proposed for Wasserstein gradient flow. Mokrov et al. (2021) focuses on the special case with Kullback-Leibler divergence as objective function. Alvarez-Melis et al. (2021) uses a density estimation method to evaluate the objective function by back-propagating to the initial distribution, which could become a computational burden when the number of time discretization is large. Yang et al. (2020) is based on a forward Euler time discretization of the Wasserstein gradient flow and is more sensitive to time stepsize. Bunne et al. (2021) utilizes JKO scheme to approximate a population dynamics given an observed trajectory, which finds application in computational biology. Bonet et al. (2021) replaces Wasserstein distance in JKO by sliced alternative but its connection to the original Wasserstein gradient flow remains unclear.

2. Background

2.1. Optimization problem

We are interested in developing algorithms for

$$\min_{P\in\mathcal{P}_{ac}(\mathbb{R}^n)} \mathcal{F}(P), \tag{1}$$

where $\mathcal{P}_{ac}(\mathbb{R}^n)$ is the space of probability distributions that admit density dP/dx with respect to Lebesgue measure. The objective function $\mathcal{F}(P)$ takes different form depending on the application. Three important examples are:

Example I: Kullback-Leibler divergence with respect to a given target distribution Q,

$$\mathcal{D}(P||Q) := \int \log\left(\frac{dP}{dQ}\right) dP \tag{2}$$

plays an important role in the sampling problem.

Example II: Generalized entropy

$$\mathcal{G}(P) := \frac{1}{m-1}\int P^m(x)dx, \quad m>1$$

is important for modeling the porous medium.

Example III: The (twice) Jensen-Shannon divergence

$$\mathrm{JSD}(P\|Q) := \mathcal{D}\left(P\middle\|\frac{P+Q}{2}\right) + \mathcal{D}\left(Q\middle\|\frac{P+Q}{2}\right)$$

is important in learning generative models.

2.2. Wasserstein gradient flow

Given a function $\mathcal{F}(P)$ over the space of probability densities, the Wasserstein gradient flow describes the dynamics of the probability density when it follows the steepest descent direction of the function $\mathcal{F}(P)$ with respect to the Riemannian metric induced by the 2-Wasserstein distance W_2 (Ambrosio et al., 2008). The Wasserstein gradient flow can be explicitly represented by the PDE $\frac{\partial P}{\partial t} = \nabla\cdot\left(P\nabla\frac{\delta\mathcal{F}}{\delta P}\right)$, where $\delta\mathcal{F}/\delta P$ represents the derivative of $\mathcal{F}$ with respect to the standard L_2 metric (Villani, 2003, Ch. 8).

Wasserstein gradient flow corresponds to various important PDEs depending on the choice of objective functions $\mathcal{F}(P)$. For instance, when $\mathcal{F}(P)$ is the free energy, i.e. $\mathcal{F}(P) = \int_{\mathbb{R}^n} \log dP(x) dP(x) + \int_{\mathbb{R}^n} V(x) dP(x)$, the gradient flow is the Fokker-Planck equation $\frac{\partial P}{\partial t} = \nabla \cdot (P \nabla V) + \Delta P$ (Jordan et al., 1998). When $\mathcal{F}(P)$ is the generalized entropy $\mathcal{F}(P) = \frac{1}{m-1} \int_{\mathbb{R}^n} P^m(x) dx$ for some positive number $m > 1$, the gradient flow is the porous medium equation $\frac{\partial P}{\partial t} = \Delta P^m$ (Otto, 2001; Vázquez, 2007).

2.3. JKO scheme and reparametrization

To numerically realize the Wasserstein gradient flow, a discretization over time is needed. One such discretization is the famous JKO scheme (Jordan et al., 1998)

$$P_{k+1} = \underset{P \in \mathcal{P}_{ac}(\mathbb{R}^n)}{\arg\min} \frac{1}{2a} W_2^2 (P, P_k) + \mathcal{F}(P). \tag{3}$$

This is essentially a backward Euler discretization or a proximal point method with respect to the Wasserstein metric. The solution to (3) converges to the continuous-time Wasserstein gradient flow when the step size $a \to 0$.

Recall the definition of the Wasserstein-2 distance

$$W_2^2(P, Q) = \min_{T : T\sharp P = Q} \int_{\mathbb{R}^n} \|x - T(x)\|_2^2 dP(x),$$

where the minimization is over all the feasible transport maps that transport mass from distribution P to distribution Q. Hence, (3) can be recast as an optimization in terms of the transport maps $T : \mathbb{R}^n \to \mathbb{R}^n$ from P_k to P. By defining $P = T\sharp P_k$, the optimal T is the optimal transport map from P_k to $T\sharp P_k$ and thus the gradient of a convex function φ by Brenier's Theorem (Brenier, 1991). Bunne et al. (2021); Mokrov et al. (2021); Alvarez-Melis et al. (2021) propose to paramterize T as the gradient of Input convex neural network (ICNN) (Amos et al., 2017) and express (3) as

$$\begin{aligned} P_{k+1} &= \nabla \varphi_k \sharp P_k, \\ \varphi_k &= \underset{\varphi \in \text{CVX}}{\arg\min} \frac{1}{2a} \int_{\mathbb{R}^n} \|x - \nabla \varphi(x)\|_2^2 dP_k(x) + \mathcal{F}(\nabla \varphi \sharp P_k), \end{aligned} \tag{4}$$

where CVX stands for the space of convex functions. In our method, we extend this idea and propose to reparametrize T alternatively by a residual neural network. With this reparametrization, the JKO step (3) becomes

$$\begin{aligned} P_{k+1} &= T_k \sharp P_k, \\ T_k &= \underset{T}{\arg\min} \frac{1}{2a} \int_{\mathbb{R}^n} \|x - T(x)\|_2^2 dP_k(x) + \mathcal{F}(T\sharp P_k). \end{aligned} \tag{5}$$

We use the preceding two schemes (4) and (5) in our numerical method depending on the application.

3. Methods and algorithms

We discuss how to implement JKO scheme with our approach and its computational complexity in this section.

3.1. $\mathcal{F}(P)$ reformulation with variational formula

The main challenge in implementing the JKO scheme is to evaluate the functional $\mathcal{F}(P)$ in terms of samples from P. We achieve this goal by using a variational formulation of $\mathcal{F}$. In order to do so, we use the notion of f-divergence between the two distributions P and Q:

$$D_f(P\|Q) = \mathbb{E}_Q \left[f\left(\frac{dP}{dQ} \right) \right] \tag{6}$$

where P admits density with respect to Q (denoted as $P \ll Q$) and $f : [0, +\infty) \to \mathbb{R}$ is a convex and lower semi-continuous function. Without loss of generality, we assume $f(1) = 0$ so that D_f attains its minimum at $P = Q$.

Proposition 3.1. *(Nguyen et al., 2010) $\forall P, Q \in \mathcal{P}(\mathbb{R}^n)$ such that $P \ll Q$ and differentiable f:*

$$D_f(P\|Q) = \sup_{h \in \mathcal{C}} \mathbb{E}_P[h(X)] - \mathbb{E}_Q[f^*(h(Y))]. \tag{7}$$

where $f^*(y) = \sup_{x \in \mathbb{R}}[xy - f(x)]$ is the convex conjugate of f and $\mathcal{C}$ is all measurable functions $h : \mathbb{R}^n \to \mathbb{R}$. The supremum is attained at $h = f'(dP/dQ)$.

The variational form has the distinguishing feature that it does not involve the density of P and Q explicitly and can be approximated in terms of samples from P and Q. In general, our scheme can be applied to any f-divergence, but we focus on the functionals in Section 2.1.

With the help of the f-divergence variational formula, when $\mathcal{F}(P) = \mathcal{D}(P\|Q)$, $\mathcal{G}(P)$ or $\text{JSD}(P\|Q)$, the JKO scheme (5) can be equivalently expressed as

$$\begin{aligned} P_{k+1} &= T_k \sharp P_k, \\ T_k &= \underset{T}{\arg\min} \left\{ \frac{1}{2a} \mathbb{E}_{P_k}[\|X - T(X)\|^2] + \sup_h \mathcal{V}(T, h) \right\}. \end{aligned} \tag{8}$$

where $\mathcal{V}(T, h) = \mathbb{E}_{X \sim P_k}[\mathcal{A}_h(T(X))] - \mathbb{E}_{Z \sim \Gamma}[\mathcal{B}_h(Z)]$, Γ is a user designed distribution which is easy to sample from, and $\mathcal{A}$ and $\mathcal{B}$ are functionals whose form depends on $\mathcal{F}$. The specializations of $\mathcal{A}$ and $\mathcal{B}$ appear in Table 1.

The following lemma implies that if $\mathcal{F}(P)$ can be written as $D_f(P\|Q)$, then $\mathcal{F}(P)$ monotonically decreases along its Wasserstein gradient flow, which makes it reasonable to solve (1) by using JKO scheme. It also justifies that the gradient flow finally converges to Q.

Lemma 3.2. *Gao et al. (2019, Lemma 2.2)*
$\frac{d}{dt} \mathcal{F}(P_t) = -\mathbb{E}_{P_t}(\|\nabla f'(P_t/Q)\|^2)$.

3.1.1. KL DIVERGENCE

The KL divergence is the special instance of the f-divergence obtained by replacing f with $f_1(x) = x \log x$.

Corollary 3.3. *The variational form for $\mathcal{D}(P\|Q)$ reads*

$$\mathcal{D}(P\|Q) = 1 + \sup_h \mathbb{E}_P \left[\log \frac{h(X)\mu(X)}{Q(X)} \right] - \mathbb{E}_\mu \left[h(Z) \right],$$

where μ is a user designed distribution which is easy to sample from. The optimal function h is equal to $dP/d\mu$.

This variational formula becomes practical when we have only access to un-normalized density of Q, which is the case for the sampling problem. In practice, we choose $\mu = \mu_k$ adaptively, where μ_k is the Gaussian with the same mean and covariance as P_k. We noticed that this choice improves the numerical stability of the the algorithm.

3.1.2. GENERALIZED ENTROPY

The generalized entropy can be also represented as f-divergence. In particular, let $f_2(x) = \frac{1}{m-1}(x^m - x)$ and let Q be the uniform distribution on a set which is the superset of the support of density $P(x)$ and has volume Ω. Then

$$D_{f_2}(P\|Q) = \frac{\Omega^{m-1}}{m-1} \int P^m(x)dx - \frac{1}{m-1}. \quad (9)$$

Corollary 3.4. *The variational formulation for $\mathcal{G}(P)$ reads*

$$\mathcal{G}(P) = \frac{\sup_h \left(\mathbb{E}_P \left[\frac{m h^{m-1}(X)}{m-1} \right] - \mathbb{E}_Q \left[h^m(Z) \right] \right)}{\Omega^{m-1}}. \quad (10)$$

The optimal function h is equal to dP/dQ.

In practice, we choose $\Omega = \Omega_k$ which is the volume of a set that guarantees to contain the support of $T\sharp P_k$. In view of the connection between generalized energy and f-divergence, it is justified that the solution of Porous Media equation develops towards a uniform distribution. Especially, when $m = 2$, (9) recovers the Pearson divergence between P and the uniform distribution Q.

3.1.3. JENSEN-SHANNON DIVERGENCE

JSD$(P\|Q)$ corresponds to $D_{f_3}(P\|Q)$, where $f_3(x) = -(x+1)\log((1+x)/2) + x \log x$.

Corollary 3.5. *The variational form for JSD$(P\|Q)$ is*

$$\log 4 + \sup_h \mathbb{E}_P \left[\log(1 - h(X)) \right] + \mathbb{E}_Q \left[\log h(Z) \right]. \quad (11)$$

In particular, we apply JSD to the learn the image generative model, therefore we assume samples from Q are accessible.

Table 1: Variational formula for $\mathcal{F}(P)$

$\mathcal{F}(P)$	$\mathcal{A}_h$	$\mathcal{B}_h$	Γ
$\mathcal{D}(P\|Q)$	$\log\left(\frac{h \cdot \mu_k}{Q}\right)$	h	Gaussian dist. μ_k
$\mathcal{G}(P)$	$\frac{m}{m-1} \cdot \frac{h^{m-1}}{\Omega_k^{m-1}}$	$\frac{h^m}{\Omega_k^{m-1}}$	Uniform dist. Q_k
JSD$(P\|Q)$	$\log(1-h)$	$-\log h$	Empirical dist. Q

Algorithm 1 Primal-dual gradient flow

Input: Objective functional $\mathcal{F}(P)$, initial distribution P_0, JKO step size a, number of JKO steps K.
Initialization: Parameterized T_θ and h_λ
for $k = 1, 2, \ldots, K$ **do**
  $T_\theta \leftarrow T_{k-1}$ if $k > 1$
  for $j_1 = 1, 2, \ldots, J_1$ **do**
    Sample $X_1, \ldots, X_M \sim P_k$, $Z_1, \ldots, Z_M \sim \Gamma$.
    Maximize $\frac{1}{M}\sum_{i=1}^M \left[\mathcal{A}(T_\theta(X_i), h_\lambda) - \mathcal{B}(h_\lambda(Z_i))\right]$ over λ for J_2 steps.
    Minimize $\frac{1}{M}\sum_{i=1}^M \left[\frac{\|X_i - T_\theta(X_i)\|^2}{2a} + \mathcal{A}(T_\theta(X_i), h_\lambda)\right]$ over θ for J_3 steps.
  end for
  $T_k \leftarrow T_\theta$
end for
Output: $\{T_k\}_{k=1}^K$

3.2. Parametrization of T and h

The two optimization variables T and h in our minimax formulation (8) can be both parameterized by neural networks, denoted by T_θ and h_λ. With this neural network parametrization, we can then solve the problem by iteratively updating T_θ and h_λ. This primal-dual method to solve (1) is depicted in Algorithm 1.

