Computing high-dimensional optimal transport by flow neural networks

Chen Xu
Georgia Institute of Technology

Xiuyuan Cheng
Duke University

Yao Xie
Georgia Institute of Technology

Abstract

Computing optimal transport (OT) for general high-dimensional data has been a long-standing challenge. Despite much progress, most of the efforts including neural network methods have been focused on the static formulation of the OT problem. The current work proposes to compute the dynamic OT between two arbitrary distributions P and Q by optimizing a flow model, where both distributions are only accessible via finite samples. Our method learns the dynamic OT by finding an invertible flow that minimizes the transport cost. The trained optimal transport flow subsequently allows for performing many downstream tasks, including infinitesimal density ratio estimation (DRE) and domain adaptation by interpolating distributions in the latent space. The effectiveness of the proposed model on high-dimensional data is demonstrated by strong empirical performance on OT baselines, image-to-image translation, and high-dimensional DRE.

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 the data. Despite 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 making use of 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 (optimal) transport has also been applied to achieve model fairness (Silvia et al., 2020). By transporting distributions corresponding to different sensitive attributes to a common distribution, an unfair model is calibrated to match certain desired fairness criteria (e.g., demographic parity (Jiang et al., 2020)). Additionally, the transport map can provide intermediate interpolating distributions between P and Q. In density ratio estimation, 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 help solve problems in computer vision, such as 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 parameterized by time $t \in [0, 1]$ and satisfying that $T_0 = \mathrm{Id}$ (the identity map) 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 where we represent T_t as the solution map of an ODE, whose velocity field is parameterized by a neural network (Chen et al., 2018). 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 from P and Q. The velocity field is optimized to minimize the transport cost to approximate the optimal velocity in the dynamic OT formulation, i.e., the Benamou-Brenier equation.

The neural-ODE model has been intensively developed

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

in Continuous Normalizing Flows (CNF) (Kobyzev et al., 2020). In CNF, the continuous-time flow model, usually parameterized 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 and 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. We emphasize that our motivation is to solve high dimensional OT by a flow model rather than CNF generative applications.

After developing a general approach to the Q-flow model, in the second half of the paper, we focus on the application of telescopic DRE. After training a Q-flow model (the "flow net"), we leverage the intermediate densities $\rho(x, t)$, which is accessed by finite sampled by pushing the P samples by T_t, to train an additional continuous-time classification network (the "ratio net") over time $t \in [0, 1]$. The ratio net estimates the infinitesimal change of the log-density $\log \rho(x, t)$ over time, and its time-integral from 0 to 1 yields an estimate of the (log) ratio q/p. The efficiency of the proposed Q-flow net and the infinitesimal DRE net is experimentally demonstrated on high-dimensional data.

In summary, the contributions of the work include:

• We develop a flow-based model *Q-flow* net to learn a continuous invertible transport map between arbitrary pair of distributions P and Q in $\mathbb{R}^d$ from two sets of data samples. We propose training a neural ODE model to minimize the transport cost so that the flow approximates the optimal transport in dynamic OT. The end-to-end training of the model refines any initial flow that may not attain the optimal transport, e.g., obtained by training two CNFs or other interpolating schemes.

• Leveraging a trained Q-flow net, we propose to train a separate continuous-time network, called *flow-ratio* net, to perform infinitesimal DRE from P to Q given finite samples. The flow-ratio net is trained by minimizing a classification loss to distinguish neighboring distributions on a discrete-time grid along the flow, and it improves the performance over prior models on high-dimensional mutual information estimation and energy-based generative models.

• We show the effectiveness of the Q-flow net on simulated and real data. On public OT benchmarks, we demonstrate improved performance over popular baselines. On the image-to-image translation task, the proposed Q-flow learns a trajectory from an input image to a target sample that resembles the input in style, and it also achieves comparable or better quantitative metrics than the state-of-the-art neural OT model.

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; Papamakarios et al., 2021). The transport is parameterized by deep neural networks, whose parameters are trained via minimizing the Kullback-Leibler (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., 2017; Behrmann et al., 2019) normalizing flow models have been developed, along with proposed regularization techniques (Onken et al., 2021; Xu et al., 2022, 2023) that facilitate the training of such models in practice. Since our Q-flow can be viewed as 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-parameterized velocity field. (Onken et al., 2021) proposed to regularize the flow trajectory by ℓ_2 transport cost and the deviation from the Hamilton–Jacobi–Bellman (HJB) equation. These regularizations have been shown to improve effectively 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. Several works have been done to establish a continuous-time interpolation between general high-dimensional distributions. (Albergo and Vanden-Eijnden, 2023) proposed to use a stochastic interpolant map between two arbitrary distributions and train a neural network parameterized velocity field to transport the distribution along the interpolated trajectory, and the method is also known as Flow Matching. (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. Similar to Flow-Matching methods (Albergo and Vanden-Eijnden, 2023; Lipman et al., 2023; Liu, 2022), our approach also computes a deterministic probability transport map. However, the interpolant mapping used in these prior works is generally not the optimal transport interpolation, while our proposed Q-flow is optimized to find the optimal velocity in dynamic OT (see Section 2). Generally, the flow attaining optimal transport can improve model efficiency and generalization performance (Huang et al., 2023), and in this work we experimentally show the benefits in high-dimensional DRE and image-to-image translation.

Computation of OT. Many mathematical theories and computational tools have been developed to tackle the OT problem (Villani et al., 2009; Benamou and Brenier, 2000; Peyré et al., 2019). In this work, we focus on the Wasserstein-2 distance, which suffices for many applications. Compared to non-deep approaches, neural network OT methods enjoy scalability to high dimensional data; However, most works in the literature adopt the *static* OT formulation (Xie et al., 2019; Huang et al., 2021; Morel et al., 2023; Fan et al., 2023; Korotin et al., 2021a, 2023; Amos, 2023). By static OT, we mean the problem that, in Monge formulation, looks for a transport T that minimizes $\mathbb{E}_{x\sim P}\|x - T(x)\|^2$ and satisfies $T_{\#}P = Q$. The concept is versus the *dynamic* OT problem (the Benamou-Brenier equation) (Villani et al., 2009; Benamou and Brenier, 2000), which is less studied computationally, especially in high dimensions, with a few exceptions:

Trajectorynet (Tong et al., 2020) proposed a regularized CNF approach to learn the OT trajectory from a reference distribution P – assumed to be Gaussian (so that the KL can be estimated in the CNF) – to a data distribution, motivated by the application of interpolating cellular distributions in single-cell data. Later, (Tong et al., 2024) proposed to learn the velocity field in the dynamic OT by Flow Matching, assuming that the static OT solutions on mini-batches have been pre-computed. In comparison, we parameterize the flow by a neural ODE and directly solve the Benamou-Brenier equation from finite samples, avoiding any pre-computation of OT couplings. We also allow the two endpoint distributions P and Q to be arbitrary, and only finite samples from each distribution are provided.

Meanwhile, the rectified flow (Liu, 2022), as a form of Flow Matching, is closely related to the OT; however, the iterative refining approach in (Liu et al., 2023) may not guarantee the optimality of the coupling. (Kornilov et al., 2024) proposed to learn dynamic OT using Flow Matching, but the framework relied on input convex neural networks which may have limited expressiveness. In addition, each training step in (Kornilov et al., 2024) requires costly Hessian inversion of the parametrized deep network. Recently, (Shi et al., 2024; Tong et al., 2024) proposed to use diffusion or flow models to solve the Schrödinger Bridge (SB) problem as entropy-regularized dynamic OT. We will experimentally compare with recent neural network OT baselines, both static and dynamic, and including SB baselines, in Section 5, where our model shows better performance e.g., on the image-to-image translation task.

