Computing high-dimensional optimal transport by flow neural networks

Chen Xu
School of Industrial and Systems Engineering
Georgia Tech

Xiuyuan Cheng
Department of Mathematics
Duke University

Yao Xie
School of Industrial and Systems Engineering
Georgia Tech

Abstract

Flow-based models are widely used in generative tasks, including normalizing flow, where a neural network transports from a data distribution P to a normal distribution. This work develops a flow-based model that transports from P to an arbitrary Q where both distributions are only accessible via finite samples. We propose to learn the dynamic optimal transport between P and Q by training a flow neural network. The model is trained to find an invertible transport map between P and Q optimally by minimizing the transport cost. The trained optimal transport flow allows for performing many downstream tasks, including infinitesimal density ratio estimation and distribution interpolation in the latent space for generative models. The effectiveness of the proposed model on high-dimensional data is empirically demonstrated in mutual information estimation, energy-based generative models, and image-to-image translation.

1 Introduction

The problem of finding a transport map between two general distributions P and Q in high dimension is essential in statistics, optimization, and machine learning. When both distributions are only accessible via finite samples, the transport map needs to be learned from data. In spite of the modeling and computational challenges, this setting has applications in many fields. For example, transfer learning in domain adaption aims to obtain a model on the target domain at a lower cost by using an existing pre-trained model on the source domain (Courty et al., 2014, 2017), and this can be achieved by transporting the source domain samples to the target domain using the transport map. The transport map can also be used to provide intermediate interpolating distributions between P and Q. In density ratio estimation (DRE), this bridging facilitates the so-called "telescopic" DRE (Rhodes et al., 2020) which has been shown to be more accurate when P and Q significantly differ. Furthermore, learning such a transport map between two sets of images can facilitate solving problems in computer vision, such as image restoration and image-to-image translation (Isola et al., 2017).

This work focuses on a continuous-time formulation of the problem where we are to find an invertible transport map $T_t : \mathbb{R}^d \to \mathbb{R}^d$ continuously parametrized by time $t \in [0, 1]$ and satisfying that $T_0 = \mathrm{Id}$ and $(T_1)_\# P = Q$. Here we denote by $T_\# P$ the push-forward of distribution P by a mapping T, such that $(T_\# P)(\cdot) = P(T^{-1}(\cdot))$. Suppose P and Q have densities p and q respectively in $\mathbb{R}^d$ (we also use the push-forward notation $_\#$ on densities), the transport map T_t defines

$$\rho(x, t) := (T_t)_\# p, \quad \text{s.t.} \quad \rho(x, 0) = p, \quad \rho(x, 1) = q.$$

We will adopt the neural Ordinary Differential Equation (ODE) approach Chen et al. (2018) where we represent T_t as the solution map of an ODE, which is further parametrized by a continuous-time residual network. The resulting map T_t is invertible, and the inversion can be computed by integrating the neural ODE reverse in time. Our model learns the flow from two sets of finite samples

Accepted to the Optimal Transport and Machine Learning Workshop at NeurIPS 2023.

from P and Q. The velocity field in the neural ODE will be optimized to minimize the transport cost so as to approximate the optimal velocity in dynamic optimal transport (OT) formulation, i.e. Benamou-Brenier equation.

The neural-ODE model has been intensively developed in Continuous Normalizing Flows (CNF) Kobyzev et al. (2020). In CNF, the continuous-time flow model, usually parametrized by a neural ODE, transports from a data distribution P (accessible via finite samples) to a terminal analytical distribution which is typically the normal one $\mathcal{N}(0, I_d)$, per the name "normalizing". The study of normalizing flow dated back to non-deep models with statistical applications (Tabak & Vanden-Eijnden, 2010), and deep CNFs have recently developed into a popular tool for generative models and likelihood inference of high dimensional data. CNF models rely on the analytical expression of the terminal distribution in training. Since our model is also a flow model that transports from data distribution P to a general (unknown) data distribution Q, both accessible via empirical samples, we name our model "Q-flow" which is inspired by the CNF literature.

In summary, the contributions of the work include:

- We develop a flow-based model *Q-flow* net to learn a continuous invertible optimal transport map between arbitrary pair of distributions P and Q in $\mathbb{R}^d$ from two sets of samples of the distributions. We propose to train a neural ODE model to minimize the transport cost such that the flow approximates the optimal transport in dynamic OT. The end-to-end training of the model refines an initial flow that may not attain the optimal transport, e.g., obtained by training two CNFs or other interpolating schemes.
- Leveraging the trained optimal transport Q-flow net, we propose a new DRE approach by training a separate continuous-time neural network using classification losses along the time grid. The proposed DRE method improves the performance in high dimension, demonstrated by high-dimensional mutual information estimation and energy-based generative models.
- We show the effectiveness of the approach on simulated and real data. On the image-to-image translation task, our Q-flow gradually transforms an input image to a target one that resembles in style and achieves competitive quantitative metrics against the baselines.

1.1 Related works

Normalizing flows. When the target distribution Q is an isotropic Gaussian $\mathcal{N}(0, I_d)$, normalizing flow models have demonstrated vast empirical successes in building an invertible transport T_t between P and $\mathcal{N}(0, I_d)$ (Kobyzev et al., 2020). The transport is parametrized by deep neural networks, whose parameters are trained via minimizing the KL-divergence between transported distribution $(T_1)_{\#}P$ and $\mathcal{N}(0, I_d)$. Various continuous (Grathwohl et al., 2019; Finlay et al., 2020) and discrete (Dinh et al., 2016; Behrmann et al., 2019) normalizing flow models have been developed, along with proposed regularization techniques (Onken et al., 2021; Xu et al., 2022a,b) that facilitate the training of such models in practice. Since our Q-flow is in essence a transport-regularized flow between P and Q, we further review related works on building normalizing flow models with transport regularization. (Finlay et al., 2020) trained the flow trajectory with regularization based on ℓ_2 transport cost and Jacobian norm of the network-parametrized velocity field. (Onken et al., 2021) proposed to regularize the flow trajectory by ℓ_2 transport cost and the deviation from the HJB equation. These regularization have shown to effectively improve over un-regularized models at a reduced computational cost. Regularized normalizing flow models have also been used to solve high dimensional Fokker-Planck equations (Liu et al., 2022) and mean-field games (Huang et al., 2023).

Distribution interpolation by neural networks. Recently, there have been several works establishing a continuous-time interpolation between general high-dimensional distributions. (Albergo & Vanden-Eijnden, 2023) proposed to use a stochastic interpolant map between two arbitrary distributions and train a neural network parametrized velocity field to transport the distribution along the interpolated trajectory. (Neklyudov et al., 2023) proposed an action matching scheme that leverages a pre-specified trajectory between P and Q to learn the OT map between two *infinitesimally close* distributions along the trajectory. (Liu, 2022) proposed rectified flow which starts from an initial coupling of P and Q and iteratively rectifies it to converge to the optimal coupling. Same as in (Albergo & Vanden-Eijnden, 2023; Neklyudov et al., 2023; Lipman et al., 2023), our neural-ODE based approach also computes a deterministic probability transport map, in contrast to SDE-based diffusion models (Song et al., 2021). Notably, the interpolant mapping used in these prior works

is generally not the optimal transport interpolation. In comparison, our proposed Q-flow optimizes the interpolant mapping parametrized by a neural ODE and approximates the optimal velocity in dynamic OT (see Section 2). Generally, the flow attaining optimal transport can lead to improved model efficiency and generalization performance Huang et al. (2023). In this work, the proposed method aims to solve the dynamic OT trajectory by a flow network, and we experimentally show that the optimal transport flow benefits high-dimensional DRE and image-to-image translation.

Optimal transport between general distributions. The problem of OT dates back to the work by Gaspard Monge (Monge, 1781), and since then many mathematical theories and computational tools have been developed to tackle the question (Villani et al., 2009; Benamou & Brenier, 2000; Peyré et al., 2019). Several works have attempted to make computational OT scalable to high dimensions, including (Lavenant et al., 2018) which applied convex optimization using Riemannian structure of the space of discrete probability distributions, and (Lee et al., 2021) by L^1 and L^2 versions of the generalized unnormalized OT solved by Nesterov acceleration. Several deep approaches have also been developed recently. (Coeurdoux et al., 2023) leveraged normalizing flow to learn an approximate transport map between two distributions from finite samples, where the flow model has a restricted architecture and the OT constraint is replaced with sliced-Wasserstein distance which may not computationally scale to high dimensional data. Several works have also considered casting the optimal transport problem into a minimax problem based on either the Kantorovich formulation of (Xie et al., 2019; Korotin et al., 2023) or the Monge formulation (Fan et al., 2022). In comparison, our approach computes the continuous-time dynamic OT mapping parametrized by the optimal velocity field, which directly provides a continuous interpolation between two distributions and can be applied to tasks like DRE.