In this work, we implemented two different architectures for the map T. One way is to use a residual neural network to represent T directly, and another way is to parametrize T as the gradient of a ICNN φ. The latter has been widely used in optimal transport (Makkuva et al., 2020; Fan et al., 2020; Korotin et al., 2021b). However, recently several works (Rout et al., 2021; Korotin et al., 2021a; Fan et al., 2021; Bonet et al., 2021) find poor expressiveness of ICNN architecture and also propose to replace the gradient of ICNN by a neural network. In our experiments, we find that the first parameterization gives more regular results, which aligns with the result in Bonet et al. (2021, Figure 4). However, it would be very difficult to calculate the density of pushforward distribution. Therefore, with the first parametrization, our method becomes a particle-based method, i.e. we cannot query density directly. As we discuss in Section C, when density evaluation is needed, we adopt

the ICNN since we need to compute T^{-1}.

3.3. Computational complexity

Each update k in Algorithm 1, requires $O(J_1 kMH)$ operations where J_1 is the number of iterations per each JKO step, M is the batch size, and H is the size of the network. k shows up in the bound because sampling P_k requires us to pushforward $x_0 \sim P_0$ through $k-1$ maps.

In contrast, Mokrov et al. (2021) requires $O\left(J_1\left((k+n)MH+n^3\right)\right)$ operations, which has a cubic dependence (Mokrov et al., 2021, Section 5) on dimension n because they need to query the $\log\det\nabla^2\varphi$ in each iteration. There exists fast approximation (Huang et al., 2020) of $\log\det\nabla^2\varphi$ using Hutchinson trace estimator (Hutchinson, 1989). Alvarez-Melis et al. (2021) applies this technique, thus the cubic dependence on n can be improved to quadratic dependence. Noneless, this is accompanied by an additional cost, which is the number of iterations to run conjugate gradient (CG) method. CG is guaranteed to converge exactly in n steps in this setting. If one wants to obtain $\log\det\nabla^2\varphi$ precisely, the cost is still $O(n^3)$, which is the same as calculating $\log\det\nabla^2\varphi$ directly. If one uses an error ϵ stopping condition in CG, the complexity could be improved to $\sqrt{\kappa}\log(2/\epsilon)n^2$ (Shewchuk et al., 1994), where κ is the upper bound of condition number of $\nabla^2\varphi$, but this would sacrifice on the accuracy. Given the similar neural network size, our method has the advantage of independence on the dimension for the training time.

Other than training time, the complexity for evaluating the density has unavoidable dependence on n due to the standard density evaluation process (see Section C).

4. Theoretical results

We introduce approximate f-divergence notation and analyze its properties in this section.

4.1. Approximate f-divergence

Given the results in Proposition 3.1, now we consider a restriction of the optimization domain $\mathcal{C}$ to a class of functions $\mathcal{H}$, e.g parametrized by neural networks, and define the new functional

$$D_f^{\mathcal{H}}(P\|Q) = \sup_{h\in\mathcal{H}}\left\{\int h dP - \int f^*(h) dQ\right\}.$$

This functional forms a surrogate for the exact f-divergence. It is straightforward to see that the new function is always smaller than the exact f-divergence, i.e. $D_f^{\mathcal{H}}(P\|Q) \le D_f(P\|Q)$ where the inequality is achieved when $f'(\frac{dP}{dQ})$ belongs to $\mathcal{H}$. In the following lemma, we establish some important theoretical properties of the approximate f-divergence $D_f^{\mathcal{H}}(P\|Q)$. In order to do so, we introduce the integral probability metric

$$d_{\mathcal{H}}(P,Q) = \sup_{h\in\mathcal{H}}\frac{1}{\|h\|_{2,Q}}\left\{\int h dP - \int h dQ\right\},$$

where $\|h\|_{2,Q}^2 = \int h^2 dQ$.

Proposition 4.1. *The approximate f-divergence $D_f^{\mathcal{H}}(P\|Q)$ satisfies the following properties:*
1. (positivity) If $\mathcal{H}$ contains all constant functions, then

$$D_f^{\mathcal{H}}(P\|Q) \ge 0, \quad \forall P, Q$$

2. (moment-matching) If for all $h \in \mathcal{H}$, $c + \lambda h \in \mathcal{H}$ for $c, \lambda \in \mathbb{R}$, then

$$D_f^{\mathcal{H}}(P\|Q) = 0 \;\Leftrightarrow\; \int h dP = \int h dQ,\ \forall h \in \mathcal{H}$$

3. (embedding inequalities) Additionally, if f is strongly convex with constant α, and smooth with constant L, then,

$$\frac{\alpha}{2} d_{\mathcal{H}}(P,Q)^2 \le D_f^{\mathcal{H}}(P\|Q) \le \frac{L}{2} d_{\mathcal{H}}(P,Q)^2.$$

The lemma has important implications. Part (1) establishes the condition under which the approximate f-divergence is always positive. Part (2) identifies necessary and sufficient conditions under which the approximate divergence is zero for two given probability distributions P and Q. In particular, the divergence is zero iff the moments of P and Q are equal for all functions in the function class $\mathcal{H}$. Finally, part (3) provides lower-bound and upper-bound for the approximate f-divergence in terms of an integral probability metric defined on the function class $\mathcal{H}$, implying that the two measures are equivalent. For example, a sequence $D_f^{\mathcal{H}}(P_d\|Q_d) \to 0$ as $d \to \infty$ iff $d_{\mathcal{H}}(P_d, Q_d) \to 0$ as $d \to \infty$. Or if we are able to minimize the approximate f-divergence $D_f^{\mathcal{H}}(P\|Q)$ with optimization gap ϵ, then the error in the moments of P and Q for functions in $\mathcal{H}$ is of order $O(\sqrt{\epsilon})$. These results inform us that the proposed objective function of minimizing $D_f^{\mathcal{H}}(P\|Q)$ is meaningful and has geometrical significance.

Remark 4.2. The assumption that $c+\lambda h \in \mathcal{H}$ for all $h \in \mathcal{H}$ and $c, \lambda \in \mathbb{R}$ holds for any neural network with linear activation function at the last layer. The assumption that f is strongly convex and smooth may not hold for a typical f such as $f(x) = x\log(x)$ over $(0,\infty)$. However, It holds when the domain is restricted, which is true when either the samples are bounded or h is bounded for all $h \in \mathcal{H}$.

4.2. Computational boundness

It is also possible to obtain lower-bound for $D_f^{\mathcal{H}}(P\|Q)$ in terms of the exact f-divergence $D_f(P\|Q)$ when the class $\mathcal{H}$ is rich enough.

Proposition 4.3. *If f is α-strongly convex and the class of functions is able to approximate any function $h \in \mathcal{C}$ with $\tilde{h} \in \mathcal{H}$ such that $\|\tilde{h} - h\|_{2,Q} \leq \epsilon$, then*

$$D_f^{\mathcal{H}}(P\|Q) \geq D_f(P\|Q) - \frac{\epsilon^2}{2\alpha}, \quad \forall P, Q.$$

However, it is not easy to verify the assumption and the value of ϵ for any given function class.

Unlike the exact form of the f-divergence, the variational formulation is well-defined for empirical distributions when the function class $\mathcal{H}$ is restricted and admits a finite Rademacher complexity.

Proposition 4.4. *Let $P_N = \frac{1}{N}\sum_{i=1}^N \delta_{X_i}$, $Q_M = \frac{1}{M}\sum_{i=1}^M \delta_{Y_i}$, where $\{X_i\}_{i=1}^N, \{Y_i\}_{i=1}^M$ are i.i.d samples from P and Q respectively. Then, it follows that*

$$\begin{aligned}&\mathbb{E}[|D_f^{\mathcal{H}}(P\|Q) - D_f^{\mathcal{H}}(P_N\|Q)|]\\ \leq& 2\mathcal{R}_N(\mathcal{H}, P) + 2\mathcal{R}_M(f^* \circ \mathcal{H}, Q),\end{aligned}$$

where the expectation is over the samples and $\mathcal{R}_N(\mathcal{H}, P)$ denotes the Rademacher complexity of the function class $\mathcal{H}$ with respect to P for sample size N.

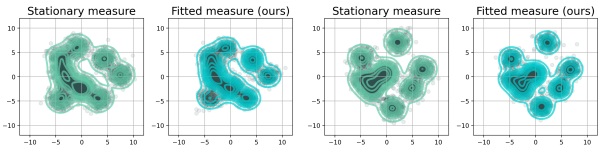
We leave the task of evaluating the Rademacher complexity for different function classes employed in this paper for future work. The preliminary results presented in this section are obtained using elementary tools in convex analysis, which is a feature of the variational approach, and serve as the starting point for further research in this direction.

5. Numerical examples

In this section, we present several numerical examples to illustrate our algorithm. We mainly compare with the JKO-ICNN-d (Mokrov et al., 2021), JKO-ICNN-a (Alvarez-Melis et al., 2021). The difference between JKO-ICNN-d and JKO-ICNN-a is that the former computes the $\log\det(\nabla^2\varphi)$ **d**irectly and the latter adopts fast **a**pproximation. We use the default hyper-parameters in the authors' implementation.

5.1. Sampling from Gaussian Mixture Model

We first consider the sampling problem to sample from a target distribution Q. Note that Q doesn't have to be normalized. To this end, we consider the Wasserstein gradient flow with objective function $\mathcal{F}(P) = \mathcal{D}(P\|Q)$, that is, the KL divergence between distributions P and Q. When this objective is minimized, $P \propto Q$. In our experiments, we consider the Gaussian mixture model (GMM) with 10 equal-weighted spherical Gaussian components. The mean of Gaussian components are randomly uniformly sampled inside a cube. The step size is set to be $a = 0.1$ and the initial measure is a spherical Gaussian $\mathcal{N}(0, 16I_n)$. In Figure 1, we show our generated samples are in concordance with the target measure.



(a) Dimension $n = 64$ (b) Dimension $n = 128$

Figure 1: Comparison between the target GMM and fitted measure of generated samples by our method. Samples are projected onto 2D plane by performing PCA.

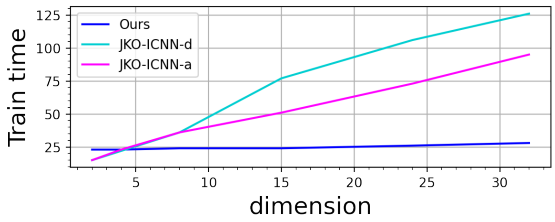
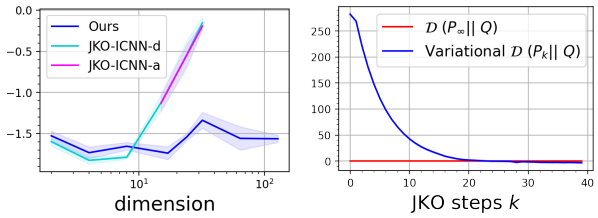


Figure 2: Averaged training time (in minutes) of 40 JKO steps for sampling from GMM.



(a) $\log_{10}$KSD (b) Objective functional

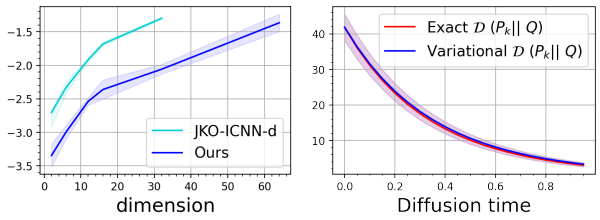
Figure 3: (a) We perform experiments in $n = 2, 4, 8, 15, 24, 32$ for all methods and additionally $n = 64, 128$ for our method. With the constraint of similar training time, our method gives smaller error in high dimension. (b) With the variational formula, we use only samples to estimate the objective functional $\mathcal{D}(P_k\|Q)$ in dimension $n = 64$. It converges to the ideal objective minimum $\mathcal{D}(P_\infty\|Q) = 0$.

