

---

# Taming Hyperparameter Tuning in Continuous Normalizing Flows Using the JKO Scheme

---

**Alexander Vidal\***

Dept. of Applied Mathematics and Statistics  
Colorado School of Mines

**Samy Wu Fung**

Dept. of Applied Mathematics and Statistics  
Dept. of Computer Science  
Colorado School of Mines

**Luis Tenorio**

Dept. of Applied Mathematics and Statistics  
Colorado School of Mines

**Stanley Osher**

Dept. of Mathematics  
University of California, Los Angeles

**Levon Nurbekyan**

Dept. of Mathematics  
University of California, Los Angeles

## Abstract

A normalizing flow (NF) is a mapping that transforms a chosen probability distribution to a normal distribution. Such flows are a common technique used for data generation and density estimation in machine learning and data science. The density estimate obtained with a NF requires a change of variables formula that involves the computation of the Jacobian determinant of the NF transformation. In order to tractably compute this determinant, continuous normalizing flows (CNF) estimate the mapping and its Jacobian determinant using a neural ODE. Optimal transport (OT) theory has been successfully used to assist in finding CNFs by formulating them as OT problems with a soft penalty for enforcing the standard normal distribution as a target measure. A drawback of OT-based CNFs is the addition of a hyperparameter,  $\alpha$ , that controls the strength of the soft penalty and requires significant tuning. We present JKO-Flow, an algorithm to solve OT-based CNF without the need of tuning  $\alpha$ . This is achieved by integrating the OT CNF framework into a Wasserstein gradient flow framework, also known as the JKO scheme. Instead of tuning  $\alpha$ , we repeatedly solve the optimization problem for a fixed  $\alpha$  effectively performing a JKO update with a time-step  $\alpha$ . Hence we obtain a "divide and conquer" algorithm by repeatedly solving simpler problems instead of solving a potentially harder problem with large  $\alpha$ .

## 1 Introduction

A normalizing flow (NF) is a type of generative modeling technique that has shown great promise in applications arising in physics [40, 7, 10] as a general framework to construct probability densities for continuous random variables in high-dimensional spaces [47, 33, 52]. An NF provides a  $C^1$ -diffeomorphism  $f$  (i.e., a normalizing transformation) that transforms the density  $\rho_0$  of an initial distribution  $P_0$  to the density  $\rho_1$  of the standard multivariate normal distribution  $P_1$  – hence the term "normalizing." Given such mapping  $f$ , the density  $\rho_0$  can be recovered from the Gaussian density via

---

\*vidal@mines.edu

the change of variables formula,

$$\log \rho_0(x) = \log \rho_1(f(x)) + \log |\det J_f(x)|, \quad (1)$$

where  $J_f \in \mathbb{R}^{d \times d}$  is the Jacobian of  $f$ . Moreover, one can obtain samples with density  $\rho_0$  by pushing forward Gaussian samples via  $f^{-1}$ .

**Remark 1.** Throughout the paper we slightly abuse the notation, using the same notation for probability distributions and their density functions. Additionally, given a probability distribution  $P_0$  on  $\mathbb{R}^d$  and a measurable mapping  $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , we define the pushforward distribution of  $P_0$  through  $f$  as  $(f\#P_0)(B) = P_0(f^{-1}(B))$  for all Borel measurable  $B \subseteq \mathbb{R}^d$  [67].

There are two classes of normalizing flows: finite and continuous. A finite flow is defined as a composition of a finite number of  $\mathcal{C}^1$ -diffeomorphisms:  $f = f_1 \circ f_2 \circ \dots \circ f_n$ . To make finite flows computationally tractable, each  $f_i$  is chosen to have some regularity properties such as a Jacobian with a tractable determinant; for example,  $J_{f_i}$  may have a triangular structure [48, 5, 73].

On the other hand, continuous normalizing flows (CNFs) estimate  $f$  using a neural ODE of the form [11]:

$$\partial_t z(x, t) = v_\theta(z(x, t), t), \quad z(x, 0) = x, \quad 0 \leq t \leq T, \quad (2)$$

where  $\theta$  are the parameters of the neural ODE. In this case,  $f$  is defined as  $f(x) = z(x, T)$  (for simplicity, we remove the dependence of  $z$  on  $\theta$ ). One of the main advantages of CNFs is that we can tractably estimate the log-determinant of the Jacobian using Jacobi’s identity, which is commonly used in fluid mechanics (see, e.g., [67, p.114]):

$$\partial_t \log |\det \nabla_x z(x, t)| = \nabla_z \cdot v_\theta(z(x, t), t) = \text{trace}(\nabla_z v_\theta(z(x, t), t)). \quad (3)$$

This is computationally appealing as one can replace the expensive determinant calculation by a more tractable trace computation of  $\nabla_z v_\theta(z(x, t), t)$ . Importantly, no restrictions on  $\nabla_z v_\theta(z(x, t), t)$  (e.g., diagonal or triangular structure) are needed; thus, these Jacobians are also referred to as “free-form Jacobians” [23].

The goal in training a CNF is to find parameters,  $\theta$ , such that  $f = z(\cdot, T)$  leads to a good approximation of  $\rho_1$  or, assuming  $f$  is invertible, the pushforward of  $\rho_1$  through  $f^{-1}$  is a good approximation of  $\rho_0$  [52, 48, 47, 23]. Indeed, let  $\hat{\rho}_0$  be this pushforward density obtained with a CNF  $f$ ; that is,  $\hat{\rho}_0 = f^{-1\#}\rho_1$ . We then minimize the Kullback-Leibler (KL) divergence from  $\hat{\rho}_0$  to  $\rho_0$  given by

$$\min_{\theta} \mathbb{E}_{x \sim \rho_0} \log(\rho_0(x)/\hat{\rho}_0(x)) = \min_{\theta} \mathbb{E}_{x \sim \rho_0} [\log \rho_0(x) - \log \rho_1(z(x, T)) - \ell(x, T)],$$

where  $\ell(x, T) = \log |\det \nabla z(x, T)|$ . Dropping the  $\theta$ -independent term  $\log \rho_0$  and using (2) and (3), this previous optimization problem reduces to the minimization problem

$$\min_{\theta} \mathbb{E}_{x \sim \rho_0} C(x, T), \quad C(x, T) := -\log \rho_1(z(x, T)) - \ell(x, T) \quad (4)$$

subject to ODE constraints

$$\partial_t \begin{bmatrix} z(x, t) \\ \ell(x, t) \end{bmatrix} = \begin{bmatrix} v_\theta(z(x, t), t) \\ \text{trace}(\nabla_z v_\theta(z(x, t), t)) \end{bmatrix}, \quad \begin{bmatrix} z(x, 0) \\ \ell(x, 0) \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}. \quad (5)$$

The ODE (5) might be stiff for certain values of  $\theta$ , leading to extremely long computation times. Indeed, the dependence of  $v$  on  $\theta$  is highly nonlinear and might generate vector fields that lead to highly oscillatory trajectories with complex geometry.

Some recent work leverages optimal transport theory to find the CNF [19, 45]. In particular, a kinetic energy regularization term (among others) is added to the loss to “encourage straight trajectories”  $z(x, t)$ . That is, the flow is trained by minimizing the following instead of (4):

$$\min_{\theta} \mathbb{E}_{x \sim \rho_0} \int_0^T \frac{1}{2} \|v_\theta(z(x, t), t)\|^2 dt + \alpha C(x, T) \quad (6)$$

subject to (5). The key insight in [19, 45] is that (4) is an example of a degenerate OT problem with a soft terminal penalty and without a transportation cost. The first objective term in (6) given by the time integral is the transportation cost term, whereas  $\alpha$  is a hyperparameter that balances the soft penalty and the transportation cost. Including this cost makes the problem well-posed by forcing

the solution to be unique [68]. Additionally, it enforces straight trajectories so that (5) is not stiff. Indeed [19, 45] empirically demonstrate that including optimal transport theory leads to faster and more stable training of CNFs. Intuitively, we minimize the KL divergence *and* the arclength of the trajectories.

Although including optimal transport theory into CNFs has been very successful [45, 19, 71, 74], there are two key challenges that render them difficult to train. First, estimating the log-determinant in (4) via the trace in (5) is still computationally taxing and commonly used methods rely on stochastic approximations [23, 19], which adds extra error. Second, including the kinetic energy regularization requires tuning of the hyperparameter  $\alpha$ . Indeed, if  $\alpha$  is chosen too small in (6), then the kinetic regularization term dominates the training process, and the optimal solution consists of not moving, i.e.,  $f(x) = x$ . On the other hand, if  $\alpha$  is chosen too large, we return to the original setting where the problem is ill-posed, i.e., there are infinitely many solutions. Finally, finding an "optimal"  $\alpha$  is problem dependent and requires tuning on a cases-by-case basis.

## 1.1 Our Contribution

We present JKO-Flow, an optimal transport-based algorithm for training CNFs without the need to tune the hyperparameter  $\alpha$  in (6). Our approach also leverages fast numerical methods for exact trace estimation from the recently developed optimal transport flow (OT-Flow) [45, 55]. The key idea is to integrate the OT-Flow approach into a Wasserstein gradient flow framework, also known as the Jordan, Kinderlehrer, and Otto (JKO) scheme [30]. Rather than tuning the hyperparameter  $\alpha$  (commonly done using a grid search), the idea is to simply pick any  $\alpha$  and solve a sequence of "easier" OT problems that gradually approach the target distribution. Each solve is precisely a gradient descent in the space of distributions, a Wasserstein gradient descent, and the scheme provably converges to the desired distribution for all  $\alpha > 0$  [56]. Our experiments show that our proposed approach is effective in generating higher quality samples (and density estimates) and also allows us to reduce the number of parameters required to estimate the desired flow.

Our strategy is reminiscent of debiasing techniques commonly used in inverse problems. Indeed, the transportation cost that serves as a regularizer in (6) introduces a bias – the smaller  $\alpha$  the more bias is introduced (see, e.g., [66]), so good choices of  $\alpha$  tend to be larger. One way of removing the bias and the necessity of tuning the regularization strength is to perform a sequence of Bregman iterations [46, 9] also known as nonlinear proximal steps. Hence our approach reduces to debiasing via Bregman or proximal steps in the Wasserstein space. In the context of CNF training, Bregman iterations are advantageous due to the flexibility of the choice for  $\alpha$ . Indeed, the resulting loss function is non-convex and its optimization tends to get harder for large  $\alpha$ . Thus, instead of solving one harder problem we solve several "easier" problems.