2 Preliminaries

Neural ODE and CNF. Neural ODE Chen et al. (2018) parametrized an ODE in $\mathbb{R}^d$ by a residual network. Specifically, let $x(t)$ be the solution of

$$\dot{x}(t) = f(x(t), t; \theta), \quad x(0) \sim p. \tag{1}$$

where $f(x, t; \theta)$ is a velocity field parametrized by the neural network. Since we impose a distribution P on the initial value $x(0)$, the value of $x(t)$ at any t also observes a distribution $p(x, t)$ (though $x(t)$ is deterministic given $x(0)$). In other words, $p(\cdot, t) = (T_t)_{\#}p$, where T_t is the solution map of the ODE, namely $T_t(x) = x + \int_0^t f(x(s), s; \theta)ds$, $x(0) = x$. In the context of CNF (Kobyzev et al., 2020), the training of the flow network $f(x, t; \theta)$ is to minimize the KL divergence between the terminal density $p(x, T)$ at some T and a target density p_Z which is the normal distribution. The computation of the objective relies on the expression of normal density and can be estimated on finite samples of $x(0)$ drawn from p.

Dynamic OT (Benamou-Brenier). The Benamou-Brenier equation below provides the dynamic formulation of OT Villani et al. (2009); Benamou & Brenier (2000)

$$\begin{aligned} &\inf_{\rho, v} \mathcal{T} := \int_0^1 \mathbb{E}_{x(t)\sim\rho(\cdot,t)} \|v(x(t), t)\|^2 dt \\ &s.t. \quad \partial_t \rho + \nabla \cdot (\rho v) = 0, \quad \rho(x, 0) = p(x), \quad \rho(x, 1) = q(x), \end{aligned} \tag{2}$$

where $v(x, t)$ is a velocity field and $\rho(x, t)$ is the probability mass at time t satisfying the continuity equation with v. The action $\mathcal{T}$ is the transport cost. Under regularity conditions of p, q, the minimum $\mathcal{T}$ in (2) equals the squared Wasserstein-2 distance between p and q, and the minimizer $v(x, t)$ can be interpreted as the optimal control of the transport problem.

3 Learning dynamic OT by Q-flow network

We introduce the formulation and training objective of the proposed OT Q-flow net in Section 3.1. The training technique consists of the end-to-end training (Section 3.2) and the initial flow construction (Section 3.3).

3.1 Formulation and training objective

Given two sets of samples $\boldsymbol{X} = \{X_i\}_{i=1}^N$ and $\tilde{\boldsymbol{X}} = \{\tilde{X}_j\}_{j=1}^M$, where $X_i \sim P$ and $\tilde{X}_j \sim Q$ i.i.d., we train a neural ODE model $f(x, t; \theta)$ (1) to represent the transport map T_t. The formulation is symmetric from P to Q and vice versa, and the loss will also have symmetrically two parts. We call $P \to Q$ the forward direction and $Q \to P$ the reverse direction.

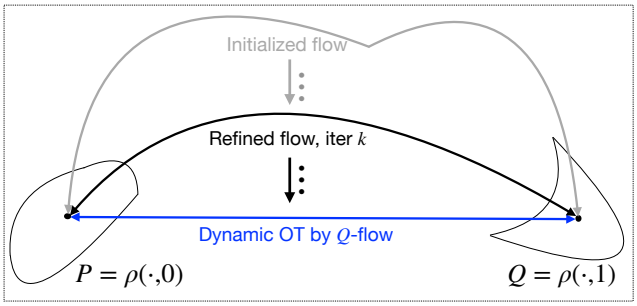


Figure 1: Illustration of learning the dynamic OT using our Q-flow (blue), which invertibly transports between P and Q over the interval $[0, 1]$ with the least transport cost. Taking any initial flow (grey) between P and Q, we iteratively refine flow trajectories to obtain flows with smaller transport cost (black), converging gradually to the dynamic OT between these two distributions.

Our training objective is based on the dynamic OT (2) on time $[0, 1]$, where we solve the velocity field $v(x, t)$ by $f(x, t; \theta)$. The terminal condition $\rho(\cdot, 1) = q$ is relaxed by a KL divergence (see, e.g., (Ruthotto et al., 2020)). The training loss in forward direction is written as

$$\mathcal{L}^{P\to Q} = \mathcal{L}_{\mathrm{KL}}^{P\to Q} + \gamma \mathcal{L}_T^{P\to Q}, \tag{3}$$

where $\mathcal{L}_{\mathrm{KL}}$ represents the relaxed terminal condition and $\mathcal{L}_T$ is the Wasserstein-2 transport cost to be specified below; $\gamma > 0$ is a weight parameter, and with small γ the terminal condition is enforced.

KL loss. Now we specify the first term in the loss (3) $\mathcal{L}_{\mathrm{KL}}^{P\to Q}$. We define the solution mapping of (1) from s to t as

$$T_s^t(x; \theta) = x(s) + \int_s^t f(x(t'), t'; \theta) dt', \tag{4}$$

which is also parametrized by θ, and we may omit the dependence below. By the continuity equation in (2), $\rho(\cdot, t) = (T_0^t)_\# p$. The terminal condition $\rho(\cdot, 1) = q$ is relaxed by minimizing $\mathrm{KL}(p_1||q) = \mathbb{E}_{x\sim p_1} \log(p_1(x)/q(x)), \quad p_1 := (T_0^1)_\# p$. The expectation $\mathbb{E}_{x\sim p_1}$ is estimated by the sample average over $(X_1)_i$ which observes density p_1 i.i.d., where $(X_1)_i := T_0^1(X_i)$ is computed by integrating the neural ODE from time 0 to 1.

It remains to have an estimator of $\log(p_1/q)$ to compute $\mathrm{KL}(p_1||q)$, and we propose to train a logistic classification network $r_1(x; \varphi_r)$ with parameters φ_r for this. The inner-loop training of r_1 is by

$$\min_{\varphi_r} \frac{1}{N} \sum_{i=1}^N \log(1 + e^{r_1(T_0^1(X_i;\theta);\varphi_r)}) + \frac{1}{M} \sum_{j=1}^M \log(1 + e^{-r_1(\tilde{X}_j;\varphi_r)}). \tag{5}$$

The functional optimal r_1^* of the population version of loss (5) equals $\log(q/p_1)$ by direct computation, and as a result, $\mathrm{KL}(p_1||q) = -\mathbb{E}_{x\sim p_1} r_1^*(x)$. Now take the trained classification network r_1 with parameter $\hat{\varphi}_r$, we can estimate the finite sample KL loss as

$$\mathcal{L}_{\mathrm{KL}}^{P\to Q}(\theta) = -\frac{1}{N} \sum_{i=1}^N r_1(T_0^1(X_i; \theta); \hat{\varphi}_r), \tag{6}$$

where $\hat{\varphi}_r$ is the computed minimizer of (5) solved by inner loops. In practice, when the density p_1 is close to q, the DRE by training classification net r_1 can be efficient and accurate. We will apply the minimization (5) after the flow net is properly initialized which guarantees the closeness of $p_1 = (T_0^1)_\# p$ and q to begin with.

W_2 regularization. Now we specify the second term in the loss (3) that defines the Wasserstein-2 regularization. To compute the transport cost $\mathcal{T}$ in (2) with velocity field $f(x, t; \theta)$, we use a time grid on $[0, 1]$ as $0 = t_0 < t_1 < \ldots < t_K = 1$. The choice of the time grid is algorithmic (since the flow model is parametrized by θ throughout time) and may vary over experiments, see more details in Section 3.2. Define $h_k = t_k - t_{k-1}$, and $X_i(t; \theta) := T_0^t(X_i; \theta)$, the W_2 regularization is written as

$$\mathcal{L}_T^{P\to Q}(\theta) = \sum_{k=1}^K \frac{1}{h_k} \left(\frac{1}{N} \sum_{i=1}^N \|X_i(t_k; \theta) - X_i(t_{k-1}; \theta)\|^2 \right). \tag{7}$$

It can be viewed as a time discretization of $\mathcal{T}$. Meanwhile, since (omitting dependence on θ) $X_i(t_k) - X_i(t_{k-1}) = T_{t_{k-1}}^{t_k}(X_i(t_{k-1}))$, the population form of (7) $\sum_{k=1}^K \mathbb{E}_{x\sim\rho(\cdot,t_{k-1})}\|T_{t_{k-1}}^{t_k}(x;\theta)\|^2/h_k$ in minimization can be interpreted as the discrete-time summed (square) Wasserstein-2 distance (Xu et al., 2022a) $\sum_{k=1}^K W_2(\rho(\cdot,t_{k-1}),\rho(\cdot,t_k))^2/h_k$. The W_2 regularization encourages a smooth flow from P to Q with small transport cost, which also guarantees the invertibility of the model in practice when the trained neural network flow approximates the optimal flow in (2).