In Figure 2, we plot the averaged training time of 5 runs for all compared methods. Note that we fix the number of conjugate descent steps to be at most 10 when approximating $\log\det\nabla^2\varphi$ in JKO-ICNN-a. That's why JKO-ICNN-d and JKO-ICNN-a have quite similar training time when $n < 10$.

To investigate the performance under the constraint of similar training time, we perform 40 JKO steps with our method and the same for JKO-ICNN methods except for $n \geq 15$, where we only let them run for $20, 15, 12$ JKO steps for $n = 15, 24, 32$ respectively. In doing so, one can verify the

training time of our method and JKO-ICNN is roughly consistent. We only report the accuracy results of JKO-ICNN-d for $n < 10$ in Figure 3 since it's prone to give higher accuracy than JKO-ICNN-a considering nearly the same training time in low dimension. We select Kernalized Stein Divergence (KSD) (Liu et al., 2016) as the error criteria because it only requires samples to estimate the divergence, which is useful in the sampling task.

5.2. Ornstein-Uhlenbeck Process



(a) $\log_{10}$SymKL (b) Objective functional

Figure 4: (a): We repeat the experiments for 15 times and compare the SymKL (Mokrov et al., 2021) between estimated distribution and the ground truth at $k = 18$ in OU process. (b): We show the comparison between our estimated $\mathcal{D}(P_k\|Q)$ and the ground truth in dimension $n = 64$. They align with each other pretty well.

We study the performance of our method in modeling the Ornstein-Uhlenbeck Process as dimension grows. The gradient flow is affiliated with the free energy (2), where $Q = e^{(x-b)^{\mathrm{T}}A(x-b)/2}$ with a positive definite matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ and $b \in \mathbb{R}^n$. Given an initial Gaussian distribution $\mathcal{N}(0, I_n)$, the gradient flow at each time t is a Gaussian distribution P_t with mean vector $\mu_t = (I_n - e^{-At})b$ and covariance $\Sigma_t = A^{-1}(I_n - e^{-2At}) + e^{-2At}$ (Vatiwutipong & Phewchean, 2019). We choose JKO step size $a = 0.05$. We only present JKO-ICNN-d accuracy results because JKO-ICNN-a has the similar or slightly worse performance.

There could be several reasons why we have better performance. 1) The proposed distribution μ is Gaussian, which is consistent with P_t for any t. This is beneficial for the inner maximization to find a precise h. 2) Parameterizing T as a neural network instead of gradient of ICNN is handier for optimization in this toy example.

We also compare the training time per every two JKO steps with JKO-ICNN method. The computation time for JKO-ICNN-d is around $25s$ when $n = 2$ and increases to $105s$ when $n = 32$. JKO-ICNN-a has slightly better scalability, which increases from $25s$ to $95s$. Our method's training time remains at $22s \pm 5s$ for all the dimensions $n = 2 \sim 32$. This is due to the fact that we fix the neural network size for both methods and our method's computation complexity does not depend on the dimension.

Table 2: Bayesian logistic regression accuracy and log-likelihood results.

Dataset	Accuracy		Log-Likelihood	
	Ours	JKO-ICNN	Ours	JKO-ICNN
covtype	0.753	0.75	-0.528	-0.515
splice	0.84	0.845	-0.38	-0.36
waveform	0.785	0.78	-0.455	-0.485
twonorm	0.982	0.98	-0.056	-0.059
ringnorm	0.73	0.74	-0.5	-0.5
german	0.67	0.67	-0.59	-0.6
image	0.866	0.82	-0.394	-0.43
diabetis	0.786	0.775	-0.45	-0.45
banana	0.55	0.55	-0.69	-0.69

5.3. Bayesian Logistic Regression

To evaluate our method on a real-world datast, we consider the bayesian logistic regression task with the same setting in Gershman et al. (2012). Given a dataset $\mathcal{L} = \{l_1, \ldots, l_S\}$, a model with parameters $x \in \mathbb{R}^n$ and the prior distribution $p_0(x)$, our target is to sample from the posterior distribution

$$p(x|\mathcal{L}) \propto p_0(x)p(\mathcal{L}|x) = p_0(x) \cdot \prod_{s=1}^{S} p(l_s|x).$$

To this end, we let the target distribution $Q(x) = p_0(x)p(\mathcal{L}|x)$ and choose $\mathcal{F}(P)$ equal to $\mathcal{D}(P\|Q)$. The parameter x takes the form of $[\omega, \log\alpha]$, where $\omega \in \mathbb{R}^{n-1}$ is the regression weights with the prior $p_0(\omega|\alpha) = \mathcal{N}(\omega, \alpha^{-1})$. α is a scalar with the prior $p_0(\alpha) = \mathrm{Gamma}(\alpha|1, 0.01)$. We test on 8 relatively small datasets ($S \leq 7400$) from Mika et al. (1999) and one large Covertype dataset[1] ($S = 0.58$M). The dataset is randomly split into training dataset and test dataset according to the ratio 4:1. The number of features scales from 2 to 60. From Table 2, we can tell that our method achieves a comparable performance as the other. The results of JKO-ICNN-d are adapted from Mokrov et al. (2021, Table 1). We present the datasets properties and comparison with another popular sampling method SVGD (Liu & Wang, 2016) in Table 5 in the Appendix.

5.4. Porous media equation

We next consider the porous media equation with only diffusion: $\partial_t P = \Delta P^m$. This is the Wasserstein gradient flow associated with the energy function $\mathcal{F}(P) = \mathcal{G}(P)$. A representative closed-form solution of the porous media equation is the Barenblatt profile (GI, 1952; Vázquez, 2007)

$$P(t,x) = (t+t_0)^{-\alpha}\left(C - \beta\|x - x_0\|^2 (t+t_0)^{\frac{-2\alpha}{n}}\right)_+^{\frac{1}{m-1}},$$

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

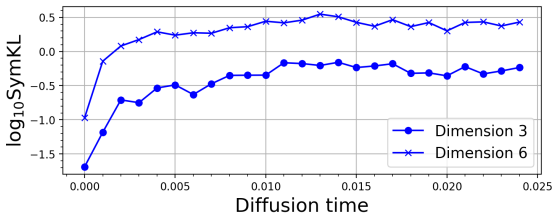
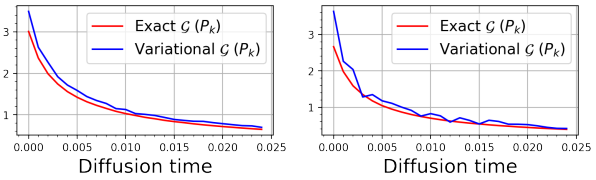


Figure 5: SymKL with respect to the Barenblatt profile ground truth in 50 JKO steps.



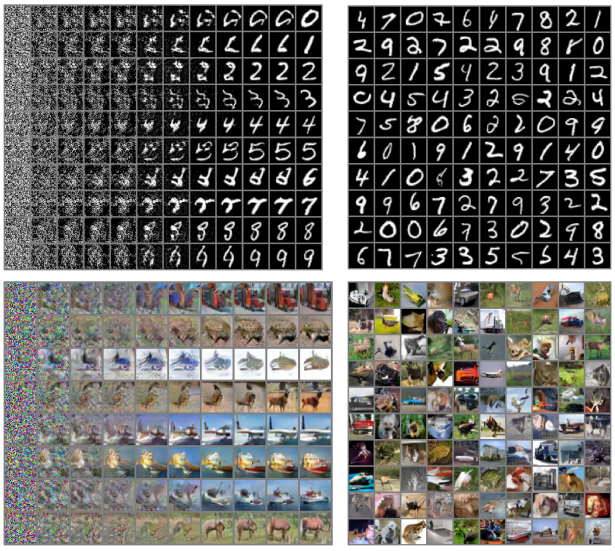
(a) Dimension $n = 3$ (b) Dimension $n = 6$

Figure 6: We use variational formula to calculate the objective functional $\mathcal{G}(P)$ with samples and compare it with ground truth.

where $\alpha = \frac{n}{n(m-1)+2}$, $\beta = \frac{(m-1)\alpha}{2mn}$, $t_0 > 0$ is the starting time, and $C > 0$ is a free parameter. In the experiments, we set $m = 2$, the stepsize for the JKO scheme to be $a = 0.0005$ and the initial time to be $t_0 = 0.001$. We parametrize the transport map T as the gradient of an ICNN and thus we can evaluate the density following Section C. From Figure 5, we observe that our method can give stable simulation results, where the error is controlled in a small region as diffusion time increases.

5.5. Gradient flow on images

We apply our scheme on real image datasets, where only samples from Q are accessible. With the variational formula (11), Algorithm 1 can be adapted to model gradient flow in image space. Specifically, we choose $\mathcal{F}(P)$ to be $\mathrm{JSD}(P\|Q)$ and $P_0 = \mathcal{N}(0, I_n)$. We name the resulted model *JKO-Flow*. Note JKO-Flow model specializes to GAN (Goodfellow et al., 2014) when $a \to \infty$ and $K = 1$. Thanks to the additional Wassersteian loss regularization, JKO-Flow enjoys stable training and suffer less from mode collapsing empirically. We evaluate JKO-Flow on popular MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009) datasets. Figure 7 shows samples and their trajectories starting from P_0 to P_K and demonstrates JKO-Flow can approximate Wassersteian gradient flow in image space empirically. To further quantify the performance of JKO-Flow, we measure discrepancy between P_K and real distribution with the popular sample metric, Fenchel Inception Distance (Heusel et al., 2017) in Table 3. We include more comparison and experiments details in Section G.



(a) Trajectory (b) Uncurated samples

Figure 7: With Wasserstein gradient flow scheme, we visualize (a): trajectories of the generated samples from JKO-Flow and (b): 100 uncurated samples from P_K.

Table 3: Results of Gradient flow (GF) based methods and GAN methods on unconditional CIFAR10 dataset.

	Method	FID score ↓
GF	VGrow (Gao et al., 2019)	28.8
	JKO-Flow	**23.1**
GANs	WGAN-GP (Arbel et al., 2018)	31.1
	SN-GAN (Miyato et al., 2018)	**21.7**

6. Conclusion

In this paper, we presented a numerical procedure to implement the Wasserstein gradient flow for objective functions in the form of f-divergence. Our procedure is based on applying the JKO scheme on a variational formulation of the f-divergence. Each step involves solving a min-max stochastic optimization problem for a transport map and a dual function that are parameterized by neural networks. We demonstrated the scalability of our approach to high-dimensional problems through numerical experiments on Gaussian mixture models and real datasets including MNIST and CIFAR10. We also provided preliminary theoretical results regarding the variational objective function. The results show that minimizing the variational objective is meaningful and serve as starting point for future research. Our method can also be adapted to Crank-Nicolson type scheme, which enjoys a faster convergence (Carrillo et al., 2021) in step size a than the classical JKO scheme (see Section B). Another possible direction for future research is to extend the variational formulation beyond f-divergence.

References

Adams, S., Dirr, N., Peletier, M. A., and Zimmer, J. From a large-deviations principle to the Wasserstein gradient flow: a new micro-macro passage. *Communications in Mathematical Physics*, 307(3):791–815, 2011.

Alvarez-Melis, D., Schiff, Y., and Mroueh, Y. Optimizing functionals on the space of probabilities with input convex neural networks. *arXiv preprint arXiv:2106.00774*, 2021.

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *International Conference on Machine Learning*, pp. 146–155. PMLR, 2017.

An, D., Guo, Y., Lei, N., Luo, Z., Yau, S.-T., and Gu, X. Ae-ot: a new generative model based on extended semi-discrete optimal transport. *ICLR 2020*, 2019.

An, D., Guo, Y., Zhang, M., Qi, X., Lei, N., and Gu, X. Ae-ot-gan: Training gans from data specific latent distribution. In *European Conference on Computer Vision*, pp. 548–564. Springer, 2020.

Arbel, M., Sutherland, D. J., Bińkowski, M. a., and Gretton, A. On gradient regularizers for mmd gans. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/07f75d9144912970de5a09f5a305e10c-Paper.pdf.

Benamou, J.-D., Carlier, G., Mérigot, Q., and Oudet, E. Discretization of functionals involving the monge–ampère operator. *Numerische mathematik*, 134(3):611–636, 2016.

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Bonet, C., Courty, N., Septier, F., and Drumetz, L. Sliced-wasserstein gradient flows. *arXiv preprint arXiv:2110.10972*, 2021.

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

Bunne, C., Meng-Papaxanthos, L., Krause, A., and Cuturi, M. Jkonet: Proximal optimal transport modeling of population dynamics. *arXiv preprint arXiv:2106.06345*, 2021.