2 PRELIMINARIES

Neural ODE and CNF. Neural ODE (Chen et al., 2018) parameterized 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 parameterized 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. The Benamou-Brenier equation below provides the dynamic formulation of OT (Villani et al., 2009; Benamou and Brenier, 2000)

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

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 first introduce the formulation and training objective in Section 3.1. The training technique consists of

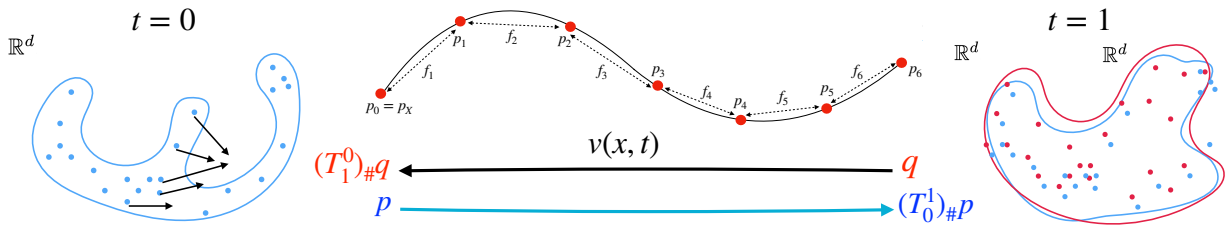


Figure 1: The Q-flow model learns the dynamic OT, an invertible transport map T_0^1 (parametrized by the velocity field $v(x,t)$) between P and Q over the time interval $[0,1]$ with the least transport cost. The push-forwarded distribution $(T_0^1)_\# p$ (as well as $(T_1^0)_\# q$, respectively) is close to the target distribution q (p, respectively).

the end-to-end refinement (Section 3.2) and constructing the initial flow (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 $v(x,t)$ and solve the dynamic OT (2). Our formulation is symmetric from P to Q and vice versa, and the training objective is formally

$$\min_\theta \quad (\mathcal{L}^{P\to Q}(\theta) + \mathcal{L}^{Q\to P}(\theta)), \tag{3}$$

where we call $P \to Q$ the forward direction and $Q \to P$ the reverse direction. The bi-directional training naturally follows the symmetry of the dynamic OT problem (Remark 1).

The dynamic OT (2) on time $[0,1]$ has two terminal conditions, which we propose to relax by a KL divergence loss (see, e.g., (Ruthotto et al., 2020)). Then the training loss in the forward direction is

$$\mathcal{L}^{P\to Q}(\theta) = \mathcal{L}_{\mathrm{KL}}^{P\to Q}(\theta) + \gamma \mathcal{L}_T^{P\to Q}(\theta), \tag{4}$$

where the first term $\mathcal{L}_{\mathrm{KL}}^{P\to Q}(\theta)$ represents the relaxed terminal condition and the second term $\mathcal{L}_T^{P\to Q}(\theta)$ is the Wasserstein-2 transport cost to be detailed below; $\gamma > 0$ is a weight parameter, and with small γ the terminal condition is enforced. $\mathcal{L}^{Q\to P}(\theta)$ is similarly defined, see more below. We now derive the finite-sample form of each term in (4).

KL loss. We specify the first term $\mathcal{L}_{\mathrm{KL}}^{P\to Q}$ in loss (4). We define the solution mapping of (1) from s to t as

$$T_s^t(x;\theta) = x(s) + \int_s^t f(x(t'),t';\theta)dt', \tag{5}$$

which is also parameterized 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 $\mathrm{KL}(p_1||q)$. As neither P nor Q is assumed to have a known density, we cannot use the change-of-variable technique from normalizing flow to estimate the KL. We propose to train a logistic classification network $c_1(x;\varphi_c) : \mathbb{R}^d \to \mathbb{R}$ with parameters φ_c, which resembles the training of discriminators in GAN (Goodfellow et al., 2014). The inner-loop training of c_1 is by

$$\begin{aligned}\min_{\varphi_c} \frac{1}{N}\sum_{i=1}^N \log(1 + e^{c_1(T_0^1(X_i;\theta);\varphi_c)}) \\ + \frac{1}{M}\sum_{j=1}^M \log(1 + e^{-c_1(\tilde{X}_j;\varphi_c)}).\end{aligned} \tag{6}$$

The functional optimal c_1^* of the population version of loss (6) equals $\log(q/p_1)$ by direct computation, and as a result, $\mathrm{KL}(p_1||q) = -\mathbb{E}_{x\sim p_1} c_1^*(x)$. Now take the trained classification network c_1 with parameter $\hat{\varphi}_c$, we can estimate the finite sample KL loss as

$$\mathcal{L}_{\mathrm{KL}}^{P\to Q}(\theta) = -\frac{1}{N}\sum_{i=1}^N c_1(T_0^1(X_i;\theta);\hat{\varphi}_c), \tag{7}$$

where $\hat{\varphi}_c$ is the computed minimizer of (6) solved by inner loops. In practice, we will first initialize the flow such that when minimizing (6), the two densities $p_1 = (T_0^1)_\# p$ and q are close, which makes the training of the classification net more efficient.

Wasserstein (W_2) loss. We specify the second term $\mathcal{L}_T^{P\to Q}(\theta)$ in the loss (4). To compute the transport cost $\mathcal{T}$ in (2), 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, see more details in Section 3.2. Defining $h_k = t_k - t_{k-1}$, and $X_i(t;\theta) := T_0^t(X_i;\theta)$, we write the

W_2 loss as

$$\mathcal{L}_T^{P\to Q}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\frac{\|X_i(t_k;\theta) - X_i(t_{k-1};\theta)\|^2}{h_k}, \quad (8)$$

which appears as a finite-difference approximation of $\mathcal{T}$ along time. 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 (8), that is, $\sum_{k=1}^{K}\mathbb{E}_{x\sim\rho(\cdot,t_{k-1})}\|T_{t_{k-1}}^{t_k}(x;\theta)\|^2/h_k$, at optimum can be interpreted as the discrete-time summed (square) Wasserstein-2 distance, that is, $\sum_{k=1}^{K} W_2(\rho(\cdot,t_{k-1}),\rho(\cdot,t_k))^2/h_k$, see (Xu et al., 2022). The W_2 loss encourages a smooth flow from P to Q with a small transport cost, which also guarantees the invertibility of the model in practice when the trained flow approximates the optimal flow in (2).

Training in both directions. The formulation in the reverse direction is similar, where we transport Q-samples $\tilde{X}_j$ from time 1 to 0 using the same neural ODE integrated in reverse time. Specifically, $\mathcal{L}^{Q\to P}(\theta) = \mathcal{L}_{\mathrm{KL}}^{Q\to P}(\theta) + \gamma\mathcal{L}_T^{Q\to P}(\theta)$, and $\mathcal{L}_{\mathrm{KL}}^{Q\to P}(\theta) = -\frac{1}{M}\sum_{j=1}^{M}\tilde{c}_0(T_1^0(\tilde{X}_j;\theta);\hat{\varphi}_{\tilde{c}})$, where $\hat{\varphi}_{\tilde{c}}$ is obtained by inner-loop training of another classification net $\tilde{c}_0(x,\varphi_{\tilde{c}})$ with parameters $\varphi_{\tilde{c}}$ via

$$\begin{aligned}\min_{\varphi_{\tilde{c}}}\ &\frac{1}{M}\sum_{j=1}^{M}\log(1+e^{\tilde{c}_0(T_1^0(\tilde{X}_j;\theta);\varphi_{\tilde{c}})})\\ &+\frac{1}{N}\sum_{i=1}^{N}\log(1+e^{-\tilde{c}_0(X_i;\varphi_{\tilde{c}})}).\end{aligned} \quad (9)$$

The reverse-time W_2 loss is

$$\mathcal{L}_T^{Q\to P}(\theta) = \frac{1}{M}\sum_{j=1}^{M}\left(\sum_{k=1}^{K}\frac{1}{h_k}\|\tilde{X}_j(t_{k-1};\theta) - \tilde{X}_j(t_k;\theta)\|^2\right),$$

where we define $\tilde{X}_j(t;\theta) := T_1^t(\tilde{X}_j;\theta)$.