Flow in both directions. To improve the numerical accuracy, we will design a training scheme that will take into account flow in both directions, T_0^1 and T_1^0; note that these transport maps are related to each other through (4). The formulation in the reverse direction is similar, where we transport Q-samples $\tilde{X}_i$ from 1 to 0 using the same neural ODE integrated in reverse time. Specifically, $\mathcal{L}^{Q\to P} = \mathcal{L}_{\mathrm{KL}}^{Q\to P} + \gamma\mathcal{L}_T^{Q\to P}$, and $\mathcal{L}_{\mathrm{KL}}^{Q\to P}(\theta) = -\frac{1}{M}\sum_{j=1}^M \tilde{r}_0(T_1^0(\tilde{X}_j;\theta);\hat{\varphi}_{\tilde{r}})$, where $\hat{\varphi}_{\tilde{r}}$ is obtained by inner-loop training of another classification net $\tilde{r}_0(x,\varphi_{\tilde{r}})$ with parameters $\varphi_{\tilde{r}}$ via

$$\min_{\varphi_{\tilde{r}}} \frac{1}{M}\sum_{j=1}^M \log(1+e^{\tilde{r}_0(T_1^0(\tilde{X}_j;\theta);\varphi_{\tilde{r}})}) + \frac{1}{N}\sum_{i=1}^N \log(1+e^{-\tilde{r}_0(X_i;\varphi_{\tilde{r}})}); \tag{8}$$

Define $\tilde{X}_j(t;\theta) := T_1^t(\tilde{X}_j;\theta)$, the reverse-time W_2 regularization can be obtained as $\mathcal{L}_T^{Q\to P}(\theta) = \sum_{k=1}^K \frac{1}{h_k}\left(\frac{1}{M}\sum_{j=1}^M \|\tilde{X}_j(t_{k-1};\theta) - \tilde{X}_j(t_k;\theta)\|^2\right)$.

3.2 End-to-end training algorithm

In the end-to-end training, we assume that the Q-flow net has already been initiated as an approximate solution of the desired Q-flow, see more in Section 3.3. We then minimize $\mathcal{L}^{P\to Q}$ and $\mathcal{L}^{Q\to P}$ in an alternative fashion per "Iter", and the procedure is given in Algorithm 1. Hyperparameter choices and network architectures are further detailed in Appendix B.

Time integration of flow. In the losses (6) and (7), one need to compute the transported samples $X_i(t;\theta)$ and $\tilde{X}_j(t;\theta)$ on time grid points $\{t_k\}_{k=0}^K$. This calls for integrating the neural ODE on $[0,1]$, which we conduct on a fine time grid $t_{k,s}$, $s=0,\ldots,S$, that divides each subinterval $[t_{k-1},t_k]$ into S mini-intervals. We compute the time integration of $f(x,t;\theta)$ using a fixed-grid four-stage Runge-Kutta method on each mini-interval. The fine grid is used to ensure the numerical accuracy of ODE integration and the numerical invertibility of the Q-flow net, i.e., the error of using reverse-time integration as the inverse map (see inversion errors in Table A.1). It is also possible to first train the flow $f(x,t;\theta)$ on a time grid to warm start the later training on a refined grid, so as to improve convergence. We also find that the W_2 regularization can be computed at a coarser grid t_k (S is usually 3-5 in our experiments) without losing the effectiveness of Wasserstein-2 regularization. Finally, one can adopt an adaptive time grid, e.g., by enforcing equal W_2 movement on each subinterval $[t_{k-1},t_k]$ Xu et al. (2022b), so that the representative points are more evenly distributed along the flow trajectory and the learning of the flow model can be further improved.

Algorithm 1 OT Q-flow refinement

input Pre-trained initial flow network $f(x(t),t;\theta)$; training data $\boldsymbol{X} \sim P$ and $\widetilde{\boldsymbol{X}} \sim Q$; hyperparameters: $\{\gamma, \{t_k\}_{k=1}^K, \texttt{Tot}, E, E_0, E_{\mathrm{in}}\}$.
output Refined flow network $f(x(t),t;\theta)$

1: **for** Iter = $1,\ldots,\texttt{Tot}$ **do**
2: (If Iter = 1) Train r_1 by minimizing (5) for E_0 epochs.
3: **for** epoch = $,1,\ldots,E$ **do** {▷ $P\to Q$ refinement}
4: Update θ of $f(x(t),t;\theta)$ by minimizing $\mathcal{L}^{P\to Q}$.
5: Update r_1 by minimizing (5) for E_{in} epochs.
6: **end for**
7: (If Iter = 1) Train $\widetilde{r}_0$ by minimizing (8) for E_0 epochs.
8: **for** epoch = $,1,\ldots,E$ **do** {▷ $Q\to P$ refinement}
9: Update θ of $f(x(t),t;\theta)$ by minimizing $\mathcal{L}^{Q\to P}$.
10: Update $\widetilde{r}_0$ by minimizing (8) for E_{in} epochs.
11: **end for**
12: **end for**

Inner-loop training of r_1 and $\tilde{r}_0$. Suppose the flow net has been successfully warm-started, the transported distributions $(T_0^1)_\# P \approx Q$ and $(T_1^0)_\# Q \approx P$. The two classification nets are first trained for E_0 epochs before the loops of training the flow model and then updated for E_{in} inner-loop epochs in each outer-loop iteration. We empirically find that the diligent updates of r_1 and $\widetilde{r}_0$ in lines 5

and 10 of Algorithm 1 are crucial for successful end-to-end training of Q-flow net. As we update the flow model $f(x, t; \theta)$, the push-forwarded distributions $(T_0^1)_\# P$ and $(T_1^0)_\# Q$ are consequently changed, and then one will need to retrain r_1 and $\widetilde{r}_0$ timely to ensure an accurate estimate of the log-density ratio and consequently the KL loss. Compared with training the flow parameter θ, the computational cost of the two classification nets is light which allows potentially a large number of inner-loop iterations if needed.

Computational complexity. We measure the computational complexity by the number of function evaluations of $f(x(t), t; \theta)$ and of the classification nets $\{r_1, \widetilde{r}_0\}$. Suppose the total number of epochs in outer loop training is $O(E)$, the dominating computational cost lies in the neural ODE integration, which takes $O(8KS \cdot E(M+N))$ function evaluations of $f(x, t; \theta)$. We remark that the Wasserstein-2 regularization (7) incurs no extra computation, since the samples $X_i(t_k; \theta)$ and $\tilde{X}_j(t_k; \theta)$ are available when computing the forward and reverse time integration of $f(x, t; \theta)$. The training of the two classification nets r_1 and $\widetilde{r}_0$ takes $O(4(E_0 + EE_{\text{in}})(M+N))$ additional evaluations of the two network functions since the samples $X_i(1; \theta)$ and $\tilde{X}_j(0; \theta)$ are already computed.

3.3 Flow initialization

We propose to initialize the Q-flow net by a flow model that approximately matches the transported distributions with the target distributions in both directions (and may not necessarily minimize the transport cost). Such an initialization will significantly accelerate the convergence of the end-to-end training, which can be viewed as a refinement of the initial flow.

The initial flow $f(x, t; \theta)$ may be specified using prior knowledge of the problem if available. Generally, when only two data sets $\boldsymbol{X}, \tilde{\boldsymbol{X}}$ are given, the initial flow can be obtained by adopting existing methods in generative flows. In this work, we adopt two approaches: The first method is to construct the initial flow as a concatenation of two CNF models, each of which flows invertibly between P and Z and Z and Q for $Z \sim \mathcal{N}(0, I_d)$. Any existing neural-ODE CNF models may be adopted for this initialization (Grathwohl et al., 2019; Xu et al., 2022b). The second method adapts distribution interpolant neural networks. Specifically, one can use the linear interpolant mapping in (Rhodes et al., 2020; Choi et al., 2022; Albergo & Vanden-Eijnden, 2023) (see Appendix C), and train the neural network velocity field $f(x, t; \theta)$ to match the interpolation (Albergo & Vanden-Eijnden, 2023). Note that any other initialization scheme is compatible with the proposed end-to-end training of the Q-flow model to obtain the OT flow.

4 Experiments

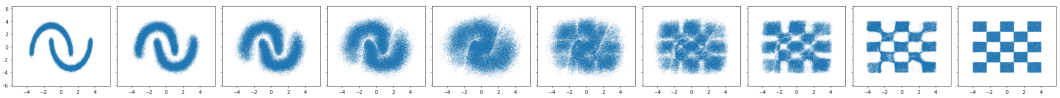
In this section, we demonstrate the effectiveness of the proposed method on several downstream tasks. The benefit of improving DRE between P and Q are shown in Sections 4.2–4.4, and the application to image-to-image translation is presented in Section 4.5. Additional ablation studies regarding hyper-parameter sensitivity are performed in Appendix B.5.