Carlier, G., Duval, V., Peyré, G., and Schmitzer, B. Convergence of entropic schemes for optimal transport and gradient flows. *SIAM Journal on Mathematical Analysis*, 49(2):1385–1418, 2017.

Carrillo, J. A., Craig, K., and Patacchini, F. S. A blob method for diffusion. *Calculus of Variations and Partial Differential Equations*, 58(2):1–53, 2019a.

Carrillo, J. A., Hittmeir, S., Volzone, B., and Yao, Y. Nonlinear aggregation-diffusion equations: radial symmetry and long time asymptotics. *Inventiones mathematicae*, 218(3):889–977, 2019b.

Carrillo, J. A., Craig, K., Wang, L., and Wei, C. Primal dual methods for Wasserstein gradient flows. *Foundations of Computational Mathematics*, pp. 1–55, 2021.

Eckhardt, R., Ulam, S., and Von Neumann, J. the monte carlo method. *Los Alamos Science*, 15:131, 1987.

Falcon, W. and Cho, K. A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*, 2020.

Fan, J., Taghvaei, A., and Chen, Y. Scalable computations of Wasserstein barycenter via input convex neural networks. *arXiv preprint arXiv:2007.04462*, 2020.

Fan, J., Liu, S., Ma, S., Chen, Y., and Zhou, H. Scalable computation of monge maps with general costs. *arXiv preprint arXiv:2106.03812*, 2021.

Frogner, C. and Poggio, T. Approximate inference with Wasserstein gradient flows. In *International Conference on Artificial Intelligence and Statistics*, pp. 2581–2590. PMLR, 2020.

Gao, Y., Jiao, Y., Wang, Y., Wang, Y., Yang, C., and Zhang, S. Deep generative learning via variational gradient flow. In *International Conference on Machine Learning*, pp. 2093–2101. PMLR, 2019.

Gershman, S., Hoffman, M., and Blei, D. Nonparametric variational inference. *arXiv preprint arXiv:1206.4665*, 2012.

GI, B. On some unsteady motions of a liquid and gas in a porous medium. *Prikl. Mat. Mekh.*, 16:67–78, 1952.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Huang, C.-W., Chen, R. T., Tsirigotis, C., and Courville, A. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *arXiv preprint arXiv:2012.05942*, 2020.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Korotin, A., Egiazarian, V., Asadulaev, A., Safin, A., and Burnaev, E. Wasserstein-2 generative networks. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=bEoxzW_EXsa.

Korotin, A., Li, L., Solomon, J., and Burnaev, E. Continuous wasserstein-2 barycenter estimation without minimax optimization. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=3tFAs5E-Pe.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, W., Lu, J., and Wang, L. Fisher information regularization schemes for Wasserstein gradient flows. *Journal of Computational Physics*, 416:109449, 2020.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf.

Liu, Q., Lee, J., and Jordan, M. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pp. 276–284. PMLR, 2016.

MacKay, D. J. and Mac Kay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Makkuva, A., Taghvaei, A., Oh, S., and Lee, J. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pp. 6672–6681. PMLR, 2020.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K.-R. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pp. 41–48. Ieee, 1999.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Mokrov, P., Korotin, A., Li, L., Genevay, A., Solomon, J., and Burnaev, E. Large-scale wasserstein gradient flows. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=nlLjIuHsMHp.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 271–279, 2016.

Otto, F. The geometry of dissipative evolution equations: the porous medium equation. 2001.

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

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Rout, L., Korotin, A., and Burnaev, E. Generative modeling with optimal transport maps. *arXiv preprint arXiv:2110.02999*, 2021.

Salim, A., Korba, A., and Luise, G. The Wasserstein proximal gradient algorithm. *arXiv preprint arXiv:2002.03035*, 2020.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Santambrogio, F. Euclidean, metric, and wasserstein gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017.

Seguy, V., Damodaran, B. B., Flamary, R., Courty, N., Rolet, A., and Blondel, M. Large-scale optimal transport and mapping estimation. *arXiv preprint arXiv:1711.02283*, 2017.

Shewchuk, J. R. et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

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

Vázquez, J. L. *The porous medium equation: mathematical theory*. Oxford University Press on Demand, 2007.

Villani, C. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

Wan, N., Li, D., and Hovakimyan, N. f-divergence variational inference. *Advances in Neural Information Processing Systems*, 33, 2020.

Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

Wellner, J. A. Empirical processes: Theory and applications. *Notes for a course given at Delft University of Technology*, 2005.

Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

Yang, Z., Zhang, Y., Chen, Y., and Wang, Z. Variational transport: A convergent particle-based algorithm for distributional optimization. *arXiv preprint arXiv:2012.11554*, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

A. Proofs

A.1. Proof of variational formulas in Section 3.1

A.1.1. KL DIVERGENCE

The KL divergence is the special instance of the f-divergence obtained by replacing f with $f_1(x) = x \log x$ in (6)

$$D_{f_1}(P\|Q) = \mathbb{E}_Q\left[\frac{P}{Q}\log\frac{P}{Q}\right] = \mathbb{E}_P\left[\log\frac{P}{Q}\right],$$

which, according to (7), admits the variational formulation

$$D_{f_1}(P\|Q) = 1 + \sup_h \mathbb{E}_P\left[h(X)\right] - \mathbb{E}_Q\left[e^{h(Z)}\right] \tag{12}$$

where the convex conjugate $f_1^*(y) = e^{y-1}$ is used.

The variational formulation can be approximated in terms of samples from P and Q. For the case where we have only access to un-normalized density of Q, which is the case for the sampling problem, we use the following change of variable: $h \to \log(h) + \log(\mu) - \log(Q)$ where μ is a user designed distribution which is easy to sample from. Under such a change of variable, the variational formulation reads

$$D_{f_1}(P\|Q) = 1 + \sup_h \mathbb{E}_P\left[\log h(X) + \log\frac{\mu(X)}{Q(X)}\right] - \mathbb{E}_\mu\left[h(Z)\right].$$

Note that the optimal function h is equal to the ratio between the densities of $T\sharp P_k$ and μ. Using this variational form in the JKO scheme (5) yields $P_{k+1} = T_k\sharp P_k$ and

$$T_k = \arg\min_T \max_h \mathbb{E}_{P_k}[\|X - T(X)\|^2/2a + \log h(T(X)) + \log\frac{\mu(T(X))}{Q(T(X))}] - \mathbb{E}_\mu\left[h(Z)\right]. \tag{13}$$

Remark A.1. The Donsker-Varadhan formula

$$\mathcal{D}(P\|Q) = \sup_h \mathbb{E}_P\left[h(X)\right] - \log\mathbb{E}_Q\left[e^{h(Z)}\right]$$

is another variational representation of KL divergence and it's a stronger than (12) because it's a upper bound of (12) for any fixed h. However, we cannot get an unbiased estimation of the objective using samples.

A.1.2. GENERALIZED ENTROPY

The generalized entropy can be also represented as f-divergence. In particular, let $f_2(x) = \frac{1}{m-1}(x^m - x)$ and let Q be the uniform distribution on a set which is the superset of the support of density $P(x)$ and has volume Ω. Then

$$D_{f_2}(P\|Q) = \frac{\Omega^{m-1}}{m-1}\int P^m(x)dx - \frac{1}{m-1}.$$

As a result, the generalized entropy can be expressed in terms of f-divergence according to

$$\mathcal{G}(P) = \frac{1}{m-1}\int P^m(x)dx = \frac{1}{\Omega^{m-1}}D_{f_2}(P\|Q) + \frac{1}{\Omega^{m-1}(m-1)}.$$

Upon using the variational representation of the f-divergence with

$$f_2^*(y) = \left(\frac{(m-1)y+1}{m}\right)^{\frac{m}{m-1}},$$

the generalized entropy admits the following variational formulation

$$\mathcal{G}(P) = \sup_h \frac{1}{\Omega^{m-1}}\left(\mathbb{E}_P[h(X)] - \mathbb{E}_Q\left[\left(\frac{(m-1)h(Z)+1}{m}\right)^{\frac{m}{m-1}}\right]\right) + \frac{1}{\Omega^{m-1}(m-1)}.$$

In practice, we find it numerically useful to let $h = \frac{1}{m-1}\left[m\left(\hat{h}\right)^{m-1} - 1\right]$ so that

$$\mathcal{G}(P) = \frac{1}{\Omega^{m-1}} \sup_{\hat{h}} \left(\mathbb{E}_{P_k}\left[\frac{m}{m-1}\hat{h}^{m-1}(X)\right] - \mathbb{E}_Q\left[\hat{h}^m(Z)\right]\right).$$

With such a change of variable, the optimal function $\hat{h} = T\sharp P_k/Q$. Using this in the JKO scheme yields $P_{k+1} = T_k\sharp P_k$, and

$$T_k = \arg\min_T \max_h \frac{1}{2a}\mathbb{E}_{P_k}\|X - T(X)\|^2 + \frac{1}{\Omega^{m-1}}\left(\mathbb{E}_{P_k}\left[\frac{m}{m-1}h^{m-1}(X)\right] - \mathbb{E}_Q\left[h^m(Z)\right]\right).$$

A.1.3. JENSEN-SHANNON DIVERGENCE

Jensen-Shannon divergence has been widely studied in GAN literature (Nowozin et al., 2016). The variational formula follows that $f(x) = -(x+1)\log((1+x)/2) + x\log x$ and $f^*(y) = -\log(2-\exp(y))$. Plugging in the variational formula in the JKO scheme gives

$$T_k = \arg\min_T \max_h \frac{1}{2a}\mathbb{E}_{P_k}\|X - T(X)\|^2 + \mathbb{E}_{P_k}\left[\log(1-h(X))\right] + \mathbb{E}_Q\left[\log h(Z)\right].$$

A.2. Proof of Propostion 4.1

We present the proof of Propostion 4.1. Let us define $J(h) := \int h dP - \int f^*(h) dQ$.

1. The proof follows from

$$D_f^{\mathcal{H}}(P,Q) = \sup_{h\in\mathcal{H}} J(h) \geq \sup_{c\in\mathbb{R}} J(c) = \sup_{c\in\mathbb{R}}\{c - f^*(c)\} = f(1) = 0$$

   where the last identity follows from the assumption that $f(1) = 0$.

2. The direction ($\Leftarrow$) follows because

$$J(h) \leq \int h dP - \int h dQ = 0, \quad \forall h \in \mathcal{H}$$

   where $f^*(y) = \sup_x\{xy - f(x)\} \geq y1 - f(1) = y$ is used. As a result, $D_f^{\mathcal{H}}(p\|Q) = \sup_{h\in\mathcal{H}} J(h) \leq 0$. Using part (1), this is only possible when $D_f^{\mathcal{H}}(P\|Q) = 0$.

   To show the other direction ($\Rightarrow$), for all $h \in \mathcal{H}$, define $g(\lambda) := J(f'(1) + \lambda h)$ where $\lambda \in \mathbb{R}$. The function $g(\lambda)$ attains its maximum at $\lambda = 0$ because $g(\lambda) = J(f'(1) + \lambda h) \leq \sup_{h\in\mathcal{H}} J(h) = D_f^{\mathcal{H}}(P\|Q) = 0$ and $g(0) = J(f'(1)) = f'(1) - f^*(f'(1)) = f(1) = 0$ by Fenchel identity. Therefore, the first-order optimality condition $g'(0) = 0$ must hold. The result follows because

$$g'(0) = \int h dP - \int h f^{*\prime}(f'(1)) dQ = \int h dP - \int h dQ$$

3. Let us define $g_h(\lambda) := J(f'(1) + \frac{\lambda h}{\|h\|_{2,Q}})$. The first and the second derivatives of $g_h(\lambda)$ with respect to λ are:

$$g_h'(\lambda) = \int \frac{h}{\|h\|} dP - \int \frac{h}{\|h\|} f^{*\prime}(f'(1) + \frac{\lambda h}{\|h\|}) dQ$$
$$g_h''(\lambda) = -\int \frac{h^2}{\|h\|^2} f^{*\prime\prime}(f'(1) + \frac{\lambda h}{\|h\|}) dQ$$

   By assumption on f, the convex conjugate f^* is strongly convex with constant $\frac{1}{L}$ and smooth with constant $\frac{1}{\alpha}$. Therefore, $\frac{1}{L} \leq f^{*\prime\prime}(y) \leq \frac{1}{\alpha}$. As a result, $\frac{1}{L} \leq -g_h''(\lambda) \leq \frac{1}{\alpha}$ where we used $\|h\|^2 = \int h^2 dQ$. Therefore, $g_h(\lambda)$ is strongly concave and smooth and satisfies the inequalities:

$$\frac{\alpha}{2} g_h'(0)^2 \leq \sup_\lambda g_h(\lambda) - g_h(0) \leq \frac{L}{2} g_h'(0)^2$$

Upon using $g_h(0) = J(f'(1)) = 0$ and taking the sup over $h \in \mathcal{H}$ of all sides,

$$\frac{\alpha}{2} \sup_{h\in\mathcal{H}} g_h'(0)^2 \le \sup_{h\in\mathcal{H}} \sup_{\lambda} g_h(\lambda) \le \frac{L}{2} \sup_{h\in\mathcal{H}} g_h'(0)^2.$$

By the assumption that for all $h \in \mathcal{H}$, $c + \lambda h \in \mathcal{H}$ for $c, \lambda \in \mathbb{R}$,

$$\sup_{h\in\mathcal{H}} \sup_{\lambda} g_h(\lambda) = \sup_{h\in\mathcal{H}} J(h) = D_f^{\mathcal{H}}(P\|Q).$$

The result follows by noting that $\sup_{h\in\mathcal{H}} g_h'(0) = d_{\mathcal{H}}(P, Q)$.