## 2 Optimal Transport Background and Connections to CNFs

Denote by  $\mathcal{P}_2(\mathbb{R}^d)$  the space of Borel probability measures on  $\mathbb{R}^d$  with finite second-order moments, and let  $\rho_0, \rho_1 \in \mathcal{P}_2(\mathbb{R}^d)$ . The quadratic optimal transportation (OT) problem (which also defines the Wasserstein metric  $W_2$ ) is then formulated as

$$W_2^2(\rho_0, \rho_1) = \inf_{\pi \in \Gamma(\rho_0, \rho_1)} \int_{\mathbb{R}^{2d}} \|x - y\|^2 d\pi(x, y), \quad (7)$$

where  $\Gamma(\rho_0, \rho_1)$  is the set of probability measures  $\pi \in \mathcal{P}(\mathbb{R}^{2d})$  with fixed  $x$  and  $y$ -marginals densities  $\rho_0$  and  $\rho_1$ , respectively. Hence the cost of transporting a unit mass from  $x$  to  $y$  is  $\|x - y\|^2$ , and one attempts to transport  $\rho_0$  to  $\rho_1$  as cheaply as possible. In (7),  $\pi$  represents a *transportation plan*, and  $\pi(x, y)$  is the mass being transported from  $x$  to  $y$ . One can prove that  $(\mathcal{P}_2(\mathbb{R}^d), W_2)$  is a complete separable metric space [67]. OT has recently become a very active research area in PDE, geometry, functional inequalities, economics, data science and elsewhere partly due to equipping the space of probability measures with a (Riemannian) metric [67, 68, 50, 60].

As observed in prior works, there are many similarities between OT and NFs [45, 19, 72, 74]. This connection becomes more transparent when considering the dynamic formulation of (7). More

precisely, the Benamou-Brenier formulation of the OT problem is given by [6]:

$$\begin{aligned} \frac{T}{2} W_2^2(\rho_0, \rho_1) &= \inf_{v, \rho} \int_0^T \int_{\mathbb{R}^d} \frac{1}{2} \|v(x, t)\|_2^2 \rho(x, t) dx dt \\ \text{s.t. } &\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t) v(x, t)) = 0 \\ &\rho(x, 0) = \rho_0(x), \quad \rho(x, T) = \rho_1(x). \end{aligned} \quad (8)$$

Hence, the OT problem can be formulated as a problem of flowing  $\rho_0$  to  $\rho_1$  with a velocity field  $v$  that achieves minimal kinetic energy. The optimal velocity field  $v$  has several appealing properties. First, particles induced by the optimal flow  $v$  travel in straight lines. Second, particles travel with constant speed. Moreover, under suitable conditions on  $\rho_0$  and  $\rho_1$ , the optimal velocity field is unique [67].

Given a velocity field  $v$ , denote by  $z(x, t)$  the solution of the ODE

$$\partial_t z(x, t) = v(z(x, t), t), \quad z(x, 0) = x, \quad 0 \leq t \leq T.$$

Then, under suitable regularity conditions, we have that the solution of the continuity equation is given by  $\rho(\cdot, t) = z(\cdot, t) \# \rho_0$ . Thus the optimization problem in (8) can be written as

$$\begin{aligned} \inf_v \int_0^T \int_{\mathbb{R}^d} \frac{1}{2} \|v(z(x, t), t)\|_2^2 \rho_0(x) dx dt \\ \text{s.t. } \partial_t z(x, t) = v(z(x, t), t), \quad z(x, 0) = x, \quad z(\cdot, T) \# \rho_0 = \rho_1. \end{aligned} \quad (9)$$

This previous problem is very similar to (4) with the following differences:

- the objective function in (4) does not have the kinetic energy of trajectories,
- the terminal constraint is imposed as a soft constraint in (4) and as a hard constraint in (9), and
- $v$  in (4) is  $\theta$ -dependent, whereas the formulation in (9) is in the non-parametric regime.

So the NF (4) can be thought of as an approximation to a degenerate transportation problem that lacks transportation cost. Based on this insight one can regularize (4) by adding the transportation cost and arrive at (6) or some closely related version of it [45, 19, 72, 74]. It has been observed that the transportation cost (kinetic energy) regularization significantly improves the training of NFs.

### 3 JKO-Flow: Wasserstein Gradient Flows for CNFs

While the OT-based formulation of CNFs in (6) has been found successful in some applications [45, 19, 72, 74], a key difficulty arises in choosing how to balance the kinetic energy term and the KL-divergence, i.e., on selecting  $\alpha$ . This difficulty is typical in problems where the constraints are imposed in a soft fashion. Standard training of CNFs typically involves tuning for a “large but hopefully stable enough” step size  $\alpha$  so that the KL divergence term is sufficiently small after training. To this end, we propose an approach that avoids the need to tune  $\alpha$  by using the fact that the solution to (6) is an approximation to a backward Euler (or proximal point) algorithm when discretizing the Wasserstein gradient flow using the Jordan-Kinderlehrer-Otto (JKO) scheme [30]. The seminal work in [30] provides a gradient flow structure of the Fokker-Planck equation using an implicit time discretization. That is, given  $\alpha > 0$ , density at  $k^{\text{th}}$  iteration,  $\rho^{(k)}$ , and terminal density  $\rho_1$ , one finds

$$\begin{aligned} \rho^{(k+1)} &= \arg \min_{\rho \in \mathcal{P}_2(\mathbb{R}^d)} \frac{1}{2\alpha} W_2^2(\rho, \rho^{(k)}) + KL(\rho || \rho_1) \\ &= \arg \min_v \frac{1}{\alpha} \int_0^1 \int_{\mathbb{R}^d} \frac{1}{2} \|v(z(x, t), t)\|_2^2 \rho_0(x) dx dt + KL(z(\cdot, 1) \# \rho^{(k)} || \rho_1) \\ \text{s.t. } &\partial_t z(x, t) = v(z(x, t), t), \quad z(x, 0) = x, \quad z(\cdot, T) \# \rho_0 = \rho_1, \end{aligned} \quad (10)$$

for  $k = 0, 1, \dots$ , and  $\rho^{(0)} = \rho_0$ . Here,  $\alpha$  takes the role of a step size when applying a proximal point method to the KL divergence using the Wasserstein-2 metric, and  $\{\rho^{(k)}\}$  provably converges to

---

**Algorithm 1** Proposed Algorithm

---

- 1: **Input:** Samples from  $\rho_0$ , step size  $\alpha > 0$ , number of steps  $K$
  - 2: Initialize  $\theta_1$  at random
  - 3: **for**  $k = 1, \dots, K$  **do**
  - 4:   Solve for  $\theta_k$  using samples by solving (11)
  - 5:   Update distribution of samples using  $v_{\theta_k}$
  - 6: **end for**
  - 7: **Output:** saved weights  $\theta_1, \dots, \theta_K$
- 

$\rho_1$  [30, 56]. Hence, repeatedly solving (9) with the KL penalty acting as a soft constraint yields an arbitrarily accurate approximation of  $\rho_1$ . In the parametric regime each iteration takes the form

$$\arg \min_{\theta} \mathbb{E}_{x \sim \rho^{(k)}} \int_0^T \frac{1}{2} \|v_{\theta}(x, t)\|^2 dt + \alpha C(x, T) \quad \text{subject to (5)}. \quad (11)$$

Thus we solve a sequence of problems (6), where the initial density of the current subproblem is given by the pushforward of the density generated in the previous subproblem. Importantly, our proposed approach *does not require tuning*  $\alpha$ . Instead, we solve a sequence of subproblems that is guaranteed to converge to  $\rho_1$  [30] prior to the neural network parameterization; see Alg. 1. While our proposed methodology can be used in tandem with any algorithm used to solve (11), an important numerical aspect in our approach is to leverage fast computational methods that use *exact* trace estimation in (5); this approach is called OT-Flow [45]. Consequently, we avoid the use of stochastic approximation methods for the trace, e.g., Hutchinson’s estimator [4, 29, 66], as is typically done in CNF methods [23, 19]. A surprising result of our proposed method is that it empirically shows improved performance even with fewer number of parameters (see Fig. 3).

## 4 Related Works

**Density Estimation.** One of the main advantages of NFs over other generative models is that they provide density estimates of probability distributions using (1). That is, we do not need to apply a separate density estimation technique after generating samples from a distribution, e.g., as in GANs [22]. Multivariate density estimation is a fundamental problem in statistics [62, 61], High Energy Physics (HEP) [14] and in other fields of science dealing with multivariate data. For instance, particle physicists in HEP study possible distributions from a set of high energy data. Another application of density estimation is in confidence level calculations of particles in Higgs searches at Large Electron Positron Colliders (LEP) [13] and discriminant methods used in the search for new particles [14].

**Finite Flows.** Finite normalizing flows [64, 52, 47, 33] use a composition of discrete transformations, where specific architectures are chosen to allow for efficient inverse and Jacobian determinant computations. NICE [15], RealNVP [16], IAF [32], and MAF [48] use either autoregressive or coupling flows where the Jacobian is triangular, so the Jacobian determinant can be tractably computed. GLOW [31] expands upon RealNVP by introducing an additional invertible convolution step. These flows are based on either coupling layers or autoregressive transformations, whose tractable invertibility allows for density evaluation and generative sampling. Neural Spline Flows [17] use splines instead of the coupling layers used in GLOW and RealNVP. Using monotonic neural networks, NAF [28] require positivity of the weights, which UMNN [69] circumvent this requirement by parameterizing the Jacobian and then integrating numerically.

**Continuous and Optimal Transport-based Flows.** Modeling flows with differential equations is a natural and commonly used method [63, 70, 39, 57, 54, 27]. In particular, CNFs model their flow via a neural ordinary differential equation [11, 12, 23]. Among the most well-known CNFs are FFJORD [23], which estimates the determinant of the Jacobian by accumulating its trace along the trajectories, and the trace is estimated using Hutchinson’s estimator [4, 29, 66]. To promote straight trajectories, RNODE [19] regularizes FFJORD with a transport cost  $L(x, T)$ . RNODE also includes the Frobenius norm of the Jacobian  $\|\nabla \mathbf{v}\|_F^2$  to stabilize training. The trace and the Frobenius norm are estimated using a stochastic estimator and report speedup by a factor of 2.8.