Remark 1 (Symmetry of dynamic OT and bi-directional training). The Benamou-Brenier formula (2) is "symmetric" with respect to P and Q, in the sense that either setting P to be at time 0 and Q at time 1, or the other way, one will obtain the same solution – transport with the same velocity field $v(x,t)$ that only differs in the direction of time – under generic conditions (e.g. both P and Q have densities). This means that training in the forward or the reversed direction, although having different objectives in appearance, is aiming at the same flow solution at optimum. Thus, the proposed bi-directional training naturally captures the symmetry of the dynamic OT, and can potentially improve the accuracy when training with finitely many samples. We empirically verify the advantage over unidirectional training in Section 5.3.3.

3.2 End-to-end Training of Q-flow

Assuming that the Q-flow net has already been initiated (see more in Section 3.3), we minimize $\mathcal{L}^{P\to Q}$ and $\mathcal{L}^{Q\to P}$ in an alternative fashion per iteration, and the procedure is summarized in Algorithm 1. We call this the "refinement" of the flow, namely to refine the flow trajectory towards the OT one from some initialization. The computational complexity is analyzed in Appendix C.1, where the computational cost is shown to scale with the size of the time grid along the flow and the sample sizes (and number of iterations).

Time integration of flow. In the losses (7) and (8), 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 $t_{k,s}$ that divides each subinterval $[t_{k-1},t_k]$ into S mini-intervals (S is usually 3-5 in our experiments). We compute the time integration of $f(x,t;\theta)$ using four-stage Runge-Kutta 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.2). To further improve efficiency, it is possible to first train the flow $f(x,t;\theta)$ on a coarse time grid to warm-start the later training on a refined grid. One can also adopt an adaptive time grid, e.g., by enforcing equal W_2 movement on each subinterval $[t_{k-1},t_k]$, so that the representative points are more evenly distributed along the flow trajectory and the learning of the flow model may be improved (Xu et al., 2023).

Algorithm 1 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 $f(x(t),t;\theta)$

for Iter $= 1,\dots,$ `Tot` **do**
  (If Iter $= 1$) Train c_1 by minimizing (6) for E_0 epochs.
  for epoch $= 1,\dots,E$ **do** {▷ $P\to Q$ refinement}
    Update θ of $f(x(t),t;\theta)$ by minimizing $\mathcal{L}^{P\to Q}$.
    Update c_1 by minimizing (6) for E_{in} epochs.
  end for
  (If Iter $= 1$) Train $\widetilde{c}_0$ by minimizing (9) for E_0 epochs.
  for epoch $= 1,\dots,E$ **do** {▷ $Q\to P$ refinement}
    Update θ of $f(x(t),t;\theta)$ by minimizing $\mathcal{L}^{Q\to P}$.
    Update $\widetilde{c}_0$ by minimizing (9) for E_{in} epochs.
  end for
end for

Inner-loop training of c_1 and $\tilde{c}_0$. Given a warm-started initialization of the flow, 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 frequently updating c_1 and $\widetilde{c}_0$ in lines 5 and 10 of Algorithm 1 are crucial for successful end-to-end training of Q-flow net. Specifically, 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. Then one will need to retrain c_1 and $\widetilde{c}_0$ timely (e.g., once every several steps of training θ) 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 (i.e., frequent updates) if needed.

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). The proposed end-to-end training 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 is available. Generally, when only two data sets $\boldsymbol{X}, \tilde{\boldsymbol{X}}$ are given, one can adopt various existing generative flows to obtain an initial flow. In this work, we consider two approaches: (i) by a concatenation of two CNF models, and (ii) by distribution interpolant neural networks. See Appendix C.2 for details. Any other initialization scheme is compatible with the proposed end-to-end training.

4 INFINITESIMAL DENSITY RATIO ESTIMATION (DRE)

The DRE problem, namely estimating $\log(q(x)/p(x))$ is a fundamental task in statistical inference. (Rhodes et al., 2020; Choi et al., 2022) showed that an interpolated sequence of distributions from P to Q can be used to compute the DRE (reviewed in more detail in Appendix A). Following the same idea, we propose to train an additional continuous-time neural network, called *flow-ratio net*, by minimizing a classification loss to distinguish distributions on neighboring time stamps along the trained Q-flow net trajectory. We will show in Section 5.3 that using the OT trajectory provided by the trained Q-flow net can benefit the accuracy of DRE.

Let $p(x,t) = (T_t)_\# p$ and T_t is the transport induced by the trained Q-flow net. By the relation (A.2), 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, which we call the flow-ratio net, and the network $r(x,t;\theta_r)$ is to perform DRE given the OT trajectory from a pre-trained Q-flow 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.3), we expect that the integral

$$\begin{aligned} R_k(x;\theta_r) :&= \int_{t_{k-1}}^{t_k} r(x,t;\theta_r)dt \\ &\approx \int_{t_{k-1}}^{t_k} \partial_t \log p(x,t)dt \\ &= \log(p(x,t_k)/p(x,t_{k-1})). \end{aligned} \tag{10}$$

Since logistic classification can reveal 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, for $k = 1, \cdots, L$,

$$\begin{aligned} L_k^{P\to Q}(\theta_r) = &\frac{1}{N}\sum_{i=1}^N \log(1 + e^{R_k(X_i(t_{k-1});\theta_r)}) \\ &+ \frac{1}{N}\sum_{i=1}^N \log(1 + e^{-R_k(X_i(t_k);\theta_r)}), \end{aligned} \tag{11}$$

where $X_i(t) := T_0^t(X_i)$ and T_0^t is computed by integrating the trained Q-flow net. 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, and then replacing the second term in (11) 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 flow-ratio net is by

$$\min_{\theta_r} \sum_{k=1}^L L_k^{P\to Q}(\theta_r) + L_k^{Q\to P}(\theta_r). \tag{12}$$

When trained successfully, 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 $[0,1]$, and in particular, the integral over $[0,1]$ yields the desired log density ratio $\log(q/p)$ as shown in (A.2). See Algorithm A.2 for more details.

5 EXPERIMENTS

We demonstrate the effectiveness of the proposed method on several downstream tasks: OT base-

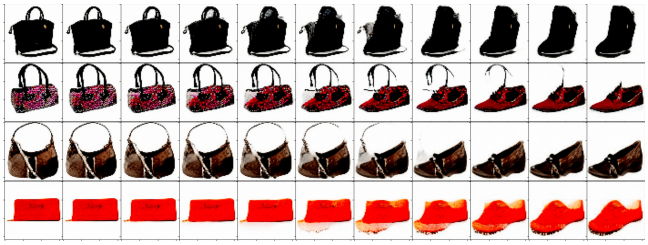
Table 1: OT benchmarks using Gaussian mixtures (left) and CelebA64 images (right). Metric values ($\mathcal{L}^2$-UVP, cos) are shown in cells, with lower $\mathcal{L}^2$-UVP and higher cos being better. Results of MM, MMv1, MMv2, MM:R, and W2 are quoted from (Korotin et al., 2021b) for comparisons.