A.3. Proof of Proposition 4.3

Proof. For a given P and Q, let $h_0 = f'(\frac{dP}{dQ})$ and $\tilde{h} \in \mathcal{H}$ be such that $\|\tilde{h} - h_0\|_{2,Q} \le \epsilon$. Similar to the proof of Proposition 4.1, define $J(h) = \int h dP - \int f^*(h) dQ$. Then,

$$D_f^{\mathcal{H}}(P\|Q) = \sup_{h\in\mathcal{H}} J(h) \ge J(\tilde{h}) = J(\tilde{h}) - J(h_0) + J(h_0) = J(\tilde{h}) - J(h_0) + D_f(P\|Q)$$

where $J(h_0) = D_f(P\|Q)$ is used in the last step. The proof follows by showing that $J(\tilde{h}) - J(h_0) \ge -\frac{1}{2\alpha}\|\tilde{h} - h_0\|_{2,Q}^2$. In order to show this, note that f^* is $\frac{1}{\alpha}$ smooth because f is α strongly convex. Then,

$$f^*(\tilde{h}(x)) - f^*(h_0(x)) \le f^{*\prime}(h_0(x))(\tilde{h}(x) - h_0(x)) + \frac{1}{2\alpha}|\tilde{h}(x) - h_0(x)|^2, \quad \forall x.$$

Taking the expectation over Q and adding $\int h_0 dP - \int \tilde{h} dP$ yields,

$$J(h_0) - J(\tilde{h}) \le \int f^{*\prime}(h_0)(\tilde{h} - h_0) dQ + \int (h_0 - \tilde{h}) dP + \frac{1}{2\alpha}\|\tilde{h} - h\|_{2,Q}^2.$$

Then, the proof follows from $f^{*\prime}(h_0) = f^{*\prime}(f'(\frac{dP}{dQ})) = \frac{dP}{dQ}$ to cancel the first two terms.

□

A.4. Proof of Proposition 4.4

Proof. We first introduce the following notations

$$\begin{aligned}
J(h) &= \int h dP - \int f^*(h) dQ \\
J_{M,N}(h) &= \int h dP_N - \int f^*(h) dQ_M, \\
G_P(h) &= \int h dP - \int h dP_N, \\
G_Q(h) &= \int f^*(h) dQ - \int f^*(h) dQ_M.
\end{aligned}$$

Assume the $\sup_{h\in\mathcal{H}} J(h)$ is attained at $h = \bar{h}$ and $\sup_{h\in\mathcal{H}} J_{M,N}(h)$ is attained at $h = h_{M,N}$.

$$\sup_{h\in\mathcal{H}} J_{M,N}(h) - \sup_{h\in\mathcal{H}} J(h) = J_{M,N}(h_{M,N}) - \sup_{h\in\mathcal{H}} J(h) \le J_N(h_{M,N}) - J(h_{M,N}) \le \sup_{h\in\mathcal{H}}\{|G_P(h)|\} + \sup_{h\in\mathcal{H}}\{|G_Q(h)|\}.$$

Similarly

$$\sup_{h\in\mathcal{H}} J(h) - \sup_{h\in\mathcal{H}} J_{M,N}(h) = J(\bar{h}) - \sup_{h\in\mathcal{H}} J_{M,N}(h) \le J_{M,N}(\bar{h}) - J(\bar{h}) \le \sup_{h\in\mathcal{H}}\{|G_P(h)|\} + \sup_{h\in\mathcal{H}}\{|G_Q(h)|\}.$$

Therefore,

$$|D_f^{\mathcal{H}}(P_N\|Q_M) - D_f^{\mathcal{H}}(P\|Q)| = |\sup_{h\in\mathcal{H}} J_{M,N}(h) - \sup_{h\in\mathcal{H}} J(h)| \le \sup_{h\in\mathcal{H}}\{|G_P(h)|\} + \sup_{h\in\mathcal{H}}\{|G_Q(h)|\}.$$

The result follows by taking the expectation and the symmetrization inequality (Wellner, 2005, Lemma 5.1) to the last two terms

$$\mathbb{E}\sup_{h\in\mathcal{H}}\{|G_P(h)|\}+\mathbb{E}\sup_{h\in\mathcal{H}}\{|G_Q(h)|\}\leq 2\mathcal{R}_N(\mathcal{H},P)+2\mathcal{R}_M(f^*\circ\mathcal{H},Q).$$

□

It's not difficult to prove the following corollary following the same logic.

Corollary A.2. *Let $P_N=\frac{1}{N}\sum_{i=1}^N\delta_{X_i}$, where $\{X_i\}_{i=1}^N$ are i.i.d samples from P. Then, it follows that*

$$\mathbb{E}[|D_f^{\mathcal{H}}(P\|Q)-D_f^{\mathcal{H}}(P_N\|Q)|]\leq 2\mathcal{R}_N(\mathcal{H},P),$$

where the expectation is over the samples and $\mathcal{R}_N(\mathcal{H},P)$ denotes the Rademacher complexity of the function class $\mathcal{H}$ with respect to P for sample size N.

B. Extension to Crank-Nicolson scheme

Consider the Crank-Nicolson inspired JKO scheme (Carrillo et al., 2021) below

$$P_{k+1}=\operatorname*{arg\,min}_{P\in\mathcal{P}_{ac}(\mathbb{R}^n)}\frac{1}{2a}W_2^2(P,P_k)+\frac{1}{2}\mathcal{F}(P)+\frac{1}{2}\int\frac{\delta\mathcal{F}}{\delta P}(P_k)P.$$

The difficulty of implementing this scheme with neural-network based method is the easy access to the density of P_k. The predecessors Mokrov et al. (2021) and Alvarez-Melis et al. (2021) don't have this property, while in our algorithm, $P_k\approx h_{k-1}\Gamma_{k-1}(k>1)$. This is because our optimal h_k is equal to or can be transformed to the ratio between densities of P_{k+1} and Γ_k. Assume h can learn to approximate P_{k+1}/Γ_k, our method can be natually extended to Crank-Nicolson inspired JKO scheme.

C. Evaluation of the density

In this section, we assume the solving process doesn't use forward-backward scheme, i.e. all the probability measures P_k are obtained by performing JKO one by one. Otherwise, the map $I-a\nabla_x(W*P_k)=I-\mathbb{E}_{y\sim P_k}\nabla_x(W(x-y))$ includes an expectation term and becomes intractable to push-backward particles to compute density.

If T is invertible, these exists a standard approach, which we present here for completeness, to evaluate the density of P_k (Alvarez-Melis et al., 2021; Mokrov et al., 2021) through the change of variables formula. More specifically, we assume T is parameterized by the gradient of an ICNN φ that is assumed to be strictly convex. Thus we can guarantee that the gradient $\nabla\varphi$ invertible. To evaluate the density $P_k(x_k)$ at point x_k, we back propagate through the sequence of maps $T_k=\nabla\varphi_k,\dots,T_1=\nabla\varphi_1$ to get

$$x_i=T_{i+1}^{-1}\circ T_{i+2}^{-1}\circ\cdots\circ T_k^{-1}(x_k).$$

The inverse map $T_j^{-1}=(\nabla\varphi_j)^{-1}=\nabla\varphi_j^*$ can be obtained by solving the convex optimization

$$x_{j-1}=\operatorname*{arg\,max}_{x\in\mathbb{R}^n}\langle x,x_j\rangle-\varphi_j(x). \tag{14}$$

Then, by the change of variables formula, we obtain

$$\log[dP_k(x_k)]=\log[dP_0(x_0)]-\sum_{i=1}^k\log\left|\nabla^2\varphi_i(x_{i-1})\right|, \tag{15}$$

where $\nabla^2\varphi_i(x_{i-1})$ is the Hessian of φ_i and $|\nabla^2\varphi_i(x_{i-1})|$ is its determinant. By iteratively solving (14) and plugging the resulting x_j into (15), we can recover the density $dP_k(x_k)$ at any point.

D. Extension to the interaction energy functional

In this section, we consider $\mathcal{F}(P)$ involves the interaction energy

$$\mathcal{F}(P) = \mathcal{W}(P) := \int\int W(x-y)P(x)P(y)dxdy,$$
$$W : \mathbb{R}^n \to \mathbb{R} \text{ is symmetric, i.e. } W(x) = W(-x).$$

D.1. Forward Backward (FB) scheme

When $\mathcal{F}(P)$ involves the interaction energy $\mathcal{W}(P)$, we add an additional forward step to solve the gradient flow:

$$P_{k+\frac{1}{2}} := (I - a\nabla_x(W * P_k))\sharp P_k \tag{16}$$
$$P_{k+1} := T_{k+\frac{1}{2}}\sharp P_{k+\frac{1}{2}}, \tag{17}$$

where I is the identity map, and $T_{k+\frac{1}{2}}$ is defined by replacing k by $k+\frac{1}{2}$ in (8). In other words, the first gradient descent step (16) is a forward discretization of the gradient flow and the second JKO step (17) is a backward discretization. $\nabla_x(W * P)$ can be written as expectation $\mathbb{E}_{y\sim P}\nabla_x(W(x-y))$, thus can also be approximated by samples. The computation complexity of step (16) is at most $O(N^2)$ where N is the total number of particles to push-forward. This scheme has been studied as a discretization of gradient flows and proved to have sublinear convergence to the minimizer of $\mathcal{F}(P)$ under some regular assumptions (Salim et al., 2020). We make this scheme practical by giving a scalable implementation of JKO.

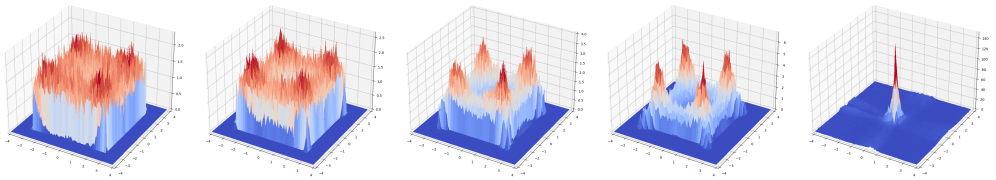
Since $\mathcal{W}(P)$ can be equivalently written as expectation $\mathbb{E}_{x,y\sim P}[W(x-y)]$, there exists another non-forward-backward (non-FB) method , i.e., removing the first step and integrating $\mathcal{W}(P)$ into a single JKO step: $P_{k+1} = T_k\sharp P_k$ and

$$T_k = \arg\min_T (\mathbb{E}_{P_k}\|X - T(X)\|^2/2a$$
$$+\mathbb{E}_{X,Y\sim P_k}[W(T(X) - T(Y))] + \sup_h \mathcal{V}(T,h)).$$

In practice, we observe the FB scheme is more stable and gives more regular results however converge slower than non-FB scheme. The detailed discussion appears in the Appendix D.3, D.4.

Remark D.1. In principle, one can single out $\log(Q)$ term from (13) and perform a similar forward step $P_{k+\frac{1}{2}} = (I - a(\nabla_x Q)/Q)\sharp P_k$ (Salim et al., 2020), but we don't observe improved performance of doing this in sampling task.

D.2. Simulations to Aggregation–Diffusion Equation with FB scheme



(a) $k = 24$ (b) $k = 36$ (c) $k = 60$ (d) $k = 84$ (e) $k = 92$

Figure 8: Histogram for simulated measures P_k by FB scheme at different k.

We finally simulate the evolution of solutions to the following aggregation-diffusion equation:

$$\partial_t P = \nabla \cdot (P\nabla W * P) + 0.1\Delta P^m,\ W(x) = -e^{-\|x\|^2}/\pi.$$

This corresponds to the energy function $\mathcal{W}(P) + 0.1\mathcal{G}(P)$. There is no explicit closed-form solution for this equation except for the known singular steady state (Carrillo et al., 2019b), thus we only provide qualitative results in Figure 8. We use the same parameters in Carrillo et al. (2021, Section 4.3.3). The initial distribution is a uniform distribution supported on $[-3,3]\times[-3,3]$ and the JKO step size $a = 0.5$. We utilize FB scheme to simulate the gradient flow for this equation with

$m = 3$ on $\mathbb{R}^2$ space. With this choice $W(x)$, $\nabla_x(W * P_k)$ is equal to $\mathbb{E}_{y\sim P_k}\left[2e^{-\|x-y\|^2}/\pi\right]$ in the gradient descent step (16). And we estimate $\nabla_x(W * P_k)$ with 10^4 samples from P_k.