Monge-Ampère Flows [74] and Potential Flow Generators [71] similarly draw from OT theory but parameterize a potential function instead of the dynamics directly. OT is also used in other generative

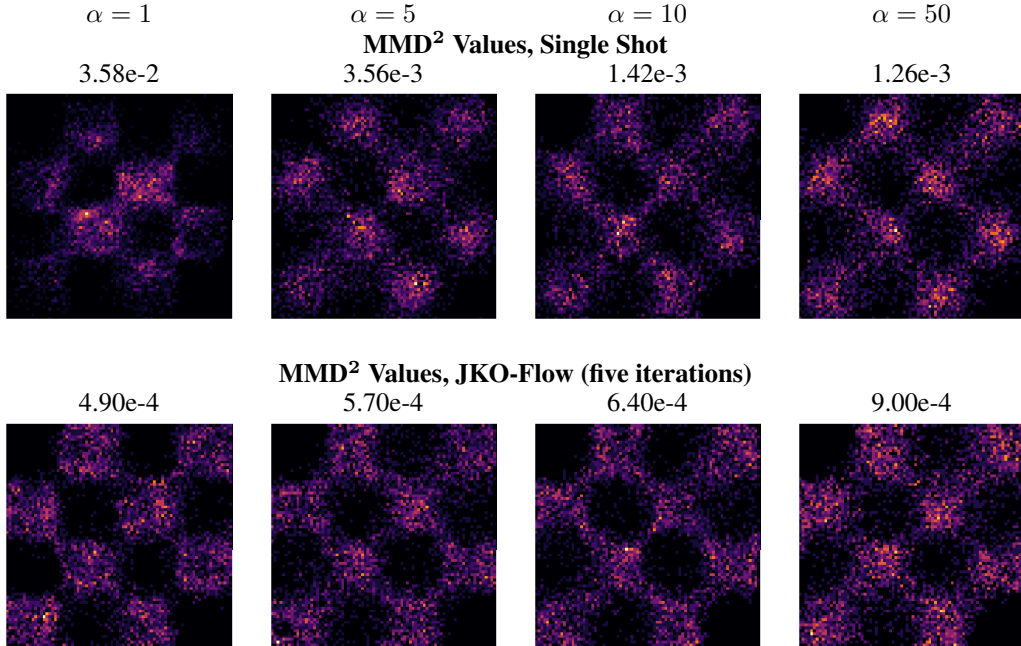


Figure 1: Checkerboard dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .

models [59, 58, 34, 36, 3, 65]. OT-Flow [45] is based on a discretize-then-optimize approach [44] that also parameterizes the potential function. To evaluate the KL divergence, OT-Flow estimates the density using an *exact* trace computation following the work of [55].

**Wasserstein Gradient Flows.** Our proposed method is most closely related to [18], which also employs a JKO-based scheme to perform generative modeling. But a key difference is that [18] reformulates the KL-divergence as an optimization over difference of expectations (See [18, Prop. 3.1]); this makes their approach akin to GANs, where the density cannot be obtained without using a separate density estimation technique. Our proposed method is also closely related to methods that use input-convex CNNs [37, 8, 2]. [37] focuses on the special case with KL divergence as objective function. [2] solve a sequence of subproblems different from the fluid flow formulation presented in (11). They also require an end-to-end training scheme that backpropagates to the initial distribution; this can become a computational burden when the number of time discretizations is large. [8] utilizes a JKO-based scheme to approximate a population dynamics given an observed trajectory and focus on applications in computational biology. Other related works include natural gradient methods [41] and implicit schemes based on the Wasserstein-1 distance [26].

## 5 Numerical Experiments

We demonstrate the effectiveness of our proposed JKO-Flow on a series of synthetic and real-world datasets. As previously mentioned, we compute each update in (10) by solving (6) using the OT-Flow solver [45], which leverages fast and exact trace computations. We also use the same architecture provided in [45]. Henceforth, we shall also call the traditional CNF approach the “single-shot” approach.

**Maximum Mean Discrepancy Metric (MMD).** Our density estimation problem requires approximating a density  $\rho_0$  by finding a transformation  $f$  such that  $f^{-1}\#\rho_1$  has density  $\hat{\rho}_0$  close to  $\rho_0$  where  $\rho_1$  is the standard Gaussian. However,  $\rho_0$  is not known in real-world density estimation scenarios, such as in physics applications, all we have are samples  $X = \{x_i\}_{i=1}^n$  from the unknown distribution. Consequently, we use the observed samples  $X$  and samples  $\hat{X} = \{\hat{x}_j\}_{j=1}^m$ ,  $\hat{x}_j = f^{-1}(q_j)$ , generated

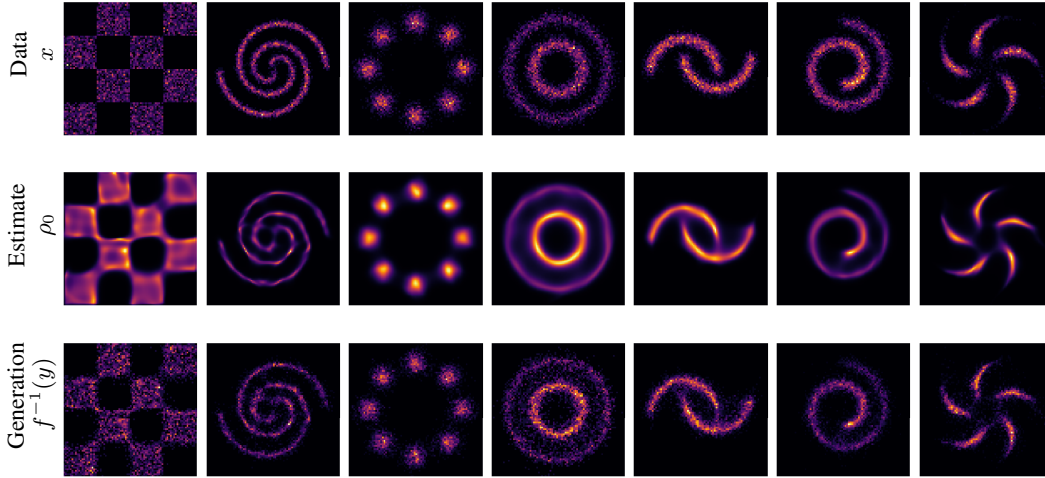


Figure 2: Density estimation on 2D toy problems using five JKO-Flow iterations. **Top:** samples from the unknown distribution  $\rho_0$ . **Middle:** density estimate for  $\rho_0$  computed by inverting the flow through the five iterations of JKO-Flow from  $\rho_1$  via (2). **Bottom:** samples generated by inverse JKO-Flow through five iterations where  $y$  has density  $\rho_1$ .

by the CNF and samples  $Q = \{q_j\}_{j=1}^m$  from  $\rho_1$  to determine if their corresponding distributions are close in some sense. To measure the discrepancy we use a particular integral probability metric [75, 51, 38] known as maximum mean discrepancy (MMD) defined as follows [24]: Let  $x$  and  $y$  be random vectors in  $\mathbb{R}^d$  with distributions  $\mu_x$  and  $\mu_y$ , respectively, and let  $\mathcal{H}$  be a reproducing kernel Hilbert space of functions on  $\mathbb{R}^d$  with Gaussian kernel (see [49] for an introduction)

$$k(x_i, x_j) = \exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right). \quad (12)$$

Then the MMD of  $\mu_x$  and  $\mu_y$  is given by

$$\text{MMD}_{\mathcal{H}}(\mu_x, \mu_y) = \sup_{\|f\|_{\mathcal{H}} \leq 1} |\mathbb{E} f(x) - \mathbb{E} f(y)|.$$

It can be shown that  $\text{MMD}_{\mathcal{H}}$  defines a metric on the class of probability measures on  $\mathbb{R}^d$  [24, 21]. The squared-MMD can be written in terms of the kernel as follows:

$$\text{MMD}_{\mathcal{H}}^2(\mu_x, \mu_y) = \mathbb{E} k(x, x') + \mathbb{E} k(y, y') - 2 \mathbb{E} k(x, y),$$

where  $x, x'$  are iid  $\mu_x$  independent of  $y, y'$  which are iid  $\mu_y$ . An unbiased estimate of the squared-MMD based on the samples  $X$  and  $\hat{X}$  defined above is given by [24]:

$$\text{MMD}_{\mathcal{H}}^2(X, \hat{X}) = \frac{1}{n(n-1)} \sum_{i \neq j} k(x_i, x_j) + \frac{1}{m(m-1)} \sum_{k \neq \ell} k(\hat{x}_k, \hat{x}_\ell) - \frac{2}{nm} \sum_{i, \ell} k(x_i, \hat{x}_\ell).$$

Note that the MMD is not used for algorithmic training of the CNF, it is only used to compare the densities  $\rho_0$  and  $\hat{\rho}_0$  based on the samples  $X$  and  $\hat{X}$ .

**Synthetic 2D Data Set.** We begin by testing our method on seven two-dimensional (2D) benchmark datasets for density estimation algorithms commonly used in machine learning [23, 69]; see Fig. 2. We generate results with JKO-Flow for different values of  $\alpha$  and for different number of iterations. We use  $\alpha = 1, 5, 10,$  and  $50,$  and for each  $\alpha$  we use the single shot approach  $k = 1$  and JKO-Flow with  $k = 5$  iterations from (10). Note that in CNFs, we are interested in estimating the density (and generating samples) from  $\rho_0$ ; consequently, once we have the optimal weights  $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(5)},$  we must “flow backwards” starting with samples from the normal distribution  $\rho_1$ . Fig. 1 shows that JKO-Flow outperforms the single shot approach for different values of  $\alpha$ . In particular, the performance for the single shot approach varies drastically for different values of  $\alpha$ , with  $\alpha = 1$  being an order of magnitude higher in MMD than  $\alpha = 5$ . On the other hand, JKO-Flow is *consistent*