4.1 Infinitesimal density ratio estimation (DRE)

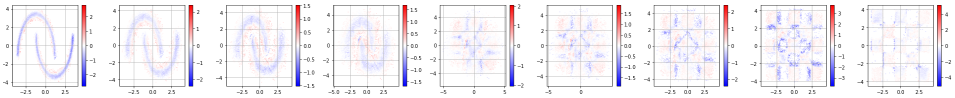
For the DRE task, using the learned OT flow network between P and Q, we propose to train a separate continuous-time neural network, called the *Q-flow-ratio* net, by minimizing a classification loss at time stamps along the flow trajectory. This differs from Choi et al. (2021) which used a 'time score matching' objective, and we also adopt a different time discretization. Details of the method are provided in Appendix A, see Algorithm A.2. In practice, we found our approach to train the density ratio network can be more efficient in some cases. In the experimental results below, we denote our method as "Ours", and compare against three baselines of DRE in high dimensions. The baseline methods are: 1 ratio (by training a single classification network using samples from P and Q), TRE (Rhodes et al., 2020), and DRE-∞ (Choi et al., 2022). We denote P_{t_k} with density $p(\cdot, t_k)$ as the pushforward distribution of P by the Q-flow transport over the interval $[0, t_k]$. The set of distributions $\{P_{t_k}\}$ for $k = 1, \ldots, L$ builds a bridge between P and Q.

4.2 Toy data in 2d

Gaussian mixtures. We simulate P and Q as two Gaussian mixture models with three and two components, respectively, see additional details in Appendix B.1. We compute ratio estimates $\hat{r}(x)$



(a) Trajectory from P to Q



(b) Estimated log-ratio between $P_{t_{k-1}}$ and P_{t_k} by the trained Q-flow-ratio net.

Figure 2: Q-flow trajectory between arbitrary 2D distributions and corresponding log-ratio estimation. **Top**: intermediate distributions by Q-flow net. **Bottom**: corresponding log-ratio estimated by Q-flow-ratio net. Bluer color indicates smaller estimates of the difference $\log(p(x, t_k)/p(x, t_{k-1}))$ evaluated at the common support of the neighboring densities.

with the true value $r(x)$, which can be computed using the analytic expressions of the densities. The results are shown in Figure A.1. We see from the top panel that the mean absolute error (MAE) of Ours is evidently smaller than those of the baseline methods, and Ours also incurs a smaller maximum error $|\hat{r} - r|$ on test samples.

Two-moon to and from checkerboard. We design two densities in $\mathbb{R}^2$ where P represents the shape of two moons and Q represents a checkerboard, see additional details in Appendix B.1. For this more challenging case, the linear interpolation scheme (17) creates a bridge between P and Q as shown in Figure A.5. The flow visually differs from the one obtained by the trained Q-flow net, as shown in Figure 2(a), and the latter is trained to minimize the transport cost. The result of Q-flow-ratio net is shown in Figure 2(b). The corresponding density ratio estimates of $\log p(x, t_k) - \log p(x, t_{k-1})$ visually reflect the actual differences in the two neighboring densities.

4.3 Mutual Information estimation for high-dimensional data

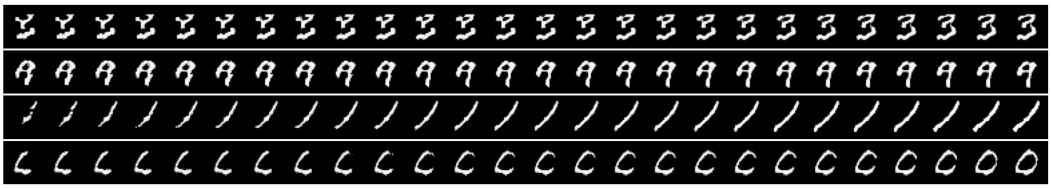
We evaluate different methods on estimating the mutual information (MI) between two correlated random variables from given samples. In this example, we let P and Q be two high-dimensional Gaussian distributions following the setup in (Rhodes et al., 2020; Choi et al., 2022), where we vary the data dimension d in the range of $\{40, 80, 160, 320\}$. Additional details can be found in Appendix B.2. Figure A.2 shows the results by different methods, where the baselines are trained under their proposed default settings. We find that the estimated MI by our method almost perfectly aligns with the ground truth MI values, reaching nearly identical performance as DRE-∞ does. Meanwhile, Ours outperforms the other two baselines and the performance gaps increase as the dimension d increases.

4.4 Energy-based modeling of MNIST

We apply our approach in evaluating and improving an energy-base model (EBM) on the MNIST dataset (LeCun & Cortes, 2005). We follow the prior setup in (Rhodes et al., 2020; Choi et al., 2022), where P is the empirical distribution of MNIST images, and Q is the generated image distributions by three given pre-trained energy-based generative models Additional details are in Appendix B.3. The performance of different DRE methods are measure by the (BPD) metric, which is a widely used metric in evaluating the performance of generative models (Theis et al., 2015; Papamakarios et al., 2017). The results show that Ours reaches the improved performance in Table 1 against baselines: it consistently reaches smaller BPD than the baseline methods across all choices of Q. Meanwhile, we also note computational benefits in training: on one A100 GPU, Ours took approximately 8 hours to converge while DRE-∞ took approximately 33 hours. In addition, we show trajectory of improved

Table 1: DRE performance on the energy-based modeling task for MNIST, reported in BPD and lower is better. Results for DRE-∞ are from (Choi et al., 2022) and results for 1 ratio and TRE are from (Rhodes et al., 2020).

Choice of Q	RQ-NSF				Copula				Gaussian			
Method	Ours	DRE-∞	TRE	1 ratio	Ours	DRE-∞	TRE	1 ratio	Ours	DRE-∞	TRE	1 ratio
BPD (↓)	**1.05**	1.09	1.09	1.09	**1.14**	1.21	1.24	1.33	**1.31**	1.33	1.39	1.96



(a) MNIST

(b) Handbag $\rightarrow$ shoes

Figure 3: The trajectory of samples (in rows) from intermediate distributions of the Q-flow, as it pushes forward the base distribution (leftmost column) to the target distribution (rightmost column). Figure (a) shows the improvement of generated digits using the Q-flow. Figure (b) shows the image-to-image translation from handbag to shoes.

samples from Q to $\tilde{Q}$ for RQ-NSF using the trained Q-flow in Figure 3a. Figure A.3 in the appendix shoes additional improved digits for all three specifications of Q.

4.5 Image-to-image translation

We use Q-flow to learn the continuous-time OT between distributions of RGB images of handbag (Zhu et al., 2016) and shoes (Yu & Grauman, 2014), which we denote as P and Q respectively. We follow the setup in (Korotin et al., 2023), where the goal of the image-to-image translation task is to conditionally generate shoe images by mapping test images of handbag through our trained Q-flow model. We train Q-flow in the latent space of a pre-trained variational auto-encoder (VAE) on P and Q. Additional details are in Appendix B.4.

Figure 3b visualizes continuous trajectories from handbags to shoes generated by the Q-flow model. We find that Q-flow can capture the style and color nuances of corresponding handbags in the generated shoes as the flow model continuously transforms handbag images. Figure A.4 in the appendix shows additional generated shoe images from handbags. Quantitatively, we reach a Frechet Inception Distance ((Heusel et al., 2017), FID) of 15.95 between generated and true images of shoes. The FID remains competitive against FIDs from previous baselines, which range from 22.42 by DiscoGAN (Kim et al., 2017) to 13.77 by NeuralOT (Korotin et al., 2023). Meanwhile, since our Q-flow model learns a *continuous* transport map from source to target domains, it directly provides the gradual interpolation between the source and target samples along the dynamic OT trajectory as depicted in Figure 3b.

5 Discussion

In this work, we develop Q-flow neural-ODE model that smoothly and invertibly transports between a pair of arbitrary distributions P and Q. The flow network is trained to find the dynamic optimal transport between the two distributions and is learned from finite samples from both distributions. The proposed flow model shows strong empirical performance on simulated and real data for the tasks of density ratio estimation and image-to-image translation. For future directions, first, the algorithm of training the Q-flow net can be further enhanced. Because the computational complexity scales with the number of time steps along the trajectory, more advanced time discretization schemes, like adaptive time grids, can further improve the computational efficiency which would be important for high dimensional problems. Second, there are many theoretical open questions, e.g., the theoretical guarantee of learning the OT trajectory, which goes beyond the scope of the current work. For the empirical results, extending to a broader class of applications and more real datasets will be useful.

References

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2019.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9), 2009.

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

Kristy Choi, Madeline Liao, and Stefano Ermon. Featurized density ratio estimation. In *Uncertainty in Artificial Intelligence*, pp. 172–182. PMLR, 2021.

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

Florentin Coeurdoux, Nicolas Dobigeon, and Pierre Chainais. Learning optimal transport between two empirical distributions with normalizing flows. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part V*, pp. 275–290. Springer, 2023.

Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I 14*, pp. 274–289. Springer, 2014.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *Advances in neural information processing systems*, 30, 2017.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Jiaojiao Fan, Shu Liu, Shaojun Ma, Yongxin Chen, and Hao-Min Zhou. Scalable computation of monge maps with general costs. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022. URL https://openreview.net/forum?id=rEnGR3VdDW5.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pp. 3154–3164. PMLR, 2020.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Kristjanson Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *ArXiv*, abs/1810.01367, 2019.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Han Huang, Jiajia Yu, Jie Chen, and Rongjie Lai. Bridging mean-field games and normalizing flows with trajectory regularization. *Journal of Computational Physics*, pp. 112155, 2023.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Masahiro Kato and Takeshi Teshima. Non-negative bregman divergence minimization for deep direct density ratio estimation. In *International Conference on Machine Learning*, pp. 5320–5333. PMLR, 2021.

Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM international conference on data mining*, pp. 389–400. SIAM, 2009.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International conference on machine learning*, pp. 1857–1865. PMLR, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43 (11):3964–3979, 2020.

Alexander Korotin, Daniil Selikhanovych, and Evgeny Burnaev. Neural optimal transport. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=d8CBRlWNkqH`.

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

Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005. URL `http://yann.lecun.com/exdb/mnist/`.

Wonjun Lee, Rongjie Lai, Wuchen Li, and Stanley Osher. Generalized unnormalized optimal transport and its fast algorithms. *Journal of Computational Physics*, 436:110041, 2021.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=PqvMRDCJT9t`.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Shu Liu, Wuchen Li, Hongyuan Zha, and Haomin Zhou. Neural parametric fokker–planck equation. *SIAM Journal on Numerical Analysis*, 60(3):1385–1449, 2022.

Xiao-Li Meng and Wing Hung Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, pp. 831–860, 1996.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pp. 666–704, 1781.

George V Moustakides and Kalliopi Basioti. Training neural networks for likelihood/density ratio estimation. *arXiv preprint arXiv:1911.00405*, 2019.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

Kirill Neklyudov, Rob Brekelmans, Daniel Severo, and Alireza Makhzani. Action matching: Learning stochastic dynamics from samples. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25858–25889. PMLR, 23–29 Jul 2023.

Derek Onken, S Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

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

Jing Qin. Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630, 1998.

Benjamin Rhodes, Kai Xu, and Michael U. Gutmann. Telescoping density-ratio estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4905–4916. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/33d3b157ddc0896addfb22fa2a519097-Paper.pdf`.

Lars Ruthotto, Stanley J Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu 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.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul Von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008.

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

Esteban G Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Yujia Xie, Minshuo Chen, Haoming Jiang, Tuo Zhao, and Hongyuan Zha. On scalable and efficient computation of large scale optimal transport. In *International Conference on Machine Learning*, pp. 6882–6892. PMLR, 2019.

Chen Xu, Xiuyuan Cheng, and Yao Xie. Invertible neural networks for graph prediction. *IEEE Journal on Selected Areas in Information Theory*, 3(3):454–467, 2022a. doi: 10.1109/JSAIT.2022.3221864.

Chen Xu, Xiuyuan Cheng, and Yao Xie. Invertible normalizing flow neural networks by jko scheme. *arXiv preprint arXiv:2212.14424*, 2022b.

Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 192–199, 2014.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pp. 597–613. Springer, 2016.

A Infinitesimal density ratio estimation

We first introduce related works and preliminaries of DRE. We then propose training infinitesimal DRE by logistic classification in Section A.1. We present the complete algorithm in Section A.2.

DRE literature. Density ratio estimation between distributions P and Q is a fundamental problem in statistics and machine learning (Meng & Wong, 1996; Sugiyama et al., 2012; Choi et al., 2021). It has direct applications in important fields such as importance sampling (Neal, 2001), change-point detection (Kawahara & Sugiyama, 2009), outlier detection (Kato & Teshima, 2021), mutual information estimation (Belghazi et al., 2018), etc. Various techniques have been developed, including probabilistic classification (Qin, 1998; Bickel et al., 2009), moment matching (Gretton et al., 2009), density matching (Sugiyama et al., 2008), etc. Deep NN models have been leveraged in classification approach Moustakides & Basioti (2019) due to their expressive power. However, as has been pointed out in (Rhodes et al., 2020), the estimation accuracy by a single classification may degrade when P and Q differ significantly.

To overcome this issue, (Rhodes et al., 2020) introduced a telescopic DRE approach by constructing intermediate distributions to bridge between P and Q. (Choi et al., 2022) further proposed to train an infinitesimal, continuous-time ratio net via the so-called time score matching. Despite their improvement over the prior classification methods, both approaches rely on construction of the intermediate distributions between P and Q that is not optimal. In contrast, our proposed Q-flow network leverages the expressiveness of deep networks to construct the intermediate distributions by the continuous-time flow transport, and the flow trajectory is regularized to minimize the transport cost in dynamic OT. The model empirically improves the DRE accuracy (see Section 4). In computation, (Choi et al., 2022) applies score matching to compute the infinitesimal change of log-density. The proposed Q-flow-ratio net is based on classification loss training using a fixed time grid which avoids score matching and is computationally lighter (Section A.2).

Telescopic and infinitesimal DRE preliminaries. To circumvent the problem of DRE distinctly different p and q, the *telescopic DRE* (Rhodes et al., 2020) proposes to "bridge" the two densities by a sequence of intermediate densities p_k, $k = 0, \cdots, L$, where $p_0 = p$ and $p_L = q$. The consecutive pairs of (p_k, p_{k+1}) are chosen to be close so that the DRE can be computed more accurately, and then by

$$\log(q(x)/p(x)) = \log p_L(x) - \log p_0(x) = \sum_{k=0}^{L-1} \log p_{k+1}(x) - \log p_k(x), \tag{9}$$

the log-density ratio between q and p can be computed with improved accuracy than a one-step DRE. The *infinitesmal DRE* (Choi et al., 2022) considers a time continuity version of (9). Specifically, suppose the time-parametrized density $p(x, t)$ is differentiable on $t \in [0, 1]$ with $p(x, 0) = p$ and $p(x, 1) = q$, then

$$\log(q(x)/p(x)) = \log p(x, 1) - \log p(x, 0) = \int_0^1 \partial_t \log p(x, t) dt. \tag{10}$$

The quantity $\partial_t \log p(x, t)$ was called the "time score" and can be parametrized by a neural network.

We use a trained Q-flow network $f(x, t; \theta)$ for infinitesmal DRE as a focused application.

A.1 Training by logistic classification

Let $p(x, t) = (T_t)_\# p$ and T_t is the transport induced by the trained Q-flow net in Section 3. Using (10), we propose to parametrize the time score $\partial_t \log p(x, t)$ by a neural network $r(x, t; \theta_r)$ with parameter θ_r, called the *Q-flow-ratio* net. The training is by logistic classification applied to transported data distributions on consecutive time grid points: Given a deterministic time grid $0 = t_0 < t_1 < \ldots < t_L = 1$ (which again is an algorithmic choice, see Section A.2), we expect that the integral

$$R_k(x; \theta_r) := \int_{t_{k-1}}^{t_k} r(x, t; \theta_r) dt \approx \int_{t_{k-1}}^{t_k} \partial_t \log p(x, t) dt = \log(p(x, t_k)/p(x, t_{k-1})). \tag{11}$$

Algorithm A.2 Infinitesimal DRE training

input Training samples $\boldsymbol{X} \sim P$ and $\widetilde{\boldsymbol{X}} \sim Q$; pre-trained Q-flow net $f(x(t), t; \theta)$; hyperparameters: $\{\{t_k\}_{k=1}^L, \texttt{Tot_iter}\}$

output Trained network $r(x, t; \theta_r)$.