(a) Gaussian mixtures

Dimension	**64**	**128**	**256**
Q-flow (Ours)	**(4.00, 0.98)**	**(2.12, 0.99)**	**(1.97, 0.99)**
OTCFM	(4.64, 0.97)	(2.78, 0.99)	(3.02, 0.98)
MMv1	(8.1, 0.97)	(2.2, 0.99)	(2.6, 0.99)
MMv2	(10.1, 0.96)	(3.2, 0.99)	(2.7, 0.99)
W2	(7.2, 0.97)	(2.0, 1.00)	(2.7, 1.00)

(b) CelebA64 images

Ckpt	**Early**	**Mid**	**Late**
Q-flow (Ours)	**(0.87, 0.99)**	**(0.27, 0.97)**	**(0.12, 0.97)**
NOT	(0.99, 0.99)	(0.34, 0.96)	(0.12, 0.96)
OTCFM	(1.2, 0.99)	(0.39, 0.96)	(0.16, 0.95)
MM	(2.2, 0.98)	(0.9, 0.90)	(0.53, 0.87)
MM:R	(1.4, 0.99)	(0.4, 0.96)	(0.22, 0.94)



(a) Handbag → shoes

(b) CelebA male → female

Figure 2: Image-to-image translation: the trajectory of samples (in rows) from intermediate distributions of the Q-flow in the VAE latent space, as it pushes forward the base distribution (leftmost column) to the target distribution (rightmost column). Figure (a) shows the transition from handbag to shoe. Figure (b) shows the transition from CelebA male to female.

lines (Section 5.1), image-to-image translation (Section 5.2), high dimensional DRE (Section 5.3). Additional details, including hyperparameter choices and sensitivity, are provided in Appendix B. Code can be found at `https://github.com/hamrel-cxu/FlowOT`.

5.1 High-dimensional OT Baselines

We compare our Q-flow with popular OT baselines on the OT benchmark (Korotin et al., 2021b). Competing methods are MM (Dam et al., 2019), MMv1 (Taghvaei and Jalali, 2019), MM:R (Makkuva et al., 2020), MMv2 (Fan et al., 2021), W2 (Korotin et al., 2021c), NOT (Korotin et al., 2023), and OTCFM (Tong et al., 2024). The goal is to match outputs from a trained OT map as closely as possible with those from the ground truth in the benchmark. Two metrics, $\mathcal{L}^2$-UVP in (A.3) and cos in (A.4), are used to evaluate performance, where lower $\mathcal{L}^2$-UVP and higher cos values indicate better performance. Details of the setup are provided in Appendix B.1. Table A.1 reports the training time of each method, where Q-flow is computationally competitive.

High-dimensional Gaussian mixtures. The goal is to transport between high-dimensional Gaussian mixtures optimally. We vary dimension $d \in \{64, 128, 256\}$, and in each dimension, the distribution P is a mixture with three components, and Q is a weighted average of two 10-component Gaussian mixtures. Table 1a shows that our Q-flow consistently reaches lower $\mathcal{L}^2$-UVP and higher or equal cos than OT baselines, indicating comparable or better performance on this example.

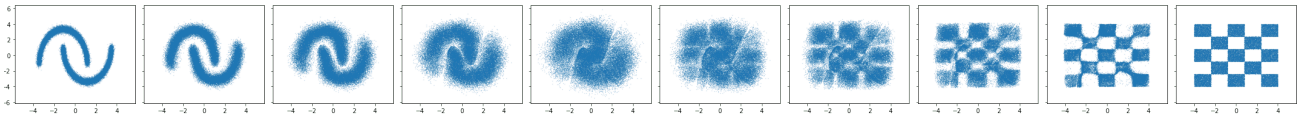
CelebA64 images. The goal is to align CelebA64 faces (Liu et al., 2015) (denoted as Q) with faces generated by a pre-trained WGAN-QC (Liu et al., 2019) generator (denoted as P_{Ckpt} for different checkpoints). In particular, three checkpoints (Ckpt) of the WGAN-QC are considered (i.e., Ckpt $\in$ {Early, Mid, Late}), where P_{Early} contains generated faces that are the most blurry and faces from P_{Late} are closest to true faces among the three. Quantitatively, results in Table 1b show that Q-flow outperforms other OT baselines. Figure A.1 in the appendix further shows high-quality faces as a result of using the trained Q-flow net on samples from P_{Ckpt} for different Ckpt.

5.2 Image-to-image Translation

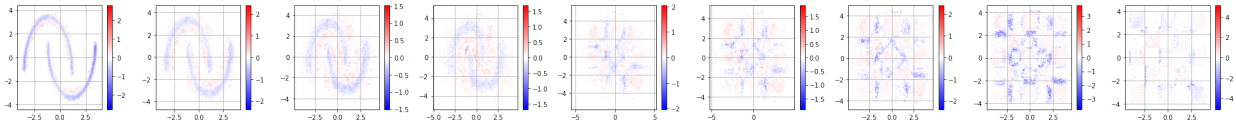
We use Q-flow to learn the continuous-time OT between distributions of two sets of RGB images and

Table 2: Image-to-image translation: FID on the test sets, lower is better. FIDs of Disco GAN, Cycle GAN, and NOT are quoted from (Korotin et al., 2023).

	Q-flow (ours)	OTCFM	Re-flow	SBCFM	DSBM	W2GN	MM:R	Disco GAN	Cycle GAN	NOT
Handbag → shoes	**12.34**	15.96	25.92	17.22	25.71	34.23	33.04	22.42	16.00	13.77
CelebA male → female	**9.66**	9.76	20.24	11.32	25.82	15.23	12.34	35.64	17.74	13.23



(a) Trajectory from P (two-moon) to Q (checkerboard)



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

Figure 3: Q-flow trajectory and corresponding log-ratio estimation. **Top**: intermediate distributions by Q-flow net. **Bottom**: corresponding log-ratio estimated by flow-ratio net. Blue color indicates negative estimates of the difference $\log(p(x, t_k)/p(x, t_{k-1}))$ evaluated at the common support of the neighboring densities.

compare with Disco GAN (Kim et al., 2017), Cycle GAN (Zhu et al., 2017), W2GN (Korotin et al., 2021a), MM:R, NOT, Re-flow (Liu et al., 2023), OTCFM, SBCFM (Tong et al., 2024), and DSBM (Shi et al., 2024). The first set contains handbags (Zhu et al., 2016) and shoes (Yu and Grauman, 2014), and the second set contains CelebA male and female images. We denote handbags/males as P and shoes/females as Q. We follow the setup in (Korotin et al., 2023), where the goal of the image-to-image translation task is to conditionally generate shoe/female images by mapping images of handbag/male 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. For a fair comparison, we train OTCFM, Re-flow, SBCFM, DSBM, W2GN, and MM:R in the same latent space for the same number of mini-batches as Q-flow net, and all methods use models of the same size. Additional details are in Appendix B.2.

Figure 2 visualizes continuous trajectories from handbags/males to shoes/females generated by the Q-flow model. We find that Q-flow can capture the style and color nuances of corresponding handbags/males in the generated shoes/females as the flow model continuously transforms handbag/male images. Figure A.2 and A.3 in the appendix show additional generated shoes/females from handbags/males, respectively. Quantitatively, FIDs between generated and true images in the test set, as shown in Table 2 indicate Q-flow performs better than all baselines. Meanwhile, as our Q-flow model learns a *continuous* transport map from source to target domains, it directly provides the gradual interpolation between source and target samples along the dynamic OT trajectory (Figure 2).

5.3 High-dimensional DRE

We show the benefits of our flow-ratio net proposed in Section 4 on various DRE tasks, where flow-ratio net leverages the flow trajectory of a trained Q-flow net. In the experiments 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 $k = 1, \dots, L$ builds a bridge between P and Q.

5.3.1 Toy Data in Two-dimension

Gaussian mixtures. We simulate P and Q as two Gaussian mixture models with three and two components, respectively, see additional details in Appendix B.3.1. We compare ratio estimates $\hat{r}(x)$ with the true values $r(x)$. The results are shown in Figure A.4. We see from the top panel that the mean absolute error (MAE) of Ours is the smallest, and Ours also incurs a smaller $\max_x |\hat{r}(x) - r(x)|$. This is consistent with the

Table 3: BPD on the energy-based modeling of MNIST (lower is better). Results for DRE-∞ are from (Choi et al., 2022), and results for one-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

closest resemblance of Ours to the ground truth (first column) in the bottom panel. In comparison, DRE-∞ tends to over-estimate $r(x)$ on the support of Q, while TRE and one ratio under-estimate $r(x)$ on the support of P. As both the DRE-∞ and TRE models use the linear interpolant scheme (A.8), the result suggests the benefit of using Q-flow trajectory for DRE.