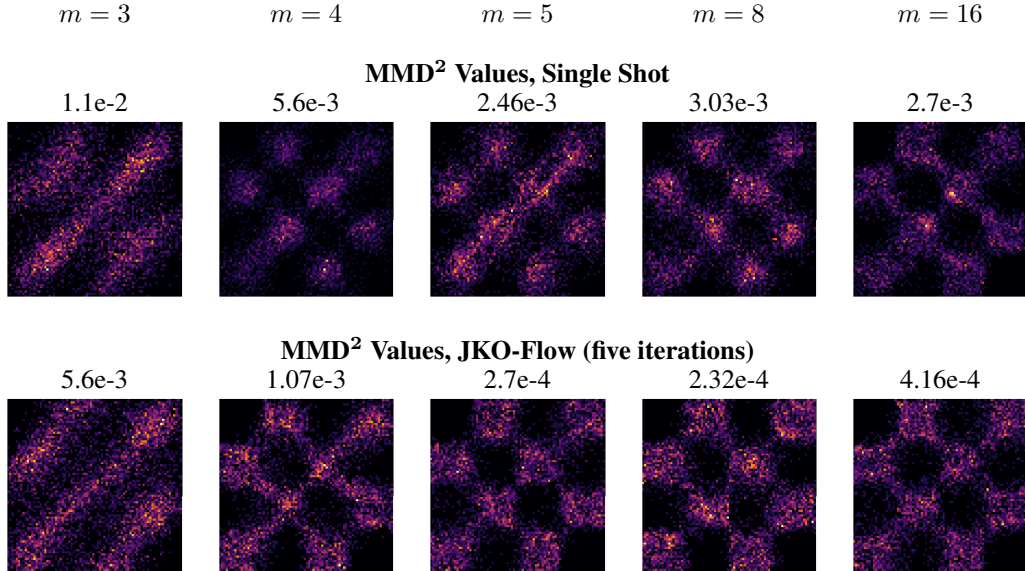


Figure 3: Checkerboard dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 5$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with fewer parameters.

regardless of the value of  $\alpha$ . As previously mentioned, this is expected as JKO-Flow is a proximal point algorithm that converges regardless of the step size  $\alpha$ . In this case, five JKO-Flow iterations are enough to obtain this consistency. Additional plots and hyperparameter setups for different benchmark datasets with similar performance results are shown in the Appendix. Tab. 1 summarizes the comparison between the single shot and JKO-Flow on all synthetic 2D datasets for different values of  $\alpha$ . We also show an illustration of all the datasets, estimated densities, and generated samples with JKO-Flow in Fig. 2.

**Varying Network Size.** In addition to obtaining consistent results for different values of  $\alpha$ , we also *empirically* observe that JKO-Flow outperforms the single shot approach for different numbers of network parameters, i.e., network size. We illustrate this in Fig. 3. This is also intuitive as we reformulate the problem of finding a single “difficult” optimal transportation problem as a sequence of “smaller and easier” OT problems. In this setup, we vary the width of a two-layer ResNet [25]. In particular, we choose the widths to be  $m = 3, 4, 5, 8,$  and  $16$ . These correspond to 40, 53, 68, 125, and 365 parameters. The hyperparameter  $\alpha$  is chosen to be the best performing value for each synthetic dataset. All datasets vary  $m$  for fixed  $\alpha = 5$ , except the 2 Spiral dataset, which uses  $\alpha = 50$ ; we chose these  $\alpha$  values as they performed the best in the fixed  $m$  experiments. Similar results are also shown for the remaining synthetic datasets in the appendix. Tab. 2 summarizes the comparison between the single shot and JKO-Flow on all synthetic 2D datasets.

**Density Estimation on a Physics Dataset** We train JKO-Flow on the 43-dimensional MINIBOONE dataset which is a high-dimensional, real-world physics dataset used as benchmark for high-dimensional density estimation algorithms in physics [53]. For this physics problem, our method is trained for  $\alpha = 0.5, 1, 5, 10, 50$  and using 10 JKO-Flow iterations. Fig 4 shows generated samples with JKO-Flow and the standard single-shot approach for  $\alpha = 5$ . Since MINIBOONE is a high-dimensional dataset, we follow [45] and plot two-dimensional slices. JKO-Flow generates better quality samples. Similar experiments for  $\alpha = 1, 10,$  and  $50$  are shown in Figs 17, 18, and 19 in the Appendix. Tab. 3 summarizes the results for all values of  $\alpha$ . Note that we compute MMD values for all the dimensions as well as 2D slices; this is because we only have limited data ( 3000 testing samples) and the 2D slice MMD give a better indication on the improvement of the generated samples. Results show that *the MMD is consistent across all  $\alpha$  values for JKO-Flow*. We also show the convergence (in MMD<sup>2</sup>) of the miniboone dataset across each 2D slice in Fig. 5. As expected,



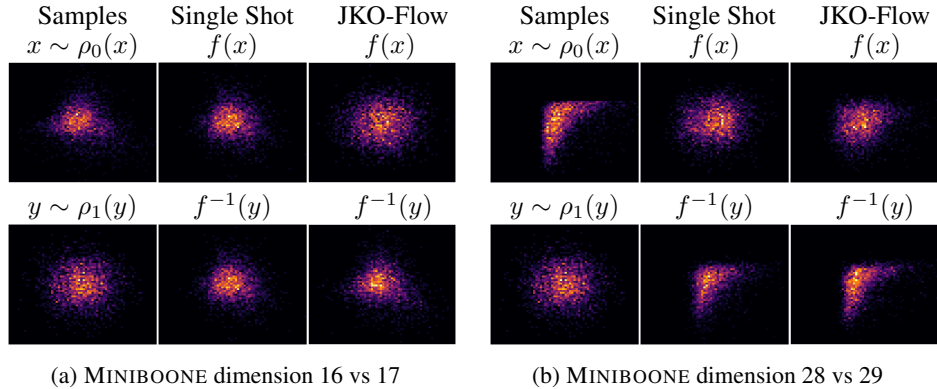


Figure 4: Generated samples for the 43-dimensional MINIBOONE dataset using the single shot approach and JKO-Flow with 10 iterations. To visualize the dataset, we show 2-dimensional slices. We show the forward flow  $f(x)$  where  $x \sim \rho_0$  and the generated samples  $f^{-1}(y)$  where  $y \sim \rho_1$ .

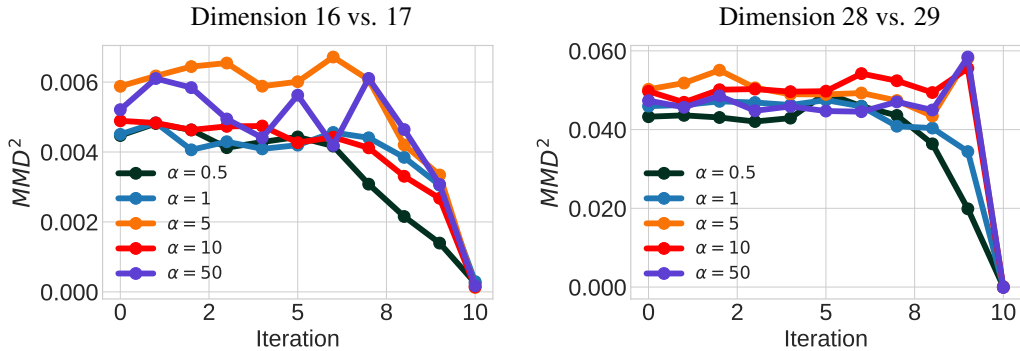


Figure 5:  $MMD^2$  per iteration when using JKO-Flow after training.  $MMD^2$  values vs. JKO-Flow iteration for 2-dimensional slice (dimensions 16-17 on the left and dimensions 28-29 on the right) of the MINIBOONE dataset. JKO-Flow achieves same accuracy *regardless* of the value of  $\alpha$ .

smaller step size  $\alpha$  values converge slower (see  $\alpha = 0.5$ ), but all converge to similar accuracy (unlike the single-shot).

## 6 Conclusion

We propose a new approach we call JKO-Flow to train OT-regularized CNFs without having to tune the regularization parameter  $\alpha$ . The key idea is to embed an underlying OT-based CNF solver into a Wasserstein gradient flow framework, also known as the JKO scheme; this approach makes the regularization parameter act as a “time” variable. Thus, instead of tuning  $\alpha$ , we repeatedly solve proximal updates for a fixed (time variable)  $\alpha$ . In our setting, we choose OT-Flow [45], which leverages exact trace estimation for fast CNF training. Our numerical experiments show that JKO-Flow leads to improved performance over the traditional approach. Moreover, *JKO-Flow achieves similar results regardless of the choice of  $\alpha$* . We also empirically observe improved performance when varying the size of the neural network. Future work will investigate JKO-Flow on similar problems such as deep learning-based methods for optimal control [20, 43, 42] and mean field games [35, 55, 1].

## Acknowledgments

LN and SO were partially funded by AFOSR MURI FA9550-18-502, ONR N00014-18-1-2527, N00014-18-20-1-2093 and N00014-20-1-2787.

Table 1: Synthetic 2D Data: JKO-Flow performance for different values of  $\alpha$ . JKO-Flow returns consistent performance for different  $\alpha$ .

$\alpha$		1	5	10	50
Dataset	Approach	MMD <sup>2</sup>			
Checkerboard	single Shot	3.58e-2	3.56e-3	1.42e-3	1.26e-3
	JKO-Flow (5 iters)	4.9e-4	5.67e-4	6.40e-4	9.00e-4
2 Spirals	Single Shot	7.21e-2	2.30e-2	1.84e-2	7.73e-4
	JKO-Flow (5 iters)	2.10e-2	4.62e-4	1.02e-4	5.37e-5
Swiss Roll	Single Shot	4.74e-3	7.33e-4	2.86e-4	7.03e-4
	JKO-Flow (5 iters)	5.16e-4	8.3e-5	3.27e-5	6.07e-4
8 Gaussians	Single Shot	9.18e-3	2.69e-4	3.94e-4	7.10e-4
	JKO-Flow (5 iters)	1.07e-4	4.13e-5	2.67e-4	7.27e-6
Circles	Single Shot	9.84e-3	2.24e-4	6.51e-4	1.04e-4
	JKO-Flow (5 iters)	9.49e-4	9.97e-6	2.38e-5	9.28e-5
Pinwheel	Single Shot	1.18e-2	1.8e-3	1.37e-3	2.2e-5
	JKO-Flow (5 iters)	4.7e-4	2.63e-4	3.84e-4	4.30e-4
Moons	Single Shot	1.45e-3	2.05e-3	2.49e-4	2.42e-4
	JKO-Flow (5 iters)	1.92e-4	4.65e-5	4.3e-5	1.08e-4

Table 2: Synthetic 2D Data: Network width comparison for 1 and 5 iterations given a fixed, best performing  $\alpha$ . JKO-Flow performs better than the single shot approach for different network sizes.