1: **for** $k = 1, \ldots, L-1$ **do**
2: Obtain $\{X_i(t_k)\}_{i=1}^N, \{\tilde{X}_j(t_k)\}_{j=1}^M$ by transporting all training samples $\{\boldsymbol{X}, \tilde{\boldsymbol{X}}\}$ using the given Q-flow net $f(x, t; \theta)$
3: **end for**
4: **for** Iter = $1, \ldots, \texttt{Tot_iter}$ **do**
5: Draw mini-batches of samples from $\{X_i(t_k)\}_{i=1}^N, \{\tilde{X}_j(t_k)\}_{j=1}^M$.
6: Train θ_r upon minimizing (13).
7: **end for**

By that logistic classification recovers the log density ratio as has been used in Section 3.1, this suggests the loss on interval $[t_{k-1}, t_k]$ as follows, where $X_i(t) := T_0^t(X_i)$ and T_0^t is computed by integrating the trained Q-flow net,

$$L_k^{P\to Q}(\theta_r) = \frac{1}{N}\sum_{i=1}^N \log(1 + e^{R_k(X_i(t_{k-1});\theta_r)}) + \frac{1}{N}\sum_{i=1}^N \log(1 + e^{-R_k(X_i(t_k);\theta_r)}). \tag{12}$$

When $k = L$, the distribution of $X_i(t_L)$ may slightly differ from that of Q due to the error in matching the terminal densities in Q-flow net. Thus by replacing the 2nd term in (12) with an empirical average over the Q-samples $\tilde{X}_j$ may be beneficial. In the reverse direction, define $\tilde{X}_j(t) := T_1^t(\tilde{X}_j)$, we similarly have $L_k^{Q\to P}(\theta_r) = \frac{1}{M}\sum_{j=1}^M \log(1+e^{R_k(\tilde{X}_j(t_{k-1});\theta_r)}) + \frac{1}{M}\sum_{j=1}^M \log(1+e^{-R_k(\tilde{X}_j(t_k);\theta_r)})$, and when $k = 1$, we replace the 1st term with an empirical average over the P-samples X_i. The training of the Q-flow-ratio net is by

$$\min_{\theta_r} \sum_{k=1}^L L_k^{P\to Q}(\theta_r) + L_k^{Q\to P}(\theta_r). \tag{13}$$

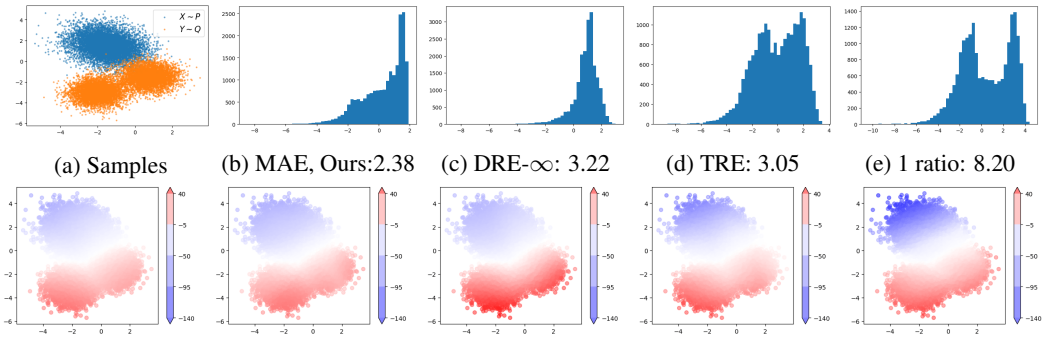
When trained successfully, the integral of $r(x, t; \theta_r)$ over $t \in [0, 1]$ yields the desired log density ratio $\log(q/p)$ by (10), and furtherly the integral $\int_s^t r(x, t'; \theta_r)dt'$ provides an estimate of $\log(p(x, t)/p(x, s))$ for any $s < t$ on $[0, 1]$.

A.2 Algorithm and computational complexity

The details of minimizing (13) is given in Algorithm A.2. We use an evenly spaced time grid $t_k = k/L$ in all experiments. In practice, one can also progressively refine the time grid in training, starting from a coarse grid to train a Q-flow-ratio net $r(x, t; \theta_r)$ and use it as a warmstart for training the network parameter θ_r on a refined grid. When the time grid is fixed, it allows us to compute the transported samples $\{X_i(t_k)\}_{i=1}^N, \{\tilde{X}_j(t_k)\}_{j=1}^M$ on all t_k once before the training loops of Q-flow-ratio net (line 1-3). This part takes $O(8KS(M + N))$ function evaluations of the pre-trained Q-flow net $f(x, t; \theta)$. Suppose the training loops of line 4-6 conducts E epochs in total. Assume each time integral in R_k (11) is computed by a fixed-grid four-stage Runge-Kutta method, then $O(4LE(M + N))$ function evaluations of $r(x, t; \theta_r)$ is needed to compute the overall loss (13).

B Additional experimental details

When training all networks, we use the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of `1e-3`.



(a) Samples (b) MAE, Ours:2.38 (c) DRE-∞: 3.22 (d) TRE: 3.05 (e) 1 ratio: 8.20

(f) Left to right: True ratio $r(x)$, $\hat{r}(x)$ by Ours, DRE-∞, TRE, and 1 ratio.

Figure A.1: Estimated log density ratio between 2D Gaussian mixture distributions P (three components) and Q (two components). **Top:** (a) training samples from P and Q. (b)-(d) histograms of errors $\log(|r(x) - \hat{r}(x)|)$ computed at 10K test samples shown in log-scale. The MAE (14) are shown in the captions. **Bottom:** true and estimated $\log(q/p)$ from different models shown under shared colorbars.

B.1 Toy data in 2d

Gaussian mixtures. *Setup*: We design the Gaussian mixtures P and Q as follows:

$$P = \frac{1}{3}\left(\mathcal{N}(\begin{bmatrix}-2\\2\end{bmatrix}, 0.75I_2) + \mathcal{N}(\begin{bmatrix}-1.5\\1.5\end{bmatrix}, 0.25I_2) + \mathcal{N}(\begin{bmatrix}-1\\1\end{bmatrix}, 0.75I_2)\right)$$
$$Q = \frac{1}{2}\left(\mathcal{N}(\begin{bmatrix}0.75\\-1.5\end{bmatrix}, 0.5I_2) + \mathcal{N}(\begin{bmatrix}-2\\-3\end{bmatrix}, 0.5I_2)\right).$$

Then, 60K training samples and 10K test samples are randomly drawn from P and Q. We intentionally designed the Gaussian mixtures so that their supports barely overlap. The goal is to estimate the log-density ratio $r(x) = \log q(x) - \log p(x)$ on test samples.

Given a trained ratio estimator $\hat{r}(x)$, we measure its performance based on the MAE

$$\frac{1}{N'}\sum_{i=1}^{N'} |r(X_i) - \hat{r}(X_i)| + \frac{1}{M'}\sum_{j=1}^{M'} |r(\tilde{X}_j) - \hat{r}(\tilde{X}_j)|, \tag{14}$$

where we use N' and M' test samples from P and Q, and $r(x)$ denotes the true density between P and Q.

Q-flow : To initialize the two JKO-iFlow models that consists of the initial Q-flow , we specify the JKO-iFlow as:

- The flow network $f(x(t), t; \theta_P)$ and $f(x(t), t; \theta_Q)$ consists of fully-connected layers 3→128→128→2. The Softplus activation with $\beta = 20$ is used. We concatenate t along x to form an augmented input into the network.
- We train the initial flow with a batch size of 2000 for 100 epochs along the grid $[0, 0.25), [0.25, 0.625), [0.625, 1)$.

To refine the Q-flow , we concatenate the trained $f(x(t), t; \theta_P)$ and $f(x(t), t; \theta_Q)$, where the former flows in $[0, 1)$ to transport P to Z and the latter flows in $[1, 0)$ to transport Z to Q. We then use the

Table A.1: Inversion error $\mathbb{E}_{x\sim P}\|T_1^0(T_0^1(x)) - x\|_2^2 + \mathbb{E}_{y\sim Q}\|T_0^1(T_1^0(y)) - y\|_2^2$ of Q-flow computed via sample average on the test split of the data set.

2d Gaussian mixture	moon-to-checkerboard	High-dimenisonal Gaussians ($d = 320$)	MNIST (Q by RQ-NSF)
5.45e-7	7.24e-7	3.44e-5	5.23e-5

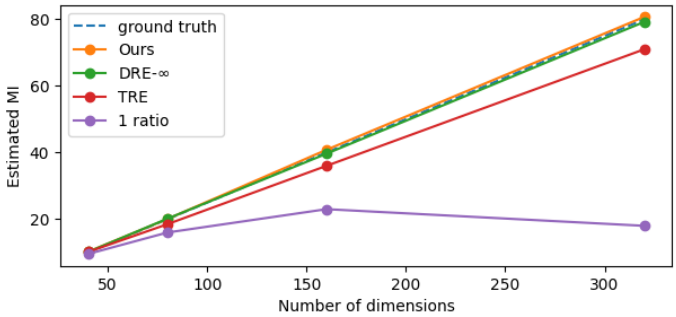


Figure A.2: Estimated MI between two correlated high-dimensional Gaussian random variables.

time grid $[0, 0.25), [0.25, 0.625), [0.625, 1), [1, 0.625), [0.625, 0.25), [0.25, 0)$ to train $f(x(t), t; \theta)$ with $\theta = \{\theta_P, \theta_Q\}$; we note that the above time grid can be re-scaled to obtain the time grid $\{t_k\}_{k=1}^K$ over $[0, 1]$. The hyperparameters for Algorithm 1 are: `Tot`=2, $E_0 = 300$, $E = 50$, $E_{\text{in}} = 4, \gamma = 0.5$. The classification networks $\{r_1, \widetilde{r}_0\}$ consists of fully-connected layers `2`→`312`→`312`→`312`→`1` with the Softplus activation with $\beta = 20$, and it is trained with a batch of 200.