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.3.1. For this more challenging case, the linear interpolation scheme (A.8) creates a bridge between P and Q as shown in Figure A.9. The flow visually differs from the one obtained by the trained Q-flow net, as shown in Figure 3(a), and the latter is trained to minimize the transport cost. The result of flow-ratio net is shown in Figure 3(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. Figure A.5 additionally shows the estimates $\log q(x) - \log p(x)$ using flow-ratio net.

5.3.2 High-dimensional Mutual Information

We estimate the mutual information (MI) between two correlated random variables in dimensions $d \in \{40, 80, 160, 320\}$, following the setup in (Rhodes et al., 2020). Additional details can be found in Appendix B.3.2. As shown in Figure A.6, the estimated MI by our method aligns well with the ground truth MI values, reaching nearly identical performance as DRE-∞ and outperforming the other two baselines.

5.3.3 Energy-based Modeling of MNIST

We apply our approach in evaluating and improving an energy-base model (EBM) on the MNIST dataset (LeCun and 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 pre-trained energy-based generative models: Gaussian, Copula, and RQ-NSF (Durkan et al., 2019). The performance of DRE is measured using the "bits per dimension" (BPD) metric in (A.7). Additional details are in Appendix B.3.3. The results show that Ours reaches the improved performance in Table 3 against baselines: it consistently reaches the smallest BPD 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 the trajectory of improved samples from Q to $\tilde{Q}$ for RQ-NSF using the trained Q-flow in Figure A.7. Figure A.8 shows additional improved digits for all three specifications of Q. Lastly, Table A.3 shows the empirical benefit of bi-directional over uni-directional training of Q-flow for this task.

6 DISCUSSION

The Q-flow model developed in this work is optimized to find the dynamic OT transport between two distributions to learn from data samples. One limitation is the computational cost associated with the neural ODE training. To save computation, one can explore more advanced time discretization schemes, such as adaptive time grids, as well as customized neural ODE solvers (Shaul et al., 2024). One can try to develop a simulation-free approach under the proposed framework; more efficient computation can also be achieved via progressive distillation (Salimans and Ho, 2022) from the trained OT trajectory. Meanwhile, there remain open theoretical questions, such as the theoretical guarantee of learning the OT trajectory. In particular, the boundary condition in the Benamou-Brenier equation is handled by KL divergences in the current model, and it would be helpful to analyze the consistency of such an approach, especially under a finite-sample scenario. It would also be interesting to explore alternative distribution divergences (other than KL) to handle the two endpoint distributions and analyze them under a more general framework. For the empirical results, extending to a broader class of applications and additional real datasets will be useful.

Acknowledgments

The work is supported by NSF DMS-2134037. C.X. and Y.X. are partially supported by an NSF CAREER CCF-1650913, CMMI-2015787, CMMI-2112533, DMS-1938106, DMS-1830210, and the Coca-Cola Foundation. X.C. is also partially supported by NSF DMS-2237842 and Simons Foundation MPS-MODL-00814643.

References

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

Brandon Amos. On amortizing convex conjugates for optimal transport. In *The Eleventh International Conference on Learning Representations*, 2023.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 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*, pages 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*, pages 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*, pages 2552–2573. PMLR, 2022.

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*, pages 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.

Nhan Dam, Quan Hoang, Trung Le, Tu Dinh Nguyen, Hung Bui, and Dinh Phung. Three-player wasserstein gan via amortised duality. In *International Joint Conference on Artificial Intelligence 2019*, pages 2202–2208. Association for the Advancement of Artificial Intelligence (AAAI), 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

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*, pages 12873–12883, 2021.

Jiaojiao Fan, Amirhossein Taghvaei, and Yongxin Chen. Scalable computations of wasserstein barycenter via input convex neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1571–1581. PMLR, 18–24 Jul 2021.

Jiaojiao Fan, Shu Liu, Shaojun Ma, Hao-Min Zhou, and Yongxin Chen. Neural monge map estimation and its applications. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. Featured Certification.

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*, pages 3154–3164. PMLR, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 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.

Chin-Wei Huang, Ricky T. Q. Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*, 2021.

Han Huang, Jiajia Yu, Jie Chen, and Rongjie Lai. Bridging mean-field games and normalizing flows with trajectory regularization. *Journal of Computational Physics*, page 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*, pages 1125–1134, 2017.

Ray Jiang, Aldo Pacchiano, Tom Stepleton, Heinrich Jiang, and Silvia Chiappa. Wasserstein fair classification. In *Uncertainty in artificial intelligence*, pages 862–872. PMLR, 2020.

Masahiro Kato and Takeshi Teshima. Non-negative bregman divergence minimization for deep direct density ratio estimation. In *International Conference on Machine Learning*, pages 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*, pages 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*, pages 1857–1865. PMLR, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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.

Nikita Maksimovich Kornilov, Petr Mokrov, Alexander Gasnikov, and Alexander Korotin. Optimal flow matching: Learning straight trajectories in just one step. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, Alexander Filippov, and Evgeny Burnaev. Do neural optimal transport solvers work? a continuous wasserstein-2 benchmark. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021b.

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

Alexander Korotin, Daniil Selikhanovych, and Evgeny Burnaev. Neural optimal transport. In *The Eleventh International Conference on Learning Representations*, 2023.

Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

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.

Huidong Liu, Xianfeng Gu, and Dimitris Samaras. Wasserstein gan with quadratic transport cost. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4832–4841, 2019.

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.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

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

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

Guillaume Morel, Lucas Drumetz, Simon Benaïchouche, Nicolas Courty, and François Rousseau. Turning normalizing flows into monge maps with geodesic gaussian preserving flows. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

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, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 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.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

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, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4905–4916. Curran Associates, Inc., 2020.

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.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Neta Shaul, Juan Perez, Ricky T. Q. Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. *Advances in Neural Information Processing Systems*, 36, 2024.

Chiappa Silvia, Jiang Ray, Stepleton Tom, Pacchiano Aldo, Jiang Heinrich, and Aslanides John. A general approach to fairness with optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3633–3640, 2020.

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.

Amirhossein Taghvaei and Amin Jalali. 2-wasserstein approximation via restricted convex potentials with application to improved training for gans. *arXiv preprint arXiv:1902.07197*, 2019.

L Theis, A van den Oord, and M Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR 2016)*, pages 1–10, 2016.

Alexander Tong, Jessie Huang, Guy Wolf, David Van Dijk, and Smita Krishnaswamy. Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics. In *International conference on machine learning*, pages 9526–9536. PMLR, 2020.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. Expert Certification.

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*, pages 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, 2022. doi: 10.1109/JSAIT.2022.3221864.

Chen Xu, Xiuyuan Cheng, and Yao Xie. Normalizing flow neural networks by JKO scheme. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 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*, pages 597–613. Springer, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes. See Section 3 and 4.]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes. See Section 3.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes. Code is uploaded in supplement.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes. See Section 5 and Appendix B.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes. See Appendix B.]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes. See Section 5 and Appendix B.]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes. See Appendix B.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes.]
   (b) The license information of the assets, if applicable. [Yes.]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Infinitesimal density ratio estimation

We first introduce related works of DRE (Section A.1) and DRE preliminaries (Section A.2). We then present the complete algorithm for our proposed flow-ratio net in Section A.3.

A.1 DRE literature

Density ratio estimation between distributions P and Q is a fundamental problem in statistics and machine learning (Meng and 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 and Sugiyama, 2009), outlier detection (Kato and 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 and 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 an unoptimal construction of the intermediate distributions between P and Q. 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 5). In computation, (Choi et al., 2022) applies score matching to compute the infinitesimal change of log-density. The proposed flow-ratio net is based on classification loss training using a fixed time grid which avoids score matching and is computationally lighter (Section A.3).