$m$		3	4	5	8	16
Dataset	Approach	MMD <sup>2</sup>				
Checkerboard	Single Shot	1.10e-2	5.60e-3	2.46e-3	3.03e-3	2.70e-3
	JKO-Flow (5 iters)	5.60e-3	1.07e-3	2.7e-4	2.32e-4	4.16e-4
2 Spirals	Single Shot	5.98e-3	4.54e-3	5.47e-3	1.19e-3	3.96e-3
	JKO-Flow (5 iters)	1.42e-3	1.49e-5	6.11e-4	3.93e-5	2.19e-3
Swiss Roll	Single Shot	8.89e-3	7.71e-3	1.41e-3	1.37e-3	1.52e-3
	JKO-Flow (5 iters)	1.49e-3	2.90e-4	6.13e-4	2.29e-4	8.40e-5
8 Gaussians	Single Shot	2.20e-3	1.05e-3	1.04e-3	2.3e-4	5.05e-4
	JKO-Flow (5 iters)	1.33e-4	9.85e-4	2.40e-5	3.96e-4	1.07e-4
Circles	Single Shot	2.06e-3	1.72e-3	1.37e-3	1.69e-3	1.34e-3
	JKO-Flow (5 iters)	1.94e-3	3.24e-4	7.71e-4	5.9e-5	1.01e-4
Pinwheel	Single Shot	1.10e-2	4.03e-3	2.27e-3	3.80e-3	5.43e-4
	JKO-Flow (5 iters)	1.20e-3	8.23e-4	1.60e-3	7.00e-5	2.69e-4
Moons	Single Shot	4.98e-3	4.54e-3	5.47e-3	1.2e-3	3.96e-3
	JKO-Flow (5 iters)	1.42e-3	1.5e-5	6.11e-4	3.90e-5	2.19e-3

Table 3: MINIBOONE: Comparison of Single Shot and JKO-Flow for different values of  $\alpha$ .

$\alpha$		0.5	1	5	10	50
Dimensions	Approach	MMD <sup>2</sup>				
2d: 16 vs. 17	Single Shot	4.62e-3	4.50e-3	5.88e-3	4.89e-3	5.2e-3
	JKO-Flow (10 iters)	3.42e-4	2.94e-4	1.27e-4	1.43e-4	1.71e-4
2d: 28 vs. 29	Single Shot	4.33e-2	4.60e-2	5.02e-2	4.97e-2	4.74e-2
	JKO-Flow (10 iters)	8.02e-5	3.31e-5	4.43e-5	6.33e-5	9.97e-5
Full	Single Shot	4.7e-2	4.51e-3	4.75e-3	4.21e-3	4.27e-3
	JKO-Flow (10 iters)	4.72e-4	4.72e-4	4.71e-4	4.72e-4	4.72e-4

## References

- [1] S. Agrawal, W. Lee, S. W. Fung, and L. Nurbekyan. Random features for high-dimensional nonlocal mean-field games. *Journal of Computational Physics*, 459:111136, 2022.
- [2] D. Alvarez-Melis, Y. Schiff, and Y. Mroueh. Optimizing functionals on the space of probabilities with input convex neural networks. *arXiv preprint arXiv:2106.00774*, 2021.
- [3] G. Avraham, Y. Zuo, and T. Drummond. Parallel optimal transport GAN. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4406–4415, 2019.
- [4] H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34, 2011.
- [5] R. Baptista, Y. Marzouk, R. E. Morrison, and O. Zahm. Learning non-Gaussian graphical models via Hessian scores and triangular transport. *arXiv preprint arXiv:2101.03093*, 2021.
- [6] J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.
- [7] J. Brehmer, F. Kling, I. Espejo, and K. Cranmer. Madminer: Machine learning-based inference for particle physics. *Computing and Software for Big Science*, 4(1):1–25, 2020.
- [8] C. Bunne, L. Papaxanthos, A. Krause, and M. Cuturi. Proximal optimal transport modeling of population dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 6511–6528. PMLR, 2022.
- [9] M. Burger, S. Osher, J. Xu, and G. Gilboa. Nonlinear inverse scale space methods for image restoration. In N. Paragios, O. Faugeras, T. Chan, and C. Schnörr, editors, *Variational, Geometric, and Level Set Methods in Computer Vision*, pages 25–36, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [10] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.
- [11] C. Chen, C. Li, L. Chen, W. Wang, Y. Pu, and L. C. Duke. Continuous-time flows for efficient inference and density estimation. In *International Conference on Machine Learning (ICML)*, pages 824–833, 2018.
- [12] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6571–6583, 2018.
- [13] O. collaboration and G. Abbiendi. Search for neutral Higgs bosons in collisions at 189 gev. *The European Physical Journal C-Particles and Fields*, 12(4):567–586, 2000.
- [14] K. Cranmer. Kernel estimation in high-energy physics. *Computer Physics Communications*, 136(3):198–207, 2001.
- [15] L. Dinh, D. Krueger, and Y. Bengio. NICE: non-linear independent components estimation. In Y. Bengio and Y. LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015.
- [16] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *International Conference on Learning Representations (ICLR)*, 2017.
- [17] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7509–7520, 2019.
- [18] J. Fan, A. Taghvaei, and Y. Chen. Variational Wasserstein gradient flow. *arXiv preprint arXiv:2112.02424*, 2021.
- [19] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *International Conference on Machine Learning (ICML)*, pages 3154–3164, 2020.

- [20] W. H. Fleming and H. M. Soner. *Controlled Markov processes and viscosity solutions*, volume 25 of *Stochastic Modelling and Applied Probability*. Springer, New York, second edition, 2006.
- [21] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel measures of conditional dependence. *Advances in Neural Information Processing Systems*, 20, 2007.
- [22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [23] W. Grathwohl, R. T. Chen, J. Betterncourt, I. Sutskever, and D. Duvenaud. FFIORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019.
- [24] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13(25):723–773, 2012.
- [25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [26] H. Heaton, S. W. Fung, A. T. Lin, S. Osher, and W. Yin. Wasserstein-based projections with applications to inverse problems. *arXiv preprint arXiv:2008.02200*, 2020.
- [27] C.-W. Huang, R. T. Chen, C. Tsirigotis, and A. Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *arXiv preprint arXiv:2012.05942*, 2020.
- [28] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning (ICML)*, pages 2078–2087, 2018.
- [29] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.
- [30] R. Jordan, D. Kinderlehrer, and F. Otto. The variational formulation of the Fokker–Planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.
- [31] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10215–10224, 2018.
- [32] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4743–4751, 2016.
- [33] I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [34] N. Lei, K. Su, L. Cui, S.-T. Yau, and X. D. Gu. A geometric view of optimal transportation and generative model. *Computer Aided Geometric Design*, 68:1–21, 2019.
- [35] A. T. Lin, S. W. Fung, W. Li, L. Nurbekyan, and S. J. Osher. Alternating the population and control neural networks to solve high-dimensional stochastic mean-field games. *Proceedings of the National Academy of Sciences*, 118(31):e2024713118, 2021.
- [36] J. Lin, K. Lensink, and E. Haber. Fluid flow mass transport for generative networks. *arXiv:1910.01694*, 2019.
- [37] P. Mokrov, A. Korotin, L. Li, A. Genevay, J. M. Solomon, and E. Burnaev. Large-scale Wasserstein gradient flows. *Advances in Neural Information Processing Systems*, 34:15243–15256, 2021.
- [38] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

- [39] R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.
- [40] F. Noé, S. Olsson, J. Köhler, and H. Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019.
- [41] L. Nurbekyan, W. Lei, and Y. Yang. Efficient natural gradient descent methods for large-scale optimization problems. *arXiv preprint arXiv:2202.06236*, 2022.
- [42] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto. A neural network approach applied to multi-agent optimal control. In *2021 European Control Conference (ECC)*, pages 1036–1041. IEEE, 2021.
- [43] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto. A neural network approach for high-dimensional optimal control applied to multiagent path finding. *IEEE Transactions on Control Systems Technology*, 2022.
- [44] D. Onken and L. Ruthotto. Discretize-optimize vs. optimize-discretize for time-series regression and continuous normalizing flows. *arXiv:2005.13420*, 2020.
- [45] D. Onken, S. Wu Fung, X. Li, and L. Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.
- [46] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.
- [47] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv:1912.02762*, 2019.
- [48] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2338–2347, 2017.
- [49] V. I. Paulsen and M. Raghupathi. *An introduction to the theory of reproducing kernel Hilbert spaces*, volume 152. Cambridge University Press, 2016.
- [50] G. Peyré and M. Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.
- [51] S. T. Rachev, L. B. Klebanov, S. V. Stoyanov, and F. Fabozzi. *The Methods of Distances in the Theory of Probability and Statistics*, volume 10. Springer, 2013.
- [52] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015.
- [53] B. Roe. MiniBooNE particle identification. UCI Machine Learning Repository, 2010.
- [54] L. Ruthotto and E. Haber. An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021.
- [55] L. Ruthotto, S. J. Osher, W. Li, L. Nurbekyan, and S. W. Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193, 2020.
- [56] A. Salim, A. Korba, and G. Luise. The Wasserstein proximal gradient algorithm. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12356–12366. Curran Associates, Inc., 2020.
- [57] T. Salimans, D. Kingma, and M. Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning (ICML)*, pages 1218–1226, 2015.
- [58] T. Salimans, H. Zhang, A. Radford, and D. N. Metaxas. Improving GANs using optimal transport. In *International Conference on Learning Representations (ICLR)*, 2018.

- [59] M. Sanjabi, J. Ba, M. Razaviyayn, and J. D. Lee. On the convergence and robustness of training gans with regularized optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7091–7101, 2018.
- [60] F. Santambrogio. *Optimal Transport for Applied Mathematicians*, volume 87 of *Progress in Nonlinear Differential Equations and their Applications*. Birkhäuser/Springer, Cham, 2015. Calculus of variations, PDEs, and modeling.
- [61] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.
- [62] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. CRC Press, 1986.
- [63] J. Suykens, H. Verrelst, and J. Vandewalle. On-line learning Fokker-Planck machine. *Neural Processing Letters*, 7:81–89, 04 1998.
- [64] E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.
- [65] A. Tanaka. Discriminator optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6816–6826, 2019.
- [66] L. Tenorio. *An Introduction to Data Analysis and Uncertainty Quantification for Inverse Problems*. SIAM, 2017.
- [67] C. Villani. *Topics in optimal transportation*, volume 58 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2003.
- [68] C. Villani. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.
- [69] A. Wehenkel and G. Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1543–1553, 2019.
- [70] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning (ICML)*, pages 681–688, 2011.
- [71] L. Yang and G. E. Karniadakis. Potential flow generator with  $L_2$  optimal transport regularity for generative models. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [72] L. Yang and G. E. Karniadakis. Potential flow generator with  $l_2$  optimal transport regularity for generative models. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [73] J. Zech and Y. Marzouk. Sparse approximation of triangular transports, part II: The infinite-dimensional case. *Constructive Approximation*, 55(3):987–1036, 2022.
- [74] L. Zhang, W. E, and L. Wang. Monge-Ampère flow for generative modeling. *arXiv:1809.10188*, 2018.
- [75] V. M. Zolotarev. Metric distances in spaces of random variables and their distributions. *Mathematics of the USSR-Sbornik*, 30(3):373, 1976.