Infinitesiml DRE: The network consists of fully-connected layers `3`→`256`→`256`→`256`→`1` with the Softplus activation with $\beta = 20$. The input dimension is 3 because we concatenate time t along the input $x \in \mathbb{R}^2$ to form an augmented input. Using the trained Q-flow model, we then produce a bridge of 6 intermediate distributions using the pre-scaled grid $[0, 0.25), [0.25, 0.625), [0.625, 1), [1, 0.625), [0.625, 0.25), [0.25, 0)$ for the Q-flow . We then train the network $r(x, t; \theta_r)$ for 100 epochs with a batch size of 1000, corresponding to `Tot_iter`=6K in Algorithm A.2.

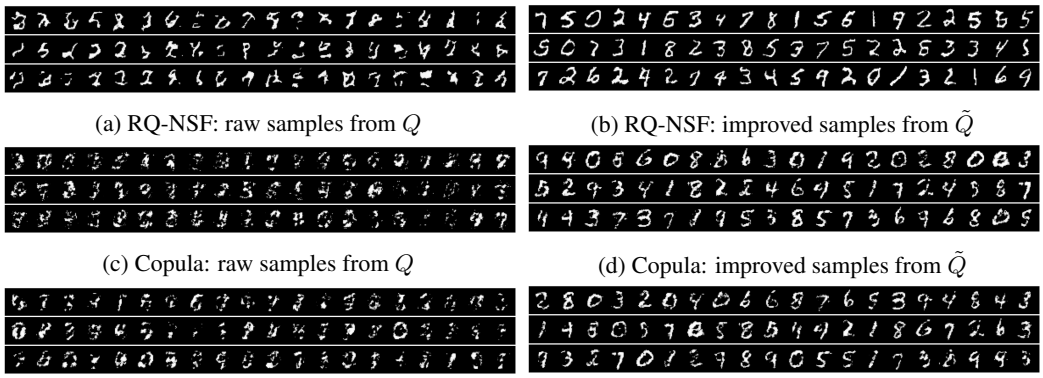
Two-moon to and from checkerboard. *Setup*: We generate 2D samples whose marginal distribution has the shape of two moons and a checkerboard (see Figure 2(a), leftmost and rightmost scatter plots). We randomly sample 100K samples from P and Q to train the Q-flow and the infinitesimal DRE.

Q-flow : To initialize the two JKO-iFlow models that consists of the initial Q-flow , we specify the JKO-iFlow as:

- The flow network $f(x(t), t; \theta_P)$ and $f(x(t), t; \theta_Q)$ consists of fully-connected layers `3`→`256`→`256`→`2`. The Softplus activation with $\beta = 20$ is used. We concatenate t along x to form an augmented input into the network.
- We train the initial flow with a batch size of 2000 for 100 epochs along the grid $[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1)$.

To refine the Q-flow , we concatenate the trained $f(x(t), t; \theta_P)$ and $f(x(t), t; \theta_Q)$, where the former flows in $[0, 1]$ to transport P to Z and the latter flows in $[1, 0]$ to transport Z to Q. We then use the grid $[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1), [1, 0.75), [0.75, 0.5), [0.5, 0.25), [0.25, 0)$ (which can be re-scaled to form the time grid over $[0, 1)$) to train $f(x(t), t; \theta)$ with $\theta = \{\theta_P, \theta_Q\}$. The hyperparameters for Algorithm 1 are: `Tot`=2, $E_0 = 300$, $E = 50$, $E_{\text{in}} = 4, \gamma = 0.5$. The classification networks $\{r_1, \widetilde{r}_0\}$ consists of fully-connected layers `2`→`312`→`312`→`312`→`1` with the Softplus activation with $\beta = 20$, and it is trained with a batch of 200.

Infinitesiml DRE: The network consists of fully-connected layers `3`→`256`→`256`→`256`→`1` with the Softplus activation with $\beta = 20$. The input dimension is 3 because we concatenate time t along the input $x \in \mathbb{R}^2$ to form an augmented input. Using the trained Q-flow model, we then produce a bridge of 8 intermediate distributions using the pre-scaled grid interval $[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1), [1, 0.75), [0.75, 0.5), [0.5, 0.25), [0.25, 0)$ for the Q-flow . We then train the network $r(x, t; \theta_r)$ for 500 epochs with a batch size of 500, corresponding to `Tot_iter`=100K in Algorithm A.2.



(a) RQ-NSF: raw samples from Q

(b) RQ-NSF: improved samples from $\tilde{Q}$

(c) Copula: raw samples from Q

(d) Copula: improved samples from $\tilde{Q}$

(e) Gaussian: raw samples from Q

(f) Gaussian: improved samples from $\tilde{Q}$

Figure A.3: Improvement in generated samples from Q, where Q is given by RQ-NSF, Copula, or Gaussian. Each of the three choices of Q is defined by a pre-trained invertible model F that yields $Q = F_{\#}\mathcal{N}(0, I_d)$.

B.2 Mutual Information estimation for high-dimensional data

Setup: The task is to estimate the $\log$-density ratio between two high-dimensional Gaussian distributions of dimension d, where this task can be viewed as an MI estimation problem. We follow the same setup as in (Rhodes et al., 2020; Choi et al., 2022). The first Gaussian distribution $P = \mathcal{N}(0, \Sigma)$, where Σ is a block-diagonal covariance matrix with 2×2 small blocks having 1 on the diagonal and 0.8 on the off-diagnal. The second Gaussian distribution $Q = \mathcal{N}(0, I_d)$ is the isotropic Gaussian in $\mathbb{R}^d$. We randomly draw 100K samples for each choice of d, which varies from 40 to 320.

To be more precise, we hereby draw the connection of the DRE task with mutual information (MI) estimation, following (Rhodes et al., 2020). We first recall the definition of MI between two correlated random variables U and V:

$$I(U;V) = \mathbb{E}_{p(U,V)}\left[\log \frac{p(U,V)}{p(U)p(V)}\right]. \tag{15}$$

Now, given $X = (x_1, \ldots, x_d) \sim P = \mathcal{N}(0, \Sigma)$, we define $U = (x_1, x_3, \ldots, x_{d-1})$ and $V = (x_2, x_4, \ldots, x_d)$. By the construction of Σ, we thus have $p(U)p(V) = Q(X)$ for $Q = \mathcal{N}(0, I_d)$. As a result, the MI in (15) between U and V is equivalent to $\mathbb{E}_{X\sim P}[-r(X)]$, where $r(x) = \log \frac{Q(x)}{P(x)}$ is the objective of interest in DRE.

Q-flow : We specify the following when training the Q-flow :

- The flow network $f(x(t), t; \theta)$ consists of fully-connected layers with dimensions `(d+1)`$\rightarrow$`min(4d,1024)`$\rightarrow$`min(4d,1024)`$\rightarrow$`d`. The Softplus activation with $\beta = 20$ is used. We concatenate t along x to form an augmented input into each network layer.
- We train the flow network for 100 epochs with a batch size of 500, in both the flow initialization phase and the end-to-end refinement phase. The flow network is trained along the evenly-spaced time grid $[t_{k-1}, t_k)$ for $k = 1, \ldots, L_d$, and we let $t_k = k/L_d$. L_d increases as the dimension d increases. We specify the choices as $(L_d, d) \in \{(4, 40), (6, 80), (7, 160), (8, 320)\}$.

The hyperparameters for Algorithm 1 are: `Tot=2`, $E_0 = 500$, $E = 100$, $E_{\text{in}} = 2, \gamma = 0.5$. The classification networks $\{r_1, \widetilde{r}_0\}$ consists of fully-connected layers $d \rightarrow \min(4d, 1024) \rightarrow \min(4d, 1024) \rightarrow \min(4d, 1024) \rightarrow 1$ with the Softplus activation with $\beta = 20$, and it is trained with a batch of 200.

Infinitesiml DRE: The network consists of fully-connected layers with dimensions

`(d+1)`$\rightarrow$`min(4d,1024)`$\rightarrow$`min(4d,1024)`$\rightarrow$`min(4d,1024)`$\rightarrow$`1`, using the Softplus activation with $\beta = 20$. The input dimension is $d + 1$ because we concatenate time t along the input $x \in \mathbb{R}^d$ to form

an augmented input. Using the trained Q-flow model, we then produce a bridge of L_d intermediate distributions using the grid $[t_{k-1}, t_k)$ specified above for the Q-flow . We then train the network $r(x, t; \theta_r)$ for 1000 epochs with a batch size of 512, corresponding to `Tot_iter`=195K in Algorithm A.2.