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

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

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 (A.1). Specifically, suppose the time-parameterized density $p(x,t)$ is differentiable on $t \in [0,1]$ with $p(x,0) = p$ and $p(x,1) = q$, then

$$\log(q(x)/p(x)) = \log p(x,1) - \log p(x,0) = \int_0^1 \partial_t \log p(x,t) dt. \tag{A.2}$$

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

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

A.3 Algorithm and computational complexity

Algorithm A.2 presents the complete flow-ratio algorithm. 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 flow-ratio net $r(x,t;\theta_r)$ and use it as a warm start 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 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 lines 4-6 conduct E epochs in total. Assume each time integral in R_k (10) 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 (12).

Algorithm A.2 Infinitesimal DRE training via flow-ratio net

input Training samples $\boldsymbol{X} \sim P$ and $\widetilde{\boldsymbol{X}} \sim Q$; pre-trained Q-flow net $f(x(t), t; \theta)$; hyperparameters: $\{\{t_k\}_{k=1}^L$, `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 $= 1, \ldots,$ `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 (12).
7: **end for**

B Additional experimental details

When training all networks, we use the Adam optimizer (Kingma and Ba, 2015). Unless otherwise specified, we use an initial learning rate of 0.001.

To provide an initialized flow, we either train a continuous-time flow via (Albergo and Vanden-Eijnden, 2023) in Sections 5.1-5.2 or concatenate two CNFs (each trained via (Xu et al., 2023)) in Section 5.3.

B.1 High-dimensional OT baselines

We summarize the setup and comparison metrics based on (Korotin et al., 2021b). To assess the quality of a trained transport map $\hat{T} : \mathbb{R}^d \to \mathbb{R}^d$ from P to Q against a ground truth $T^* : \mathbb{R}^d \to \mathbb{R}^d$, the authors use two metrics: *unexplained variance percentage* ($\mathcal{L}^2$-UVP) (Korotin et al., 2021a) and *cosine similarity* (cos), which are defined as

$$\mathcal{L}^2\text{-UVP} = 100 \cdot \frac{\mathbb{E}_{x\sim P}\|\hat{T}(x) - T^*(x)\|_2^2}{\text{Var}(Q)}\%. \tag{A.3}$$

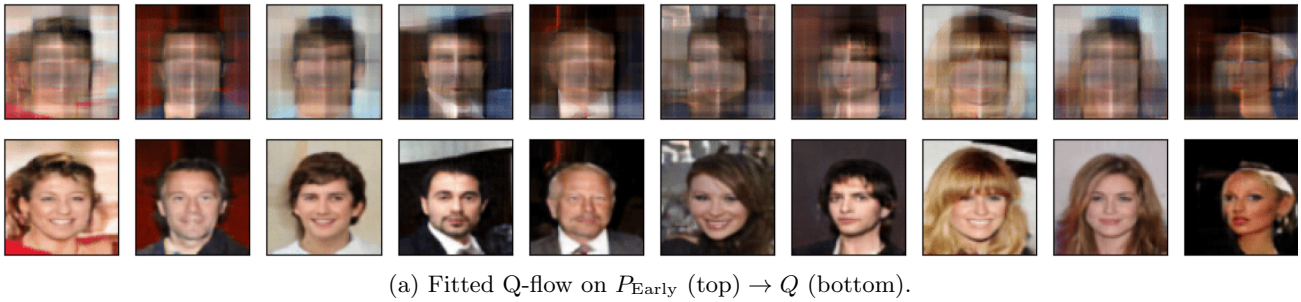
$$\cos = \frac{\mathbb{E}_{x\sim P}\left\langle \hat{T}(x) - x, T^*(x) - x \right\rangle_{\ell_2}}{\mathbb{E}_{x\sim P}\|\hat{T}(x) - x\|_2 \cdot \mathbb{E}_{x\sim P}\|T^*(x) - x\|_2}. \tag{A.4}$$

The metrics are evaluated using 2^{14} random samples from P.

High-dimensional Gaussian mixtures. In each dimension d, the distribution P is a mixture of three Gaussians. The distribution Q is constructed as follows: first, the authors construct two Gaussian mixtures Q_1 and Q_2 with 10 components. Then, they train approximate transport maps $\nabla\psi_i \# P \approx Q_i$, where ψ_i is trained via (Korotin et al., 2021c). The target Q is obtained as $Q = \frac{1}{2}(\nabla\psi_1 + \nabla\psi_2)\#P$.

In Q-flow, the flow architecture $f(x(t), t; \theta)$ consists of fully connected layers of dimensions $2d-4d-8d-4d-2d$ with ReLU activation. The initialization of the flow is done by the method of Interflow (Albergo and Vanden-Eijnden, 2023), where at each step, we draw random batches of 2048 samples from P and Q. We trained the initialized flow for 50K steps. To apply Algorithm 1, we let $\gamma = 0.1$, $t_k = k/5$ for $k = 0, \ldots, 5$, and `Tot`=1. The classifiers c_1 and $\tilde{c}_0$ consist of fully-connected layers of dimensions $4d-4d-4d-4d$ with ReLU activation. These classifiers are initially trained for 10000 batches with batch size 2048. We train flow parameters θ for 10000 batches in every iteration with a batch size of 2048, and we update the training of c_1 and $\tilde{c}_0$ every 10 batches of training θ to train them for 10 inner-loop batches.

CelebA64 images (Liu et al., 2015) The authors first fit 2 generative models using WGAN-QC (Liu et al., 2019) on the CelebA dataset. They then pick intermediate training checkpoints to product continuous measures $P^k_{\text{Early}}, P^k_{\text{Mid}}, P^k_{\text{Late}}$ for these 2 models ($k = 1, 2$). Then, for each $k \in \{1, 2\}$ and Ckpt $\in$ {Early, Mid, Late}, they use (Korotin et al., 2021c) to fit an approximate transport map $\nabla\psi^k_{\text{Ckpt}}$ such that $\nabla\psi^k_{\text{Ckpt}} \# Q \approx P^k_{\text{Ckpt}}$. The distributions P_{Ckpt} are then obtained as $P_{\text{Ckpt}} = \frac{1}{2}(\nabla\psi^1_{\text{Ckpt}} + \nabla\psi^2_{\text{Ckpt}})\#Q$.



(a) Fitted Q-flow on P_{Early} (top) $\to Q$ (bottom).

(b) Fitted Q-flow on P_{Mid} (top) $\to Q$ (bottom).

(c) Fitted Q-flow on P_{Late} (top) $\to Q$ (bottom).

Figure A.1: Fitted Q-flow between P_{Ckpt} for Ckpt $\in$ {Early, Mid, Late} and Q. In each sub-figure, the first row contains random $x \sim P_{\text{Ckpt}}$ and the second row contains generated faces $T(x)$ using the trained Q-flow net T.

In Q-flow, the flow architecture $f(x(t), t; \theta)$ consists of convolutional layers of dimensions 64-128-128-256-256-512-512, followed by convolutional transpose layers whose filters mirror the convolutional layers. The kernel sizes are 3-4-3-4-3-4-3-3-4-3-4-3-4-3 with strides 1-2-1-2-1-2-1-1-2-1-2-1-2-1. We use the ReLU activation. The initialization of the flow is done by the method of Interflow (Albergo and Vanden-Eijnden, 2023) for 30K steps with 128 batch size. To apply Algorithm 1, we let $\gamma = 1$, $t_k = k/3$ for $k = 0, \ldots, 3$, and `Tot`=1. The architecture of the classifier networks c_1 and $\tilde{c}_0$ are ResNets used in WGAN-QC (Liu et al., 2019). These classifiers are initially trained for 5000 batches with batch size 128. We train flow parameters θ for 10000 batches in every iteration with a batch size of 128, and we update the training of c_1 and $\tilde{c}_0$ every 5 batches of training θ to train them for 1 inner-loop batch.