Throughout the process, the aggregation term $\nabla \cdot (P\nabla W * P)$ and the diffusion $0.1\Delta P^m$ adversarially exert their effects and cause the probability measure split to four pulses and converge to a single pulse in the end. Our result aligns with the simulation of discretization method (Carrillo et al., 2021) well.

D.3. Simulation solutions to Aggregation equation

Alvarez-Melis et al. (2021) proposes using the neural network based JKO, i.e. the backward method, to solve (18). They parameterize T as the gradient of the ICNN. In this section, we use two cases to compare the forward method and backward when $\mathcal{F}(P) = \mathcal{W}(P)$. This could help explain the FB and non-FB scheme performance difference later in Section D.4.

We study the gradient flow associated with the aggregation equation

$$\partial_t P = \nabla \cdot (P\nabla W * P), \quad W : \mathbb{R}^n \to \mathbb{R}. \tag{18}$$

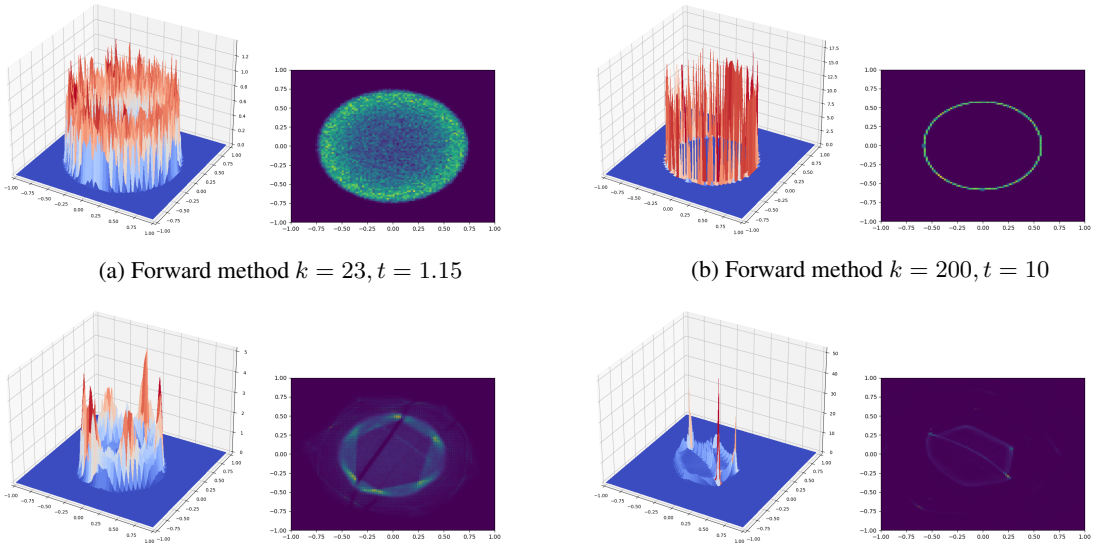
The forward method is

$$P_{k+1} := (I - a\nabla_x(W * P_k))\sharp P_k.$$

The backward method or JKO is

$$P_{k+1} := T_k\sharp P_k, \quad T_k = \arg\min_T \left\{ \frac{1}{2a}\mathbb{E}_{P_k}[\|X - T(X)\|^2] + \mathbb{E}_{X,Y\sim P_k}[W(T(X) - T(Y))] \right\}.$$

Example 1 We follow the setting in Carrillo et al. (2021, Section 4.3.1). The interaction kernel is $W(x) = \frac{\|x\|^4}{4} - \frac{\|x\|^2}{2}$, and the initial measure P_0 is a Gaussian $\mathcal{N}(0, 0.25I)$. In this case, $\nabla_x(W * P_k)$ becomes $\mathbb{E}_{y\sim P_k}\left[(\|x - y\|^2 - 1)(x - y)\right]$. We use step size $a = 0.05$ for both methods and show the results in Figure 9.



(a) Forward method $k = 23, t = 1.15$

(b) Forward method $k = 200, t = 10$

(c) Backward method $k = 23, t = 1.15$

(d) Backward method $k = 40, t = 2$

Figure 9: The steady state is supported on a ring of radius 0.5. Backward converges faster to the steady rate but is unstable. As k goes large, it cannot keep the regular ring shape and will collapse after $k > 50$.

Example 2 We follow the setting in Carrillo et al. (2021, Section 4.2.3). The interaction kernel is $W(x) = \frac{\|x\|^2}{2} - \ln\|x\|$, and the initial measure P_0 is $\mathcal{N}(0, 1)$. The unique steady state for this case is

$$P_\infty(x) = \frac{1}{\pi}\sqrt{(2 - x^2)_+}.$$

The reader can refer to Alvarez-Melis et al. (2021, Section 5.3) for the backward method performance. As for the forward method, $\nabla_x(W * P_k)$ becomes $\mathbb{E}_{y\sim P_k}\left[x - y - \frac{1}{x-y}\right]$. Because the kernel W enforces repulsion near the origin and P_0 is concentrated around origin, $\nabla_x(W * P)$ will easily blow up. So the forward method is not suitable for this kind of interaction kernel.

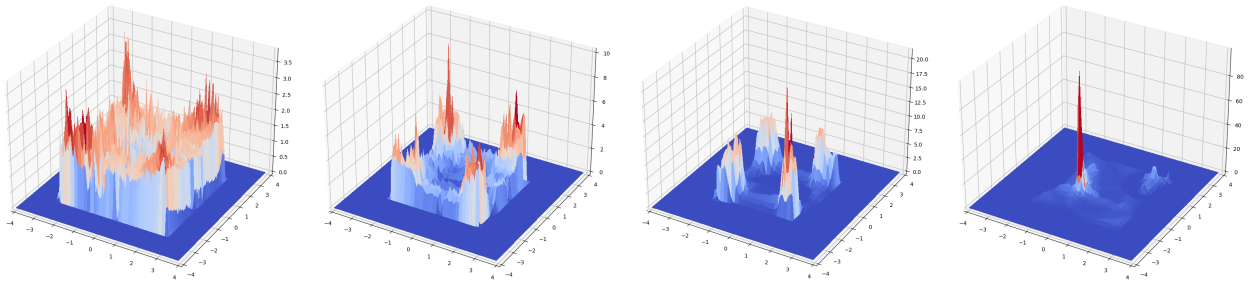
Through the above two examples, if $\nabla_x(W * P)$ is smooth, we can notice the backward method converges faster, but is not stable when solving (18). This shed light on the FB and non-FB scheme performance in Section D.2, D.4. However, if $\nabla_x(W * P)$ has bad modality such as Example 2, the forward method loses the competitivity.

D.4. Simulation solutions to Aggregation-diffusion equation with non-FB scheme

In Figure 10, we show the non-FB solutions to Aggregation-diffusion equation in Section D.2. FB scheme should be independent with the implementation of JKO, but in the following context, we assume FB and non-FB are both neural network based methods discussed in Section 3. Non-FB scheme reads

$$\begin{aligned} P_{k+1} &= T_k \sharp P_k \\ T_k &= \arg\min_T \left\{ \frac{1}{2a}\mathbb{E}_{P_k}[\|X - T(X)\|^2] + \mathbb{E}_{X,Y\sim P_k}[W(T(X) - T(Y))] + \mathcal{G}(T,h) \right\}, \end{aligned}$$

where $\mathcal{G}(T,h)$ is represented by the variational formula (10). We use the same step size $a = 0.5$ and other PDE parameters as in Section D.2.



(a) $k = 18$ (b) $k = 24$ (c) $k = 30$ (d) $k = 42$

Figure 10: Histograms for simulated measures P_k by non-FB scheme at different k.

Comparing the FB scheme results in Figure 8 and the non-FB scheme results in Figure 10, we observe non-FB converges $1.5\times$ slower than the finite difference method (Carrillo et al., 2021), and FB converges $3\times$ slower than the finite difference method. This may because splitting one JKO step to the forward-backward two steps removes the aggregation term effect in the JKO, and the diffusion term is too weak to make a difference in the loss. Note at the first several k, both P_k and Q are nearly the same uniform distributions, so h is nearly a constant and $T(x)$ exerts little effect in the variational formula of $\mathcal{G}(P)$. Another possible reason is a single forward step for aggregation term converges slower than integrating aggregation in the backward step, as we discuss in Section D.3 and Figure 9.

However, FB generates more regular measures. We can tell the four pulses given by FB are more symmetric. We speculate this is because gradient descent step in FB utilizes the geometric structure of $W(x)$ directly, but integrating $\mathcal{W}(P)$ in neural network based JKO losses the geometric meaning of $W(x)$.

E. Additional experiment results and discussions

E.1. Computational time

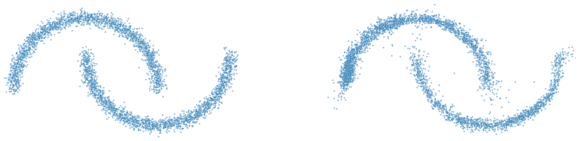
The forward step (16) takes about 14 seconds to pushforward one million points.

Other than learning generative model, assume each JKO step involves 500 iterations, the number of iterations $J_2 = 3$,

$J_3 = 2$, then the training of each JKO step (17) takes around 15 seconds.

For learning image generative model, assume $J_2 = 1$, $J_3 = 5$, then the training of each JKO step (17) takes around 20 minutes.

E.2. Low dimension sampling



(a) ground truth (b) ours

Figure 11: The left figure shows samples from the target 16-GMM distribution and the right figure shows samples obtained by our method. Each plot contains 4000 points.

The two moons distribution is a popular target distribution for sampling task. It is a 2D mixture model composed of 16 Gaussian components; each moon shape consists of 8 Gaussian components. The results are displayed in Figure 11, from which we see that our method is able to generate samples that match the target distribution.

E.3. Quality result about Porous media equation in 1D

In principle, our algorithm should match the analytical solution P_{gt} when the step size a is sufficiently small. When a is not that small, time discretization is inevitable. To account for the time-discretization error of JKO scheme, we consider the porous media equation in 1D space and use the solution via finite difference method as a reference. More specifically, in the 1D space $\mathbb{R}$, we discretize the density over a fixed grid with grid size d and grid resolution δx. With this discretization, the probability densities become (scaled) probability vectors and the problem (3) can be converted into a convex optimization

$$\min_{\pi:\delta x \pi^{\mathrm{T}}\mathbf{1}=\widehat{P}_k} \frac{(\delta x)^2}{2a}\langle \pi, M\rangle + \frac{\delta x}{m-1}(\mathbf{1}^{\mathrm{T}}\pi\delta x)^m\mathbf{1} \tag{19}$$

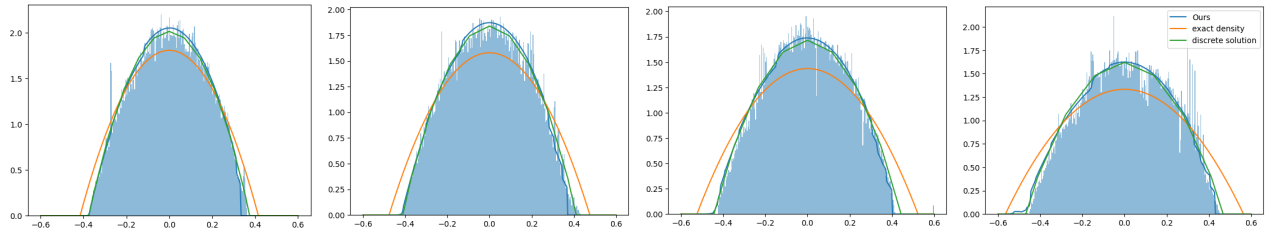
where M is the discretized unit transport cost, $\widehat{P}_k \in \mathbb{R}^d$ is the probability vector at the previous step, $\mathbf{1} \in \mathbb{R}^d$ is the all-ones vector and the optimization variable $\pi \in \mathbb{R}^d \times \mathbb{R}^d$ is the joint distribution between $\widehat{P}_k$ and $\widehat{P}_{k+1}$. This is a standard convex optimization and can be solved with generic solvers. When an optimal π is obtained, $\widehat{P}_{k+1}$ can be computed as $\widehat{P}_{k+1} = \pi\mathbf{1}$. We adopt the library `CVXOPT`[2] to solve the convex programming problem (19). In so doing, we arrive at a reference solution $\widehat{P}_0, \widehat{P}_1, \ldots, \widehat{P}_k, \ldots$ for our algorithm.