## A Appendix

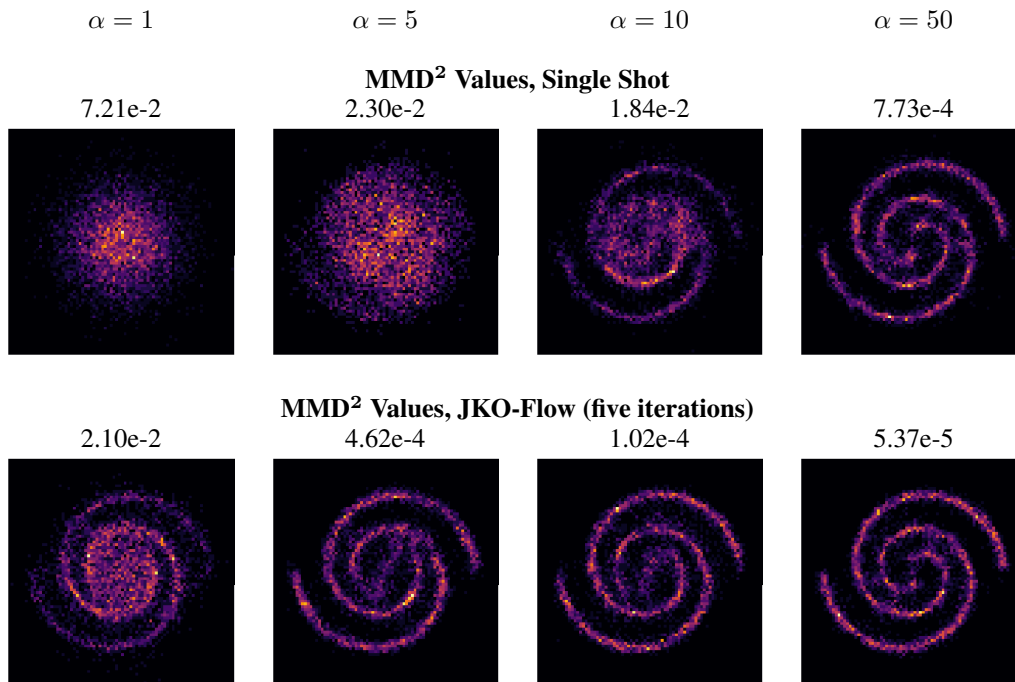


Figure 6: 2 Spirals dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .

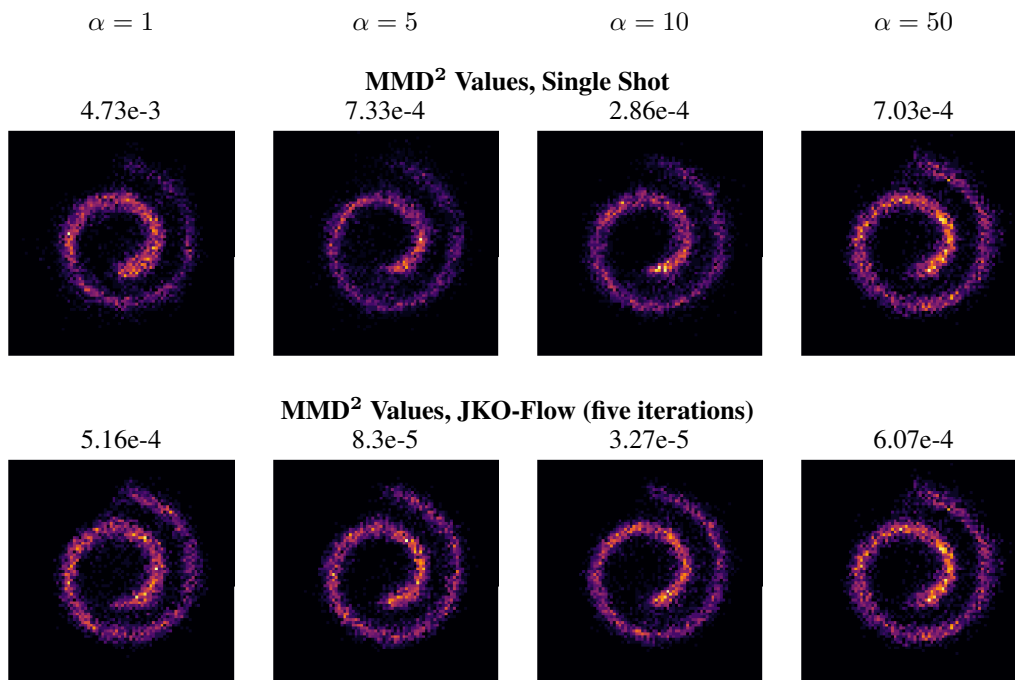


Figure 7: Swiss Roll dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .

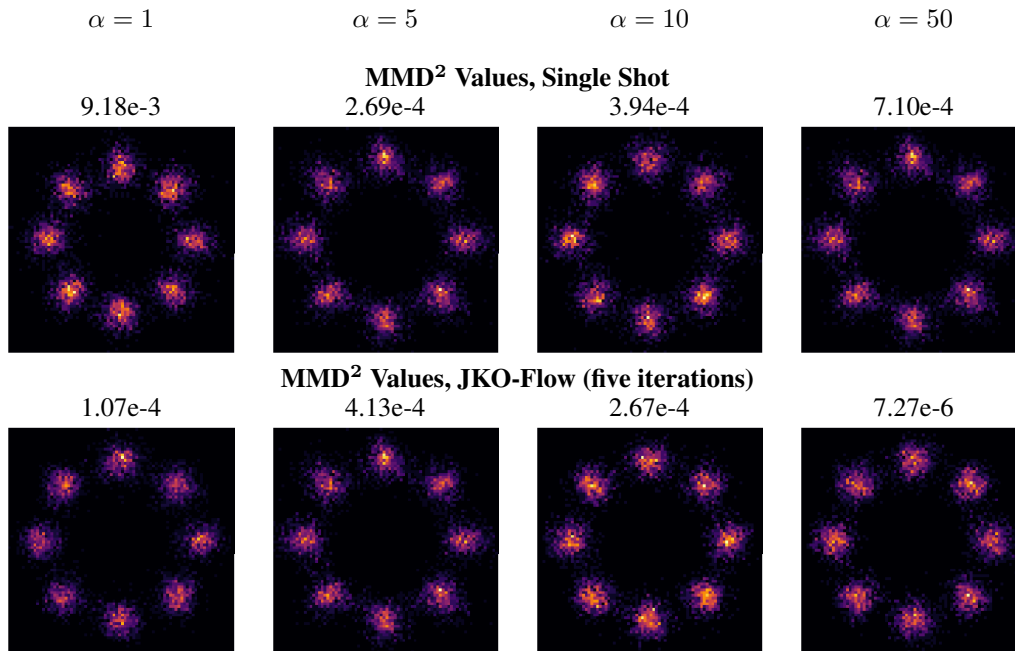


Figure 8: 8 Gaussians dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .

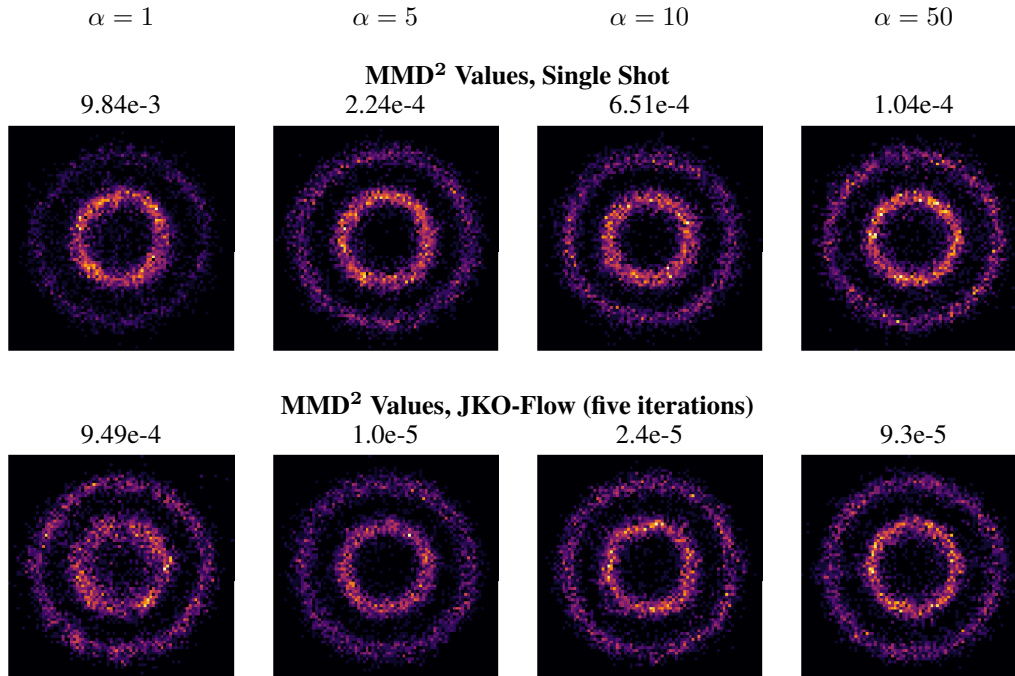


Figure 9: Circles dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .



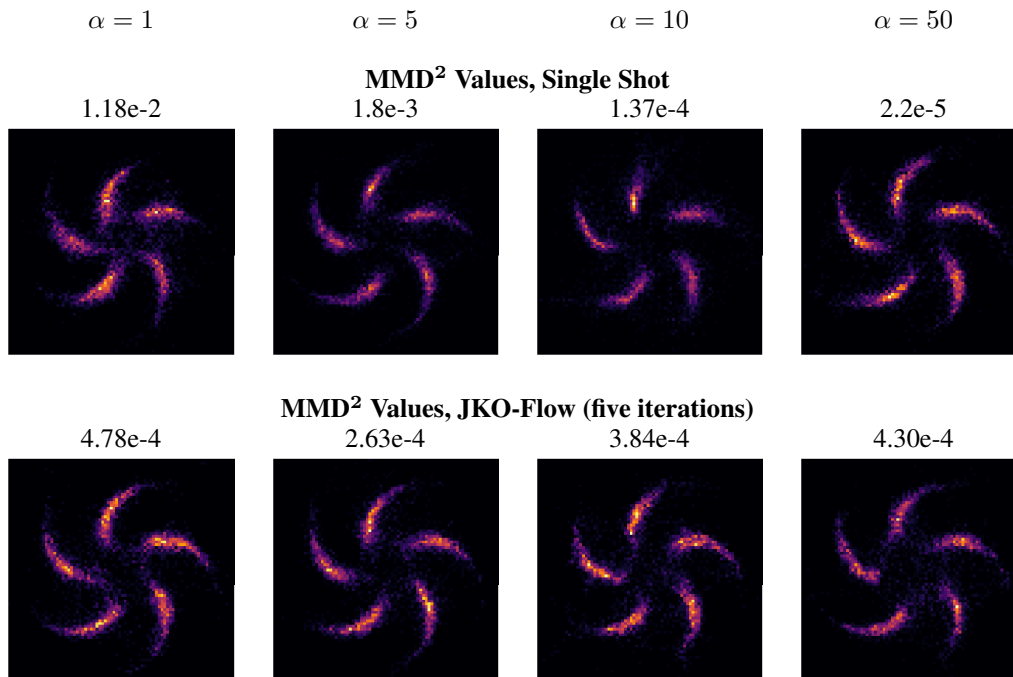


Table 4: Pinwheel dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .

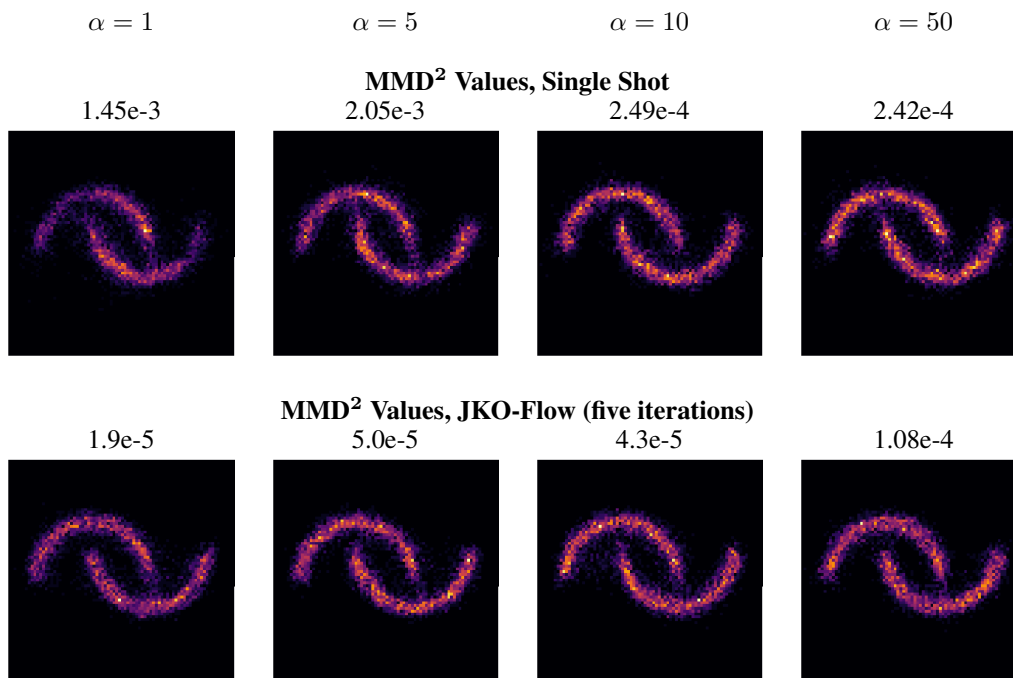


Figure 10: Moons dataset: Generated samples of  $\hat{\rho}_0$  using the standard one-shot approach (top row). Generated using our proposed JKO-Flow using five iterations (bottom row). Here, we use  $\alpha = 1, 5, 10, 50$ . JKO-Flow returns consistent results *regardless of the value of  $\alpha$* .

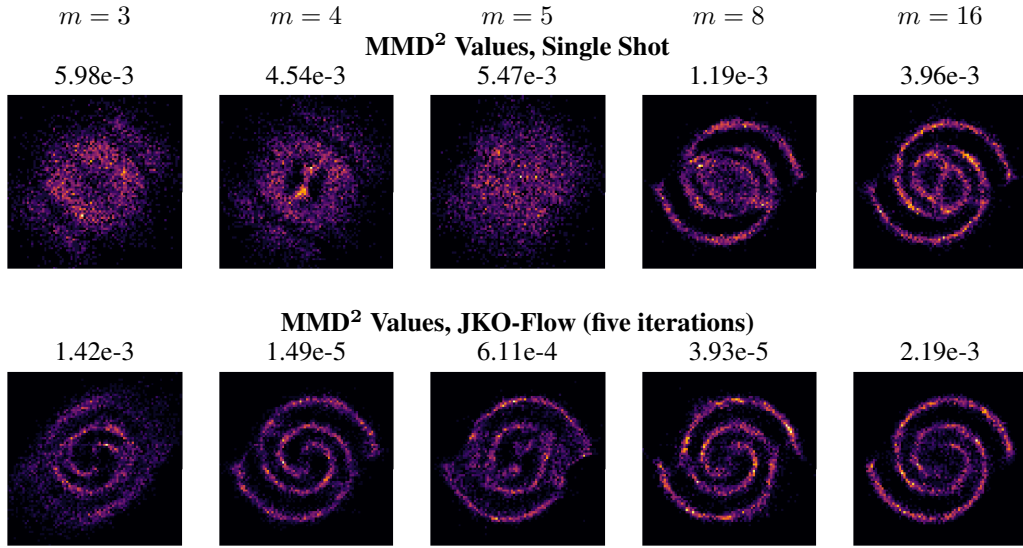


Figure 11: 2 Spirals dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 50$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with lower number of parameters.

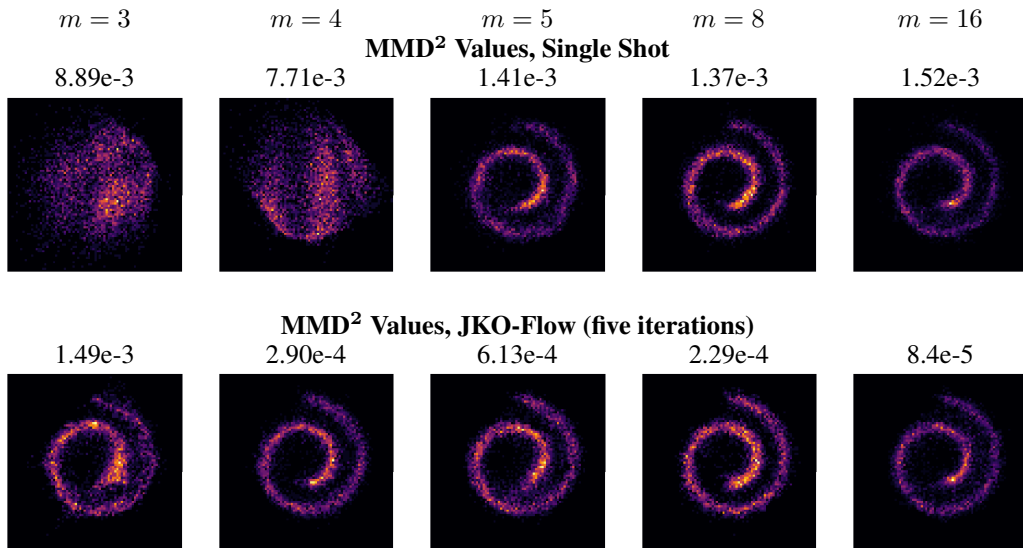


Figure 12: Swiss Roll dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 5$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with lower number of parameters.

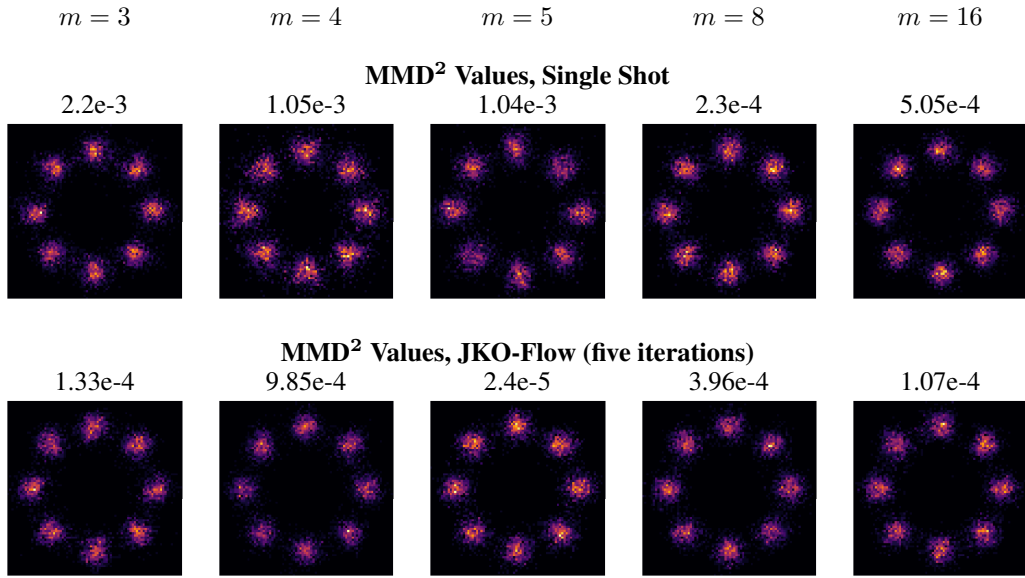


Figure 13: 8 Gaussians dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 5$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with lower number of parameters.