Training time. Table A.1 reports the training time of different methods on these examples; for a given method, the time is consistent across all examples in a given table because of the same training procedure and hyper-parameters on these examples. The training time of Q-flow does not include the time to pre-train an initialized flow, which is an input to Algorithm 1. Nevertheless, pre-training is light—on examples in Table 1a, pre-training takes roughly 7 minutes (9% of Q-flow training time) and on examples in Table A.1, pre-training takes roughly 18 minutes (12% of Q-flow training time). Thus considering the pre-training will not change the comparison much under this setting.

B.2 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. The dataset of CelebA males P has 90K images and the dataset of CelebA females Q has 110K images, which are (3,64,64) RBG images as well. Following (Korotin et al., 2023), we train on 90% of the total

Table A.1: Wall-clock training time to reach the performance in Table 1a and Table 1b. The unit is in hours on a single A100 GPU.

Table 1a time				
Q-flow	OTCFM	MMv1	MMv2	W2
1.25	1.25	2.5	1.25	1.25

Table 1b time				
Q-flow	NOT	OTCFM	MM	MM:R
2.5	2.75	2.5	2	1.75

data from P and Q and reserve the rest 10% data as the test set. The FIDs are computed between generated shoe/female images (from the test handbag/male images) and true shoe/female 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 $\mathcal{D}$

In Q-flow, 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 initialization of the flow is done by the method of Interflow (Albergo and 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 20K steps.

To apply Algorithm 1, we let $\gamma = 0.05$ (bag-shoe) or $\gamma = 0.1$ (male-female), $t_k = k/10$ for $k = 0, \ldots, 10$, and `Tot`=1. The selection of γ is based on grid searching γ within 1e-5 and 1 to find one that leads to the highest FID on training data. Specifically, we randomly pick two subsets of training images (with the same size as corresponding test sets) from P and Q, obtain the translated images of P images via trained Q-flow net, and

(a) Test handbag images from P

(b) Different generated shoe images by Q-flow model

Figure A.2: Additional test images of handbag (in true P, figure (a)) and corresponding generated shoe images by Q-flow model (in generated Q, figure (b)). To generate different shoes for a given handbag, we sample random latent codes given by the VAE, map them through the trained Q-flow model, and decode them back through the VAE decoder to visualize different generated shoes in the pixel space.

(a) Test male images from P

(b) Different generated female images by Q-flow model

Figure A.3: Additional test images of males (in true P, figure (a)) and corresponding generated female images by Q-flow model (in generated Q, figure (b)). To generate different females for a given male, 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 females in the pixel space.

compute the FID between translated images and images sampled from Q. Meanwhile, the classifiers are initially trained for 4000 batches with batch size 512 and updated every 10 batches of training θ for 20 inner-loop batches. The architecture of the classifier networks c_1 and $\tilde{c}_0$ 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 1-1-2-1-1. 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. Lastly, we train flow parameters θ for 30K batches (bag-shoe) or for 18K batches (male-female) with a batch size of 256.

B.3 High-dimensional density ratio estimation

B.3.1 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

$$\frac{1}{N'}\sum_{i=1}^{N'} |r(X_i) - \hat{r}(X_i)| + \frac{1}{M'}\sum_{j=1}^{M'} |r(\tilde{X}_j) - \hat{r}(\tilde{X}_j)|, \tag{A.5}$$

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:

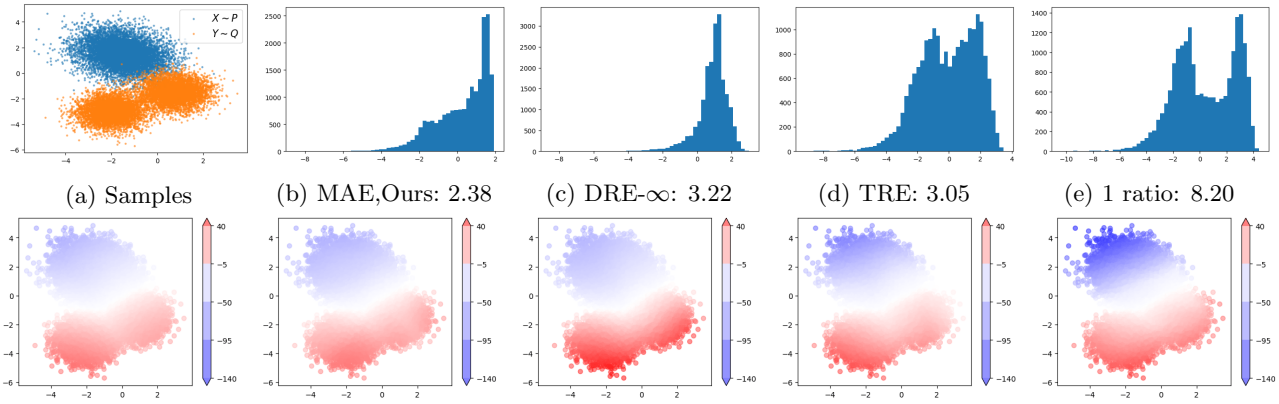
- The flow network $f(x(t), t; \theta_P)$ and $f(x(t), t; \theta_Q)$ consists of fully-connected layers `3→128→128→2`. The Softplus activation with $\beta = 20$ is used. We concatenate t along x to form an augmented input into the network.
- We train the initial flow with a batch size of 2000 for 100 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 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 $\{c_1, \widetilde{c}_0\}$ 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.

flow-ratio: 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 3(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:



(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.4: 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 (A.5) are shown in the captions. **Bottom:** true and estimated $\log(q/p)$ from different models shown under shared color bars.

Table A.2: 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.

moon-to-checkerboard	High-dimensional Gaussians ($d = 320$)	MNIST (Q by RQ-NSF)
7.24e-7	3.44e-5	5.23e-5

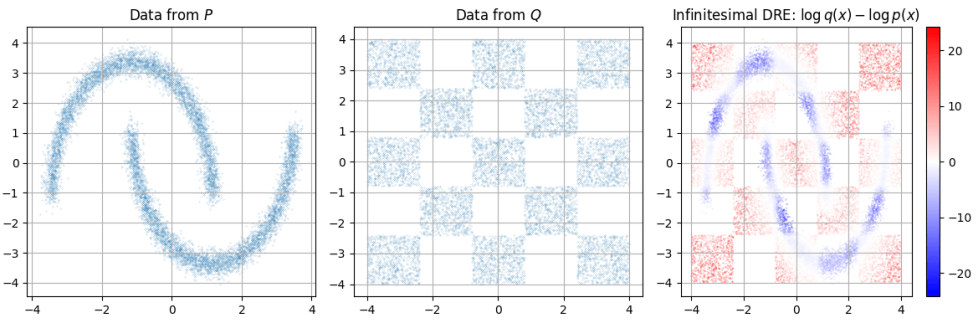


Figure A.5: The flow-ratio net estimation of $\log(q(x)/p(x))$ (right figure), evaluated at the union of test data from P and Q (left and middle figures).

- 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 $\{c_1, \widetilde{c}_0\}$ 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.

flow-ratio: The network consists of fully-connected layers `3→256→256→256→1` with the Softplus activation with $\beta = 20$. The input dimension is three 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.

B.3.2 High-dimensional Mutual Information estimation

Setup: 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 one on the diagonal and 0.8 on the off-diagonal. 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.