In the experiments, we set the stepsize for the JKO scheme to be $a = 0.001$ and the initial time to be $t_0 = 0.002$. Other parameters are chosen as $C = (3/16)^{1/3}, m = 2, n = 1, d = 300$. We parametrize the transport map T as the gradient of an ICNN and thus we can evaluate the density following Section C. In Figure 12, we observe that the gap between the density computed using our algorithm and the ground truth density P_{gt} is dominated by the time-discretization error of the JKO scheme. Our result matches the discrete time solution nicely.

F. Experiments implementation details other than image

Our experiments are conducted on GeForce RTX 3090 or RTX A6000. We always make sure the comparison is conducted on the same GPU card when comparing training time with other methods. Our code is written in Pytorch-Lightning (Falcon & Cho, 2020). We use other wonderful python libraries including W&B (Biewald, 2020), hydra (Yadan, 2019), seaborn (Waskom, 2021) etc. We also adopt the code given by Mokrov et al. (2021) for some experiments. For fast approximation of $\log\det\nabla^2\varphi$, we adapts the code given by Huang et al. (2020) with default parameters therein.

[2] `http://cvxr.com/cvx/`



(a) $t = 0.002$ (b) $t = 0.004$ (c) $t = 0.006$ (d) $t = 0.008$

Figure 12: Comparison among exact density, finite difference method solution given by `CVXOPT`, and the density given by our method. To better visualize the distributed particles from each distribution, we also plot the histograms of our method as the blue shadow.

Without further specification, we use the following parameters:

- The number of iterations: $J_1 = 600$. $J_2 = 3$. $J_3 = 1$.
- The batch size is fixed to be $M = 100$.
- The learning rate is fixed to be 0.001.
- All the activation functions are set to be PReLu.
- h has 3 layers and 16 neurons in each layer.
- T has 4 layers and 16 neurons in each layer.

The transport map T can be parametrized in different ways. We use a residual MLP network for it in Section 5.1, 5.2, 5.3, D.2, D.3, and the gradient of a strongly convex ICNN in Section 5.4, D.4. Except image task, the dual test function h is always a MLP network with quadratic or sigmoid actication function in the final layer to promise h is positive.
The networks T and h in Section 5.5 are chosen to be UNet and a normal CNN.

F.1. Learning of h

The learning of h is crucial because it determines the effectiveness of variational formula. In our KL divergence and generalized entropy variational formulas, the optimal h is equal to $T\sharp P_k/\Gamma$, which can have large Lipschitz constant in some high dimensional applications and become difficult to approximate. To tackle this issue, we replace h by $\exp(\bar{h}) - 1$, thus the optimal $\bar{h}$ is $\log(h+1)$, whose Lipschitz constant is much weakened. We apply this trick in Section 5.1 and observe the improved performance.

In image tasks, h works like a discriminator in GAN. A typical problem in GAN is that the discriminator can be too strong to let generator keep learning. To avoid this, we add the spectral normalization in h such that the Lipschitz of h is bounded by 1.

F.2. Calculation of error criteria

Sampling from GMM We estimate the kernelized Stein discrepancy (KSD) following the author's instructions (Liu et al., 2016). We draw N samples $X_1, \ldots, X_N$ from each method, and estimate KSD as

$$\mathrm{KSD}(P, Q) = \frac{1}{N(N-1)} \sum_{1 \le i \ne j \le N} u_Q(X_i, X_j),$$

where

$$u_Q(x, x') = s_q(x)^\top k(x, x')\, s_q(x') + s_q(x)^\top \nabla_{x'} k(x, x') + \nabla_x k(x, x')^\top s_q(x') + \mathrm{trace}\left(\nabla_{x,x'} k(x, x')\right),$$
$$s_Q = \nabla_x \log Q(x) = \frac{\nabla_x Q(x)}{Q(x)}.$$

Table 4: Hyper-parameters in the GMM convergence experiments.

Dimension	ℓ	Our methods				JKO-ICNN	
		T width	T depth	h width	h depth	width	depth
2	5	8	3	8	3	256	2
4	5	32	4	32	3	384	2
8	5	32	4	32	4	512	2
15	3	64	4	64	4	1024	2
17	3	64	4	64	4	1024	2
24	3	64	5	64	4	1024	2
32	3	64	5	64	4	1024	2
64	2	128	5	128	4	-	-
128	1.5	128	5	128	4	-	-

We choose the kernel ϕ to be the RBF kernel and use the same bandwidth for all methods. We fix $N = 1 \times 10^5$,

OU process For each method, we draw $5 \cdot 10^5$ samples from P_t and calculate the empirical mean $\widetilde{\mu}_t$ and covariance $\widetilde{\Sigma}_t$. Then we calculate the SymKL between $\mathcal{N}(\tilde{\mu}_t, \tilde{\Sigma}_t)$ and the exact solution.

Porous media equation We calculate the density of P_k according to Section C and estimate the SymKL using Monte Carlo according to the instructions in Mokrov et al. (2021).

F.3. Sampling from Gaussian Mixture Models (Section 5.1)

Two moons We run $K = 10$ JKO steps with $J_2 = 6, J_3 = 1$ inner iterations. h has 5 layers. T has 4 layers.

GMM The mean of Gaussian components are randomly sampled from Uniform$([-\ell/2, \ell/2]^n)$. $J_3 = 2$. The map T has dropout in each layer with probability 0.04. The learning rate of our method is $1 \cdot 10^{-3}$ for the first 20 JKO steps and $4 \cdot 10^{-4}$ for the last 20 JKO steps. The learning rate of JKO-ICNN is $5 \cdot 10^{-3}$ for the first 20 JKO steps, and then $2 \cdot 10^{-3}$ for the rest steps. The batch size is 512 and each JKO step runs 1000 iterations for all methods. The rest parameters are in Table 4.

F.4. Ornstein-Uhlenbeck Process (Section 5.2)

We use nearly all the same hyper-parameters as Mokrov et al. (2021), including learning rate, hidden layer width, and the number of iterations per JKO step. Specifically, we use a residual feed-forward NN to work as T, i.e. without activation function. h and T both have 2 layers and 64 hidden neurons per layer for all dimensions. We also train them for $J_1 = 500$ iterations per each JKO with learning rate 0.005. The batch size is $M = 1000$.

F.5. Bayesian Logistic Regression (Section 5.3)

Same as Mokrov et al. (2021), we use JKO step size $a = 0.1$ and calculate the log-likelihood and accuracy with 4096 random parameter samples. The rest parameters are in Table 6.

F.6. Porous media equation (Section 5.4)

We use rejection sampling (Eckhardt et al., 1987) to sample from P_0 because its computational time is more promising than MCMC methods. However, the rejection sampling acceptance rate is expected to be exponentially small (MacKay & Mac Kay, 2003, Ch 29.3) in dimension, and empirically it's intractable when $n > 6$. So we only give the results for $n \leq 6$.

In the experiment, h have 4 layers and 16 neurons in each layer with CELU activation functions except the last layer, which is activated by PReLU. To parameterize the map, we adopt DenseICNN (Korotin et al., 2021a) structure with width 64, depth 2 and rank 1. The batch size is $M = 1024$. Each JKO step runs $J_1 = 1000$ iterations. The learning rate for both φ and h is $1 \cdot 10^{-3}$. $J_3 = 1$ for dimension 3 and $J_3 = 2$ for dimension 6.

Table 5: Bayesian logistic regression accuracy and log-likelihood full results.

Dataset	# features	dataset size	Accuracy			Log-Likelihood		
			Ours	JKO-ICNN-d	SVGD	Ours	JKO-ICNN-d	SVGD
covtype	54	581012	0.753	0.75	0.75	-0.528	-0.515	-0.515
splice	60	2991	0.84	0.845	0.85	-0.38	-0.36	-0.355
waveform	21	5000	0.785	0.78	0.765	-0.455	-0.485	-0.465
twonorm	20	7400	0.982	0.98	0.98	-0.056	-0.059	-0.062
ringnorm	20	7400	0.73	0.74	0.74	-0.5	-0.5	-0.5
german	20	1000	0.67	0.67	0.65	-0.59	-0.6	-0.6
image	18	2086	0.866	0.82	0.815	-0.394	-0.43	-0.44
diabetis	8	768	0.786	0.775	0.78	-0.45	-0.45	-0.46
banana	2	5300	0.55	0.55	0.54	-0.69	-0.69	-0.69

Table 6: Hyper-parameters in the Bayesian logistic regression.

Dataset	K	M	J_1	T width	T depth	h width	h depth	T learning rate	h learning rate
covtype	7	1024	7000	128	4	128	3	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
splice	50	1024	400	128	5	128	4	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
waveform	5	1024	1000	32	4	32	4	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
twonorm	15	512	800	32	4	32	3	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
ringnorm	9	1024	500	32	4	32	4	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$
german	14	800	640	32	4	32	4	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
image	12	512	1000	32	4	32	4	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
diabetis	16	614	835	32	4	32	3	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
banana	16	512	1000	16	2	16	2	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$

F.7. Aggregation-diffusion equation (Section D.2 and D.4)

Each JKO step contains $J_1 = 200$ iterations. The batch size is $M = 1000$.

G. Image experiments

G.1. Hyperparameters and network architecture

We use Adam optimizer with learning rate 2×10^{-4} and other default settings in PyTorch library. We choose $J_2 = 1, J_3 = 5$. Our h network follows the architecture of ResNet classifier network (He et al., 2016). More specially, our module uses two downsampling modules, which results in three feature map resolution $(32 \times 32, 16 \times 16, 8 \times 8)$. We use two convolutional residual blocks for each resolution and pass the features extracted from at 8×8 resolution into a 2-layer MLP. We use 128 channels for CNN and 128 hidden neurons for the MLP. Similar to training generative adversarial networks, we found adding regularizers on h network can help stabilize training. Thus, we apply the spectral normalization (Miyato et al., 2018) on h network.

Our framework requires the T_k networks to approximate mappings between same dimensional data spaces. Our network architecture follows the backbone of PixelCNN++ (Salimans et al., 2017), which can be viewed as a modified U-Net (Ronneberger et al., 2015) based on Wide ResNet (Zagoruyko & Komodakis, 2016). More specifically, we use 3 downsampling and 3 upsampling modules, which results in four feature map resolutions $(32 \times 32, 16 \times 16, 8 \times 8, 4 \times 4)$. At each resolution, we have two convolutional residual blocks. We use $64, 128, 256, 512$ channels for as image resolution decreases.

Here are more training details:

- We resize MNIST image to 32×32 resolution so that we h, T_k networks can work on both MNIST and CIFAT10 with small modification of input channel.
- We use random horizontal flips during training for CIFAR10.
- We use batch size $M = 128$.

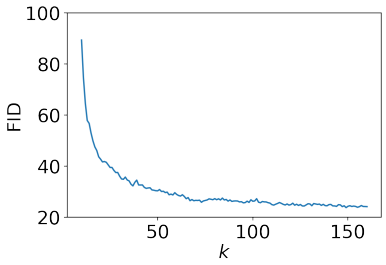


Figure 13: The FID score converges as k increases on CIFAR10 datset.

Figure 14: Mode collapsing in GANs.

- On CIFAR10, we use implementation from *torch-fidelity*[3] to calculate FID scores with 50k samples.
- The JKO step size a controls the divergence between P_k and P_{k+1}. We observe training with small a has unstable issues and mode collapse, a large a suffers from slower convergence. We found $a = 5$ works well on both MNIST and CIFAR10 datasets.
- We use 10 epochs to train each P_k, we notice P_{30} generates realistic images when $a = 5.0$. However, we find FID score decreases as k increases. We present the change of FID score of samples from different P_k in Figure 13.

G.2. More Comparison

Comparison with GANs. As we use Jensen-Shannon divergence in our scheme, JKO-Flow specializes to Jensen-Shannon GAN when $a \to \infty, K = 1$. However, we found training with $a \to \infty, K = 1$ is very unstable and suffer mode collapsing occasionally. Though training GANs can not recover the gradient flow from noise to image, it is interesting to compare JKO-Flow and GANs in term of sampling quality. To make a fair comparison, we instantiate generator network as the same as T_k network and discriminator as h for GANs. We note such choice is not optimal for GAN since generators in existing works usually map a lower dimensional Gaussian noise into images instead of mapping from same dimensional space. We believe the comparison and JKO-Flow scheme may help future research when modeling mapping between same dimensional data spaces. As shown in Table 7, JKO-Flow enjoys better sample qualities. Empirically we found training GANs is more challenging when latent space is relative large and with more complex generator networks as mode collapsing becomes more common. We find the additional Wasserstein distance loss in JKO-Flow can be viewed a regularizer to avoid mode collapsing because T_k will receive large penalty if it maps all inputs into a local minimal. However, one shortcoming of our method is the scheme of JKO-Flow needs to model a sequence of generators instead of one generate that push P_0 particles into Q, and small step size controlled by a resulted in slower convergence and more training time.