B.3 Energy-based modeling of MNIST

Setup: We follow the prior setup in (Rhodes et al., 2020; Choi et al., 2022), where P is the empirical distribution of MNIST images, and Q is the generated image distributions by three given pre-trained energy-based generative models: a Gaussian noise model, a Gaussian copula model, and a Rational Quadratic Neural Spline Flow model (RQ-NSF) (Durkan et al., 2019). Specifically, the images are in dimension $d = 28^2 = 784$, and each of the pre-trained models provides an invertible mapping $F : \mathbb{R}^d \to \mathbb{R}^d$, where $Q = F_{\#}\mathcal{N}(0, I_d)$. We train a Q-flow net between $(F^{-1})_{\#}P$ and $(F^{-1})_{\#}Q$, the latter by construction equals $\mathcal{N}(0, I_d)$. Using the trained Q-flow net, we go back to the input space and train the Q-flow-ratio net using the intermediate distributions between P and Q. The trained Q-flow-ratio $r(x, s; \hat{\theta}_r)$ provides an estimate of the data density $p(x)$ by $\hat{p}(x)$ defined as $\log \hat{p}(x) = \log q(x) - \int_0^1 r(x, s; \hat{\theta}_r) ds$, where $\log q(x)$ is given by the change-of-variable formula using the pre-trained model F and the analytic expression of $\mathcal{N}(0, I_d)$. As a by-product, since our Q-flow net provides an invertible mapping T_0^1, we can use it to obtain an improved generative model on top of F. Specifically, the improved distribution $\tilde{Q} := (F \circ T_1^0)_{\#}\mathcal{N}(0, I_d)$, that is, we first use Q-flow to transport $\mathcal{N}(0, I_d)$ and then apply F. The performance of the improved generative model can be measured using the "bits per dimension" (BPD) metric,

$$\text{BPD} = \frac{1}{N'} \sum_{i=1}^{N'} [-\log \hat{p}(X_i)/(d \log 2)], \tag{16}$$

where X_i are N'=10K test images drawn from P. which is a widely used metric in evaluating the performance of generative models (Theis et al., 2015; Papamakarios et al., 2017). In our setting, the BPD can also be used to compare the performance of the DRE.

Q-flow : We specify the following when training the Q-flow :

- The flow network $f(x(t), t; \theta)$ consists of fully-connected layers with dimensions `(d+1)`→`1024`→`1024`→`1024`→`d`. The Softplus activation with $\beta = 20$ is used. We concatenate t along x to form an augmented input into the network.
- In a block-wise fashion, we train the network with a batch size of 1000 for 100 epochs along the grid $[t_{k-1}, t_{k-1} + h_k)$ for $k = 1, \ldots, 5$. We let $h_k = 0.5 \cdot 1.1^{k-1}$.

Figure A.4: Additional test images of handbag (top row) and corresponding generated shoe images by Q-flow model (bottom 5 rows). To generate different shoes for a given handbag, we sample random latent codes given by the VAE, map them through trained Q-flow model, and decode back through the VAE decoder to visualize different generated shoes in the pixel space.

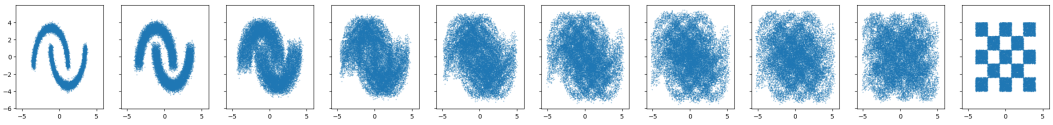


Figure A.5: Bridge construction between P (leftmost) and Q (rightmost) via the linear interpolation scheme (17). Specifically, we choose $\alpha_k = k/9$ for $k = 0, \ldots, 9$.

The hyperparameters for Algorithm 1 are: `Tot`=2, $E_0 = 100$, $E = 500$, $E_{\text{in}} = 2, \gamma = 0.5$. The classification networks $\{r_1, \widetilde{r}_0\}$ consists of fully-connected layers $784 \to 1024 \to 1024 \to 1024 \to$`1` with the Softplus activation with $\beta = 20$, and it is trained with a batch of 200.

Infinitesiml DRE: We use the same convolutional U-Net as described in (Choi et al., 2022, Table 2), which consists of an encoding block and a decoding block comprised of convolutional layers with varying filter sizes. Using the trained Q-flow model, we then produce a bridge of 5 intermediate distributions using the intervals $[t_{k-1}, t_{k-1} + h_k)$ specified above for the Q-flow . We then train the network $r(x, t; \theta_r)$ for 300 epochs with a batch size of 128, corresponding to `Tot_iter`=117K in Algorithm A.2.

B.4 Image-to-image translation

The dataset of handbags P has 137K images and the dataset of shoes Q has 50K images, which are (3,64,64) RGB images. Following (Korotin et al., 2023), we reserve 10% of data from P and Q as test set and compute the FID between generated shoe images (from the test handbag images) and true shoe images from the test set.

We first train a single deep VAE on both P and Q. We train the deep VAE in an adversarial manner following (Esser et al., 2021). Specifically, given a raw image input X, the encoder $\mathcal{E}$ of the VAE maps X to $(\mu(X), \Sigma(X))$ parametrizing a multivariate Gaussian of dimension d. Then, the VAE is trained so that for a random latent code $X_{enc} \sim \mathcal{N}(\mu(X), \Sigma(X))$, the decoded image $\mathcal{D}(X_{enc}) \approx X$. In our case, each latent code X_{enc} has shape $(12, 8, 8)$, so that $d = 768$.

The training data for Q-flow are thus sets of random latent codes X_{enc} (obtained from $X \sim P$) and Y_{enc} (obtained from $Y \sim P$), where Q-flow finds the dynamic OT between the marginal distributions of X_{enc} and Y_{enc}. We then obtain the trajectory between P and Q by mapping the OT trajectory in latent space by the decoder $\mathcal{D}$

The flow architecture $f(x(t), t; \theta)$ consists of convolutional layers of dimensions 12-64-256-512-512-1024, followed by convolutional transpose layers whose filters mirror the convolutional layers. The kernel sizes are 3-3-3-3-3-3-3-4-3-3 with strides 1-1-2-1-1-1-1-2-1-1. We use the softplus activation with $\beta = 20$. The initializaation of the flow is done by the method of Interflow Albergo & Vanden-Eijnden (2023), where at each step, we draw random batches of 128 X and 128 Y and then obtain 128 random latent codes X_{enc} and 128 Y_{enc}. We trained the initialized flow for 26K steps. To apply Algorithm 1, we let $\gamma = 0.05$, $t_k = k/10$ for $k = 0, \ldots, 10$, and `Tot`=2. Architecture of the classifier networks r_1 and $\tilde{r}_0$ is based on (Choi et al., 2022), where the encoding layers of the classifier are convolutional filters of sizes 12-256-512-512-1024-1024 with kernel size equal to 3 and strides equal to 1-1-3-1-1. The decoding layers of the classifier resemble the encoding layers, and the final classification is made by passing the deep decoded feature through a fully-connected network with size 768-768-768-1. These classifiers are initially trained for 4000 batches with batch size 512. We train flow parameters θ for 5000 batches in every iteration with a batch size of 256, and we update the training of r_1 and $\tilde{r}_0$ every 10 batches of training θ to train them for 20 batches.

B.5 Hyper-parameter sensitivity

Overall, we did not purposely tune the hyperparameters in Section 4, and found that the Algorithm 1 and A.2 are not sensitive to hyper-parameter selections. We conduct additional ablation studies by varying the combination of γ in Algorithm 1 and the time grid $\{t_k\}$ in Algorithm A.2. We tested all combinations on the MNIST example in Section 4.4 with RQ-NSF target Q. Table A.2 below presents our method's performance, with the highest BPD (1.062) remaining lower than those by other DRE baselines in Table 1 (the lowest of which is 1.09). Small variations in the table can be

Table A.2: BPD on MNIST with RQ-NSF target Q over combinations of γ in Algorithm 1 and time grid $\{t_k\}$ in Algorithm A.2.

γ & $\{t_k\}$	$t_k = k/L$	$t_k = (k/L)^2$	$t_k = \sqrt{k/L}$
0.5	1.046	1.044	1.047
1	1.042	1.041	1.044
5	1.057	1.055	1.062

attributed to the learned OT trajectory influenced by the choice of γ. Specifically, smaller γ may lead to less smooth trajectories between P and Q, while larger γ may result in a higher KL-divergence between the pushed and target distributions due to insufficient amount of distribution transportation by the refined flow, both potentially impacting DRE accuracy.

C Linear stochastic interpolation used in Rhodes et al. (2020)

The interpolation scheme used in (Rhodes et al., 2020, Eq (5)) states that given a pair of random samples $X(0) \sim P$ and $X(1) \sim Q$, the interpolated sample $X(t_k)$ is defined as

$$X(t_k) = \sqrt{1 - \alpha_k^2} X(0) + \alpha_k X(1), \tag{17}$$

where α_k forms an increasing sequence from 0 to 1. An illustration of $\{X(t_k)\}$ is given in Figure A.5.