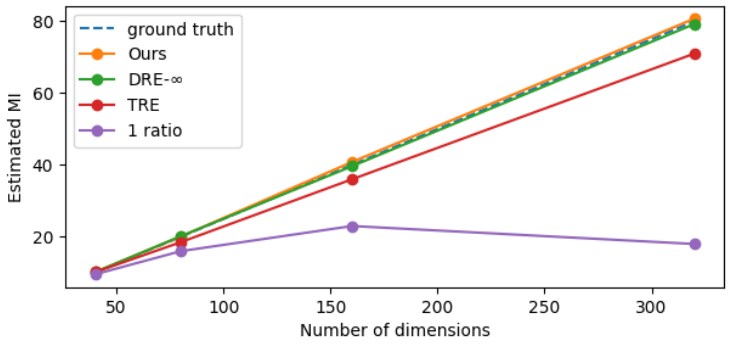


Figure A.6: Estimated MI between two correlated high-dimensional Gaussian random variables.

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:

$$I(U; V) = \mathbb{E}_{p(U,V)} \left[\log \frac{p(U, V)}{p(U)p(V)} \right]. \tag{A.6}$$

Table A.3: DRE performance on the energy-based modeling task for MNIST, reported in BPD and lower, is better. We train flow-ratio net using trajectories given by bi-directional and uni-directional Q-flow. The uni-directional Q-flow only optimizes (4) along the $P \to Q$ direction.

	RQ-NSF	Copula	Gaussian
flow-ratio net using bi-directional Q-flow	1.05	1.14	1.31
flow-ratio net using uni-directional Q-flow	1.08	1.19	1.31

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 (A.6) 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)`→`min(4d,1024)`→`min(4d,1024)`→`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 $\{c_1, \widetilde{c}_0\}$ consists of fully-connected layers $d \to \min(4d, 1024) \to \min(4d, 1024) \to \min(4d, 1024) \to 1$ with the Softplus activation with $\beta = 20$, and it is trained with a batch of 200.

flow-ratio: The network consists of fully-connected layers with dimensions

`(d+1)`→`min(4d,1024)`→`min(4d,1024)`→`min(4d,1024)`→`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.3.3 Energy-based modeling of MNIST

Setup. We discuss how each of the three Q distributions is obtained based on (Rhodes et al., 2020; Choi et al., 2022) and how we apply Q-flow and flow-ratio nets to the problem. Specifically, the MNIST 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

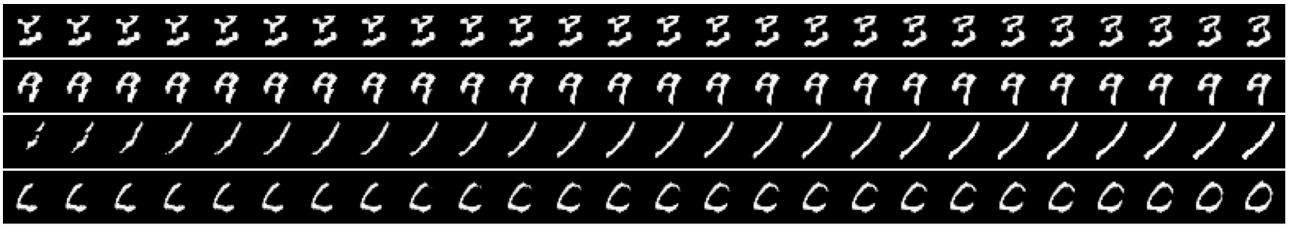
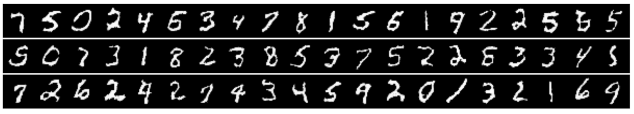


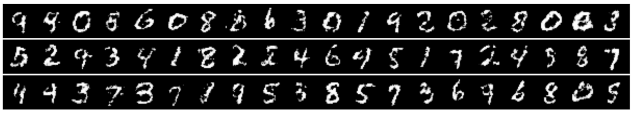
Figure A.7: The trajectory of samples (in rows) from intermediate distributions of the Q-flow, as it pushes forward the base distribution (leftmost column) to the target distribution (rightmost column). The figure shows the improvement of generated digits using the Q-flow.

(a) RQ-NSF: raw samples from Q



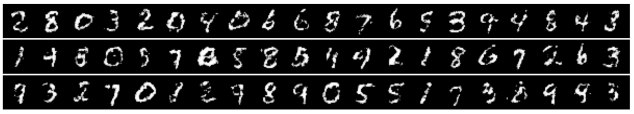
(b) RQ-NSF: improved samples from $\tilde{Q}$

(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.8: 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 $Q = F_{\#}\mathcal{N}(0, I_d)$.

equals $\mathcal{N}(0, I_d)$. Using the trained Q-flow net, we go back to the input space and train the flow-ratio net using the intermediate distributions between P and Q.

The trained 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:

$$\text{BPD} = \frac{1}{N'} \sum_{i=1}^{N'} [-\log \hat{p}(X_i)/(d \log 2)], \tag{A.7}$$

where X_i are N'=10K test images drawn from P. BPD has been a widely used metric in evaluating the performance of generative models (Theis et al., 2016; 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 $h_k = 0.5 \cdot 1.1^{k-1}$.

The hyperparameters for Algorithm 1 are: `Tot`=2, $E_0 = 100$, $E = 500$, $E_{\text{in}} = 2, \gamma = 0.5$. The classification networks $\{c_1, \widetilde{c}_0\}$ 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.

flow-ratio: 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 Hyper-parameter sensitivity

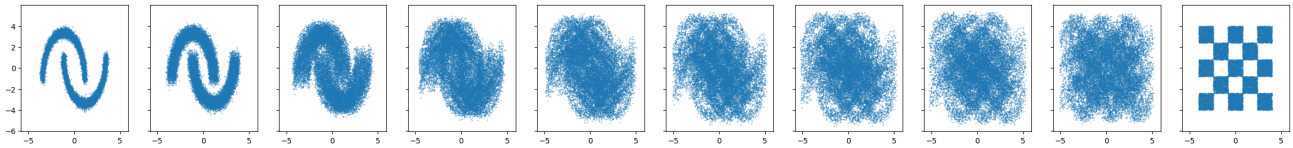


Figure A.9: Bridge construction between P (leftmost) and Q (rightmost) via the linear interpolation scheme (A.8). Specifically, we choose $\alpha_k = k/9$ for $k = 0, \ldots, 9$.

Overall, we did not purposely tune the hyperparameters in Section 5, 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 5.3.3 with RQ-NSF target Q. Table A.4 below presents our method's performance, with the highest BPD (1.062) remaining lower than those by other DRE baselines in Table 3 (the lowest of which is 1.09). Small variations in the table can be 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. In contrast, 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.

Table A.4: 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

C Additional methodology details

C.1 Computational complexity of end-to-end training

We measure the computational complexity by the number of function evaluations of $f(x(t), t; \theta)$ and of the classification nets $\{c_1, \widetilde{c}_0\}$. 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 loss (8) 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 c_1 and $\widetilde{c}_0$ 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.

C.2 Details of flow initialization

(i) By a concatenation of two CNF models.

Each of the two CNF models flows invertibly between P and Z and Z and Q respectively, where $Z \sim \mathcal{N}(0, I_d)$. Any existing neural-ODE CNF models may be adopted for this initialization (Grathwohl et al., 2019; Xu et al., 2023).

(ii) By distribution interpolant neural networks.

Specifically, one can use the linear interpolant mapping in (Rhodes et al., 2020; Choi et al., 2022; Albergo and Vanden-Eijnden, 2023) as below, and train the neural network velocity field $f(x, t; \theta)$ to match the interpolation (Albergo and Vanden-Eijnden, 2023).

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

$$X(t_k) = \sqrt{1 - \alpha_k^2} X(0) + \alpha_k X(1), \tag{A.8}$$

where α_k forms an increasing sequence from 0 to 1. An illustration of $\{X(t_k)\}$ is given in Figure A.9.