Comparison with more generative models based on gradient flows and optimal transport maps. Most existing works in this line focus on the latent spaces of pre-trained autoencoders (Seguy et al., 2017; An et al., 2019; 2020; Makkuva et al.,

[3] https://github.com/toshas/torch-fidelity

Method	FID score ↓
GAN (JKO-Flow with $a \to \infty, K = 1$)	≥ 80
WGAN-GP	62.3
SN-GAN	43.2
JKO-Flow	**23.1**

Table 7: Comparison between JKO-Flow and various GANs. The generator and discriminator networks in GANs follow same architecture of P_k and h network in JKO-Flow.

Method	FID score ↓	Inception Score ↑
AE-OT (An et al., 2019)	28.5	-
AE-OT-GAN (An et al., 2020)	17.1	-
OTM (Rout et al., 2021)	20.69	7.41 ± 0.11
JKO-Flow	23.1	7.48 ± 0.12

Table 8: More comparison among generative models on CIFAR10.

2020; Korotin et al., 2021a). The approach reduces burden of training gradients and optimal transport maps since tasks of modeling complex image modality and interactions between pixels are left to pre-trained decoders partially. We note the recent work Rout et al. (2021) investigates mappings between distributions located on the spaces with same dimensionality or unequal dimensionality. However, they only demonstrate the unconditional image generative model based on an embedding from a lower dimensional Gaussian distribution to image distributions. In contrast, we show JKO-Flow can learn complex mappings between both high dimensional distribution and achieve encouraging performance when applying such learned mappings in the challenging image generation task without additional conditional signal. We include more comparison in Table 8.

G.3. More generated samples and trajectories

We include more results of JKO-Flow. Figure 15, Figure 17, Figure 16, and Figure 18 show more generated samples from P_K and trajectories from JKO-Flow.

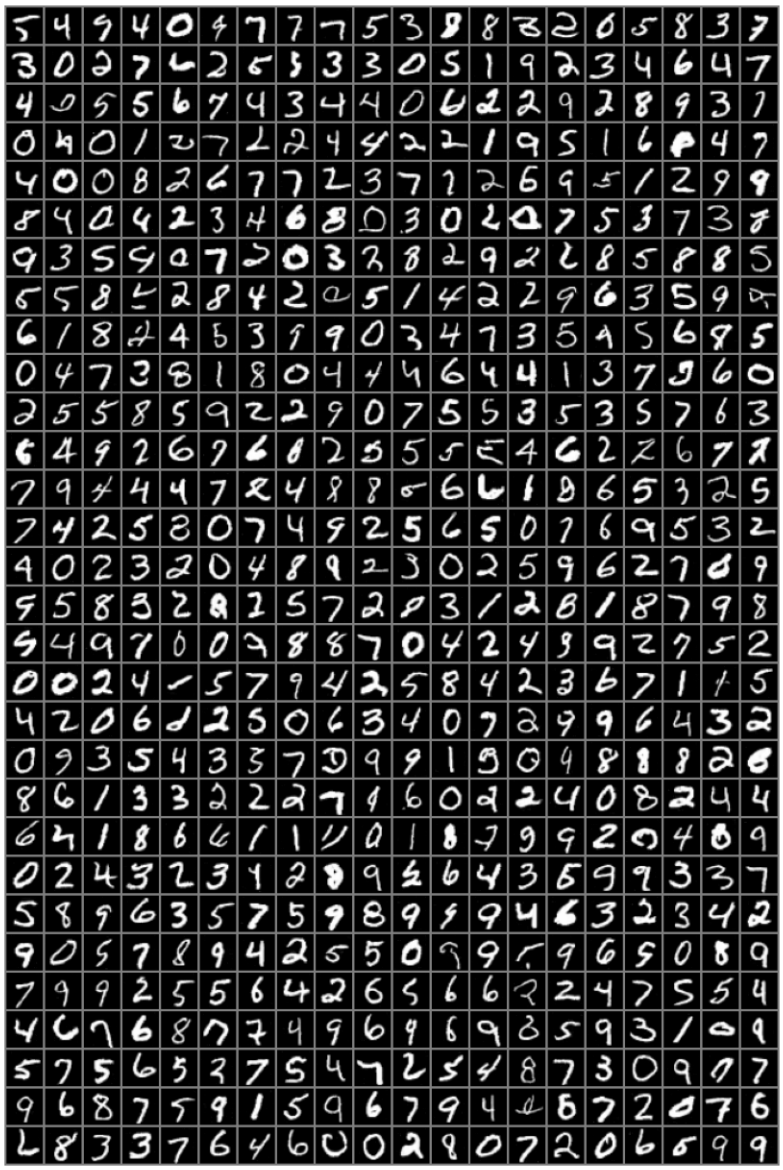


Figure 15: More MNIST sample from JKO-Flow

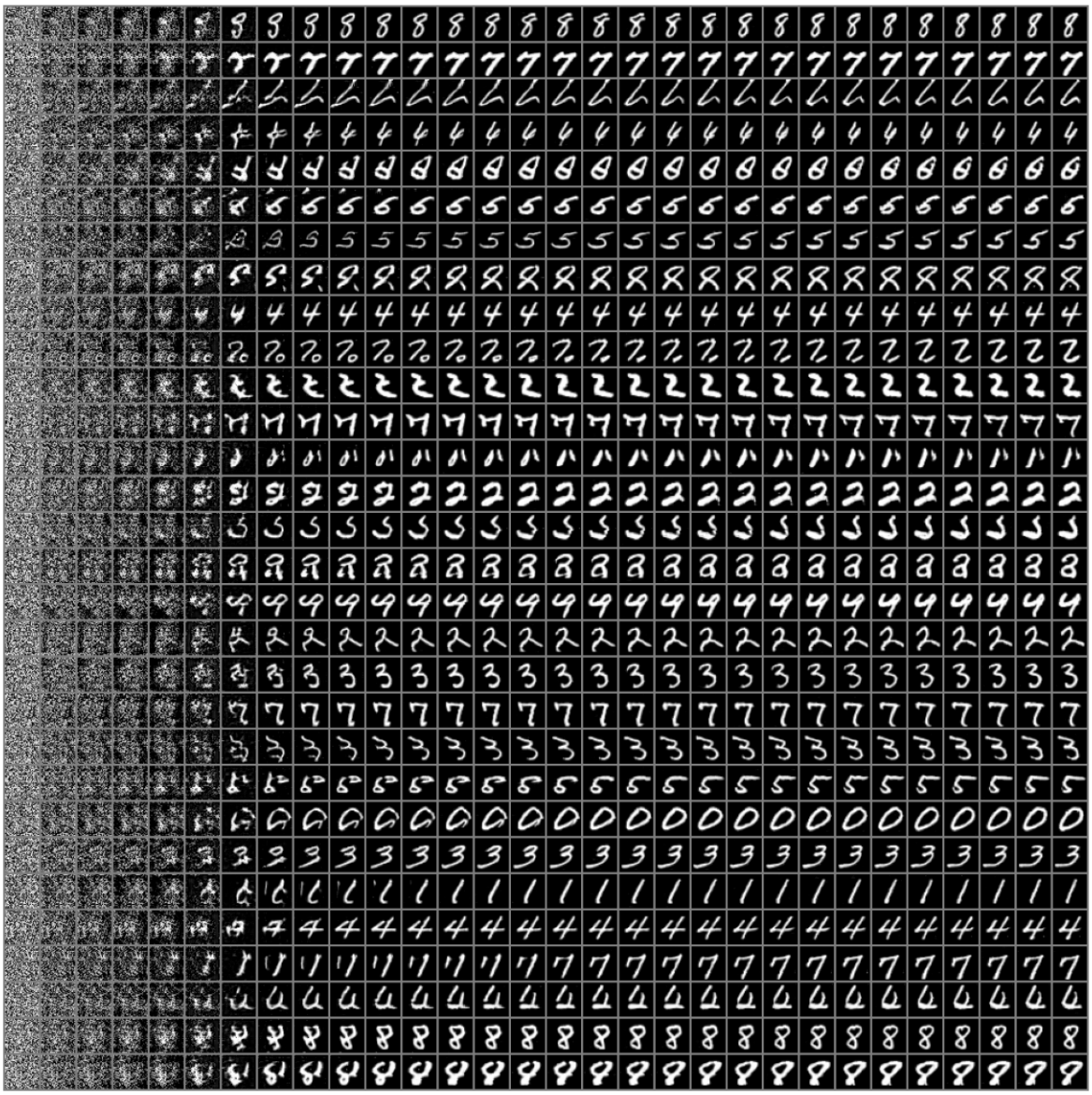


Figure 16: More MNIST trajectories from JKO-Flow with $K = 1$ to $K = 30$.

Figure 17: More CIFAR10 sample from JKO-Flow

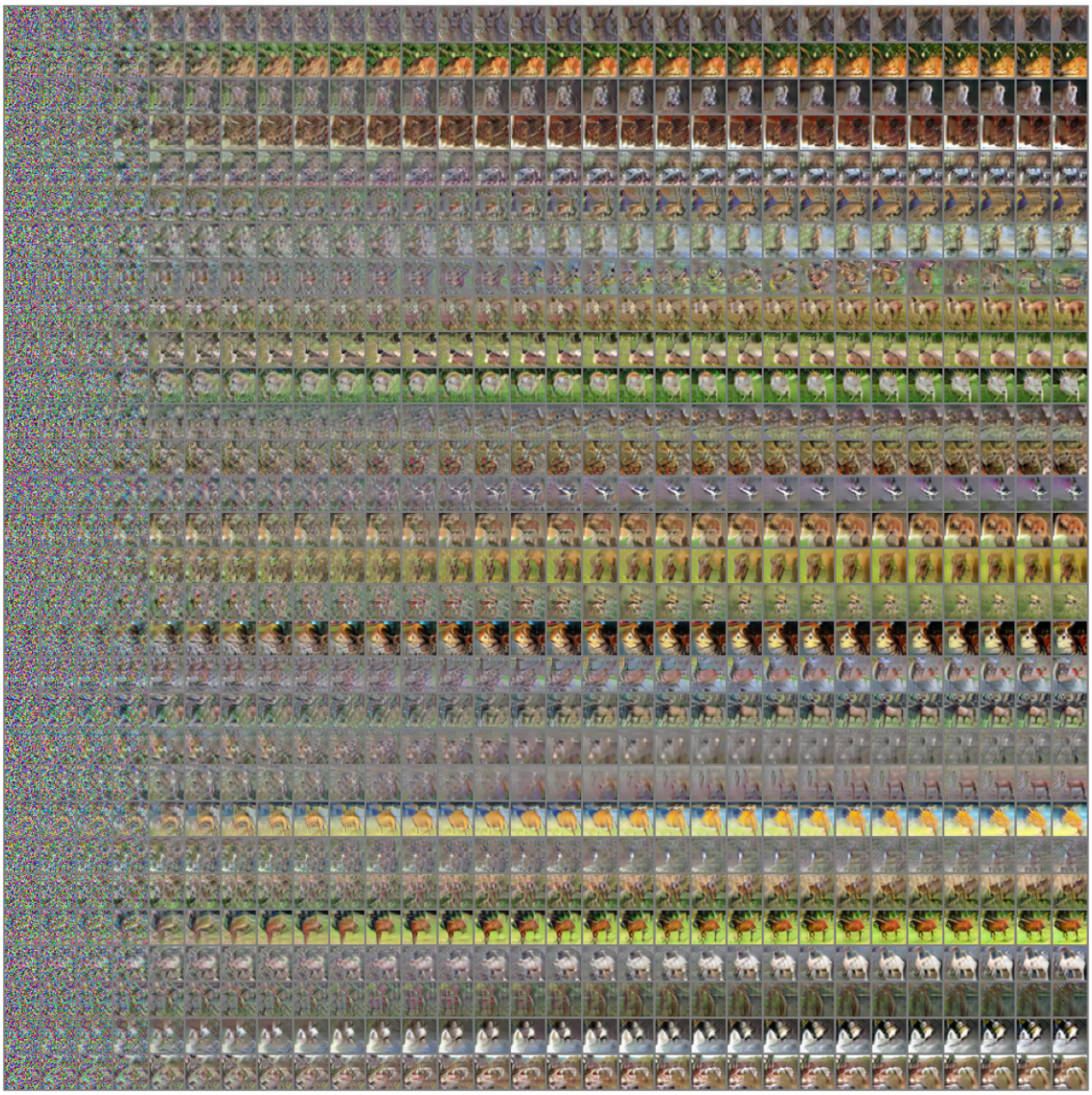


Figure 18: More CIFAR10 trajectories from JKO-Flow with $K = 1$ to $K = 30$.