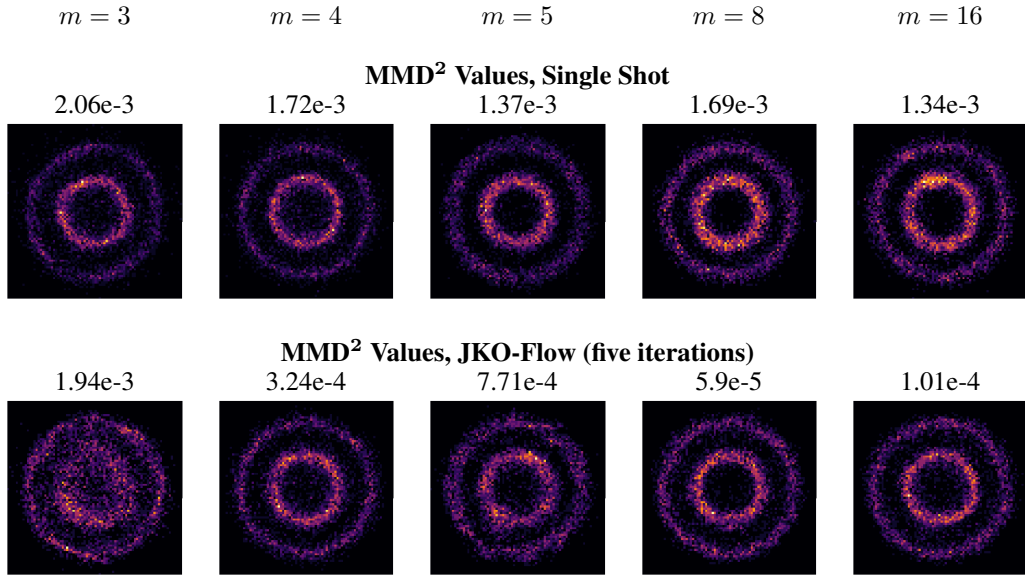


Figure 14: Circles dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 5$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with lower number of parameters.

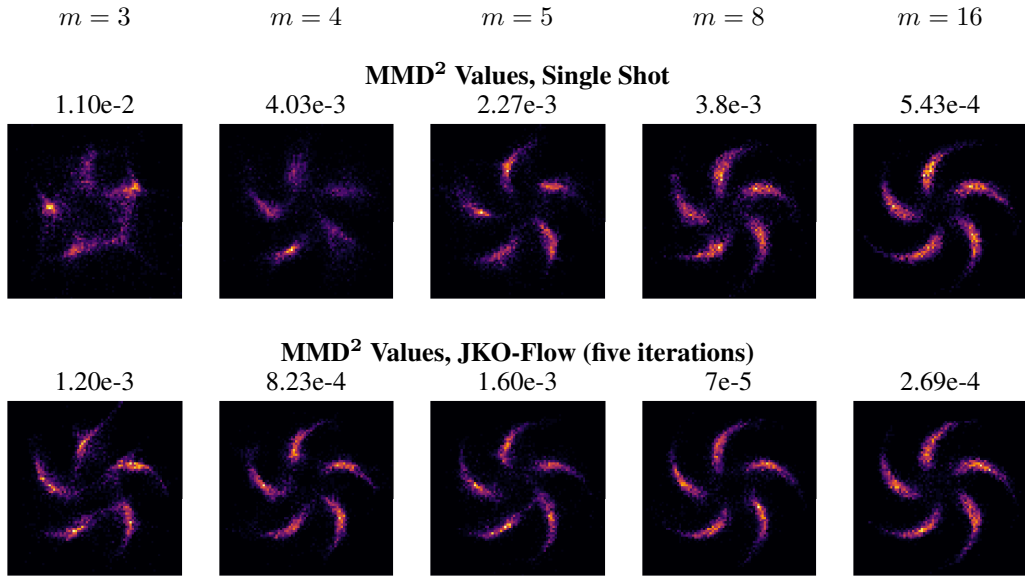


Figure 15: Pinwheel dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 5$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with lower number of parameters.

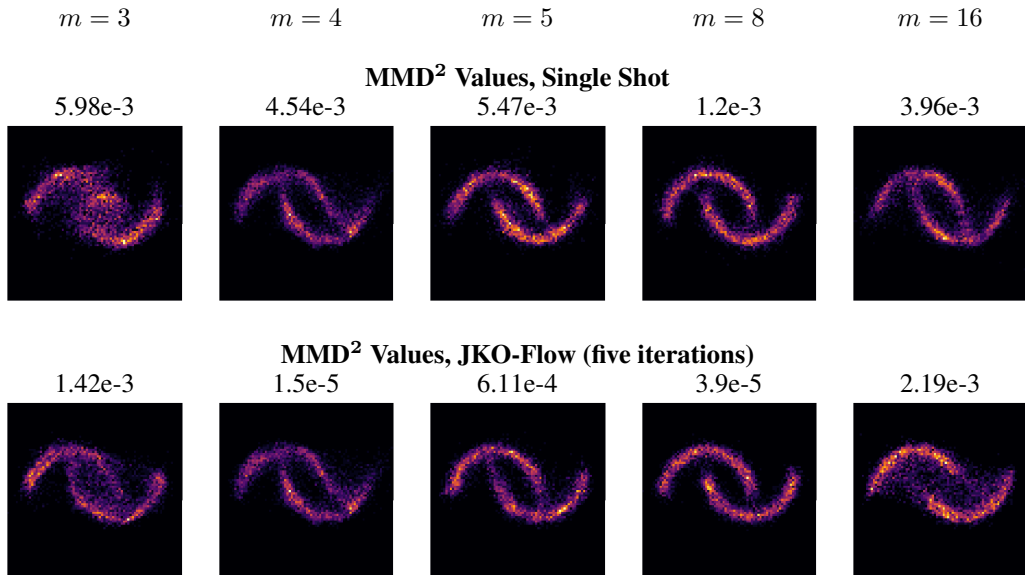


Figure 16: Moons dataset: Generated samples of  $\hat{\rho}_0$  using the standard single shot approach (top row). Generated samples using our proposed JKO-Flow using five iterations (bottom row). Here, we fix  $\alpha = 5$  and vary the network width  $m = 3, 4, 5, 8,$  and  $16$ . JKO-Flow performs competitively even with lower number of parameters.

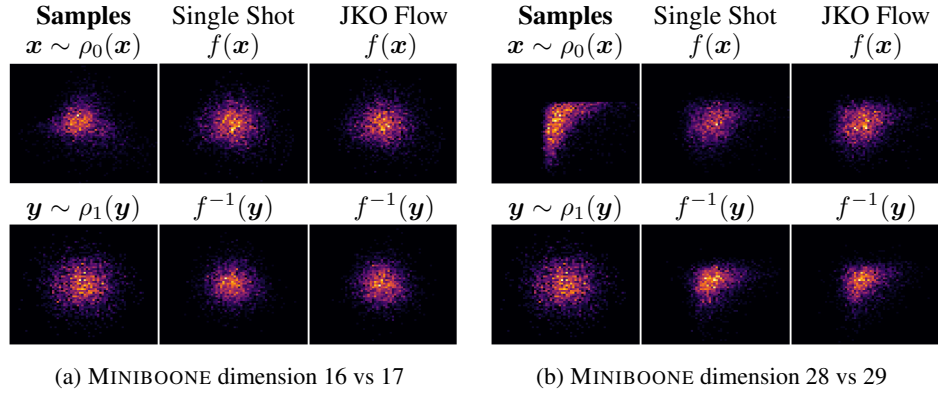


Figure 17: Generated samples for the 43-dimensional MINIBOONE dataset using the single shot approach and JKO-Flow with 10 iterations for  $\alpha = 1$ . To visualize the dataset, we show 2-dimensional slices. We show the forward flow  $f(x)$  where  $x \sim \rho_0$  and the generated samples  $f^{-1}(y)$  where  $y \sim \rho_1$ .

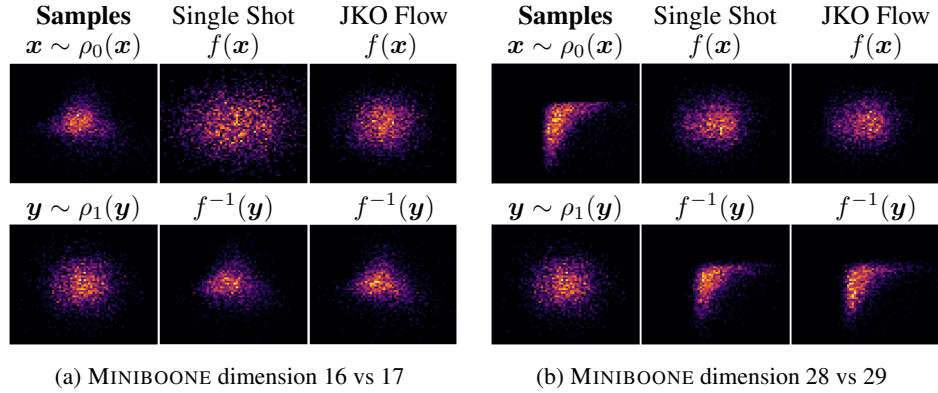


Figure 18: Generated samples for the 43-dimensional MINIBOONE dataset using the single shot approach and JKO-Flow with 10 iterations for  $\alpha = 10$ . To visualize the dataset, we show 2-dimensional slices. We show the forward flow  $f(x)$  where  $x \sim \rho_0$  and the generated samples  $f^{-1}(y)$  where  $y \sim \rho_1$ .

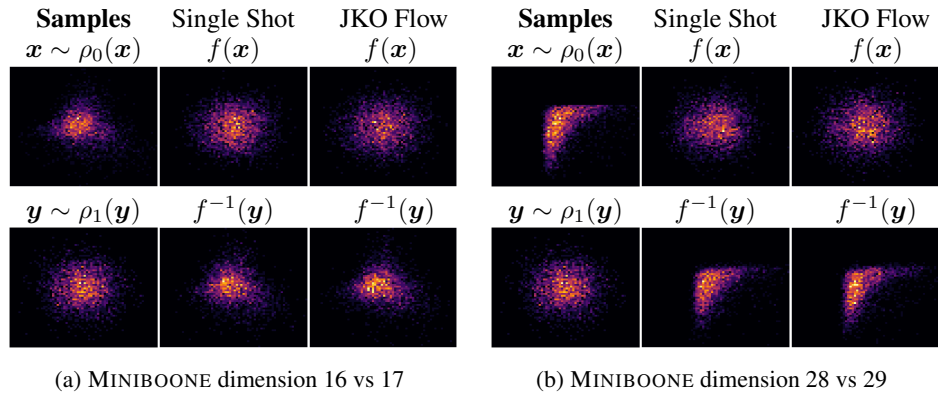


Figure 19: Generated samples for the 43-dimensional MINIBOONE dataset using the single shot approach and JKO-Flow with 10 iterations for  $\alpha = 50$ . To visualize the dataset, we show 2-dimensional slices. We show the forward flow  $f(x)$  where  $x \sim \rho_0$  and the generated samples  $f^{-1}(y)$  where  $y \sim \rho_1$ .