Density Ratio Estimation via Infinitesimal Classification

Kristy Choi* **Chenlin Meng*** **Yang Song** **Stefano Ermon**
Computer Science Department, Stanford University

Abstract

Density ratio estimation (DRE) is a fundamental machine learning technique for comparing two probability distributions. However, existing methods struggle in high-dimensional settings, as it is difficult to accurately compare probability distributions based on finite samples. In this work we propose DRE-∞, a divide-and-conquer approach to reduce DRE to a series of easier subproblems. Inspired by Monte Carlo methods, we smoothly interpolate between the two distributions via an infinite continuum of intermediate bridge distributions. We then estimate the instantaneous rate of change of the bridge distributions indexed by time (the "time score")—a quantity defined analogously to data (Stein) scores—with a novel time score matching objective. Crucially, the learned time scores can then be integrated to compute the desired density ratio. In addition, we show that traditional (Stein) scores can be used to obtain integration paths that connect regions of high density in both distributions, improving performance in practice. Empirically, we demonstrate that our approach performs well on downstream tasks such as mutual information estimation and energy-based modeling on complex, high-dimensional datasets.

1 INTRODUCTION

Machine learning algorithms often require a way to compare and contrast two probability distributions $q(\mathbf{x})$ and $p(\mathbf{x})$, given a set of finite samples from each distribution. A natural quantity for such a task is the likelihood ratio $r(\mathbf{x}) = q(\mathbf{x})/p(\mathbf{x})$ of the two densities (Nguyen et al., 2007; Sugiyama et al., 2008), which leads to the problem of *density ratio estimation (DRE).* DRE enjoys a wide range of applications such as generative modeling (Goodfellow et al., 2014; Nowozin et al., 2016), representation learning and mutual information estimation (Oord et al., 2018; Belghazi et al., 2018; Poole et al., 2019; Song and Ermon, 2019a), domain adaptation (Gretton et al., 2009; Yamada et al., 2013), importance sampling (Meng and Wong, 1996; Gelman and Meng, 1998; Neal, 2001; Sinha et al., 2020; Yao et al., 2020), and propensity score matching for causal inference (Abadie and Imbens, 2016; Shalit et al., 2017; Johansson et al., 2018).

Despite its widespread use, accurate DRE from finite samples is challenging in high dimensions. A naive construction of an estimator for this likelihood ratio can require a number of samples exponential in the Kullback-Leibler (KL) divergence of the two densities to be accurate (Chatterjee and Diaconis, 2018; McAllester and Stratos, 2020). Therefore, prior works have found success in a divide-and-conquer approach (Rhodes et al., 2020). They split the global problem into a sequence of easier DRE subproblems for $T > 0$ intermediate bridging distributions that are closer to each other, thereby "shrinking" the gap between $p(\mathbf{x})$ and $q(\mathbf{x})$. Rhodes et al. (2020) takes a discriminative approach by training multiple classifiers—one for each pair of bridge distributions—and aggregates their outputs to obtain the desired ratio estimates. Although they demonstrate that using more intermediate distributions helps performance, naively increasing the number of bridging distributions T is undesirable. Not only does the model size and complexity grow linearly with the number of bridges, but also the approach requires evaluating more classifiers at test time, which is computationally expensive.

To address such limitations, we draw inspiration from annealed importance sampling and path sampling (Neal, 1993; Meng and Wong, 1996; Gelman and Meng, 1998) to generalize this divide-and-conquer approach by considering its limiting case. We connect $q(\mathbf{x})$ and $p(\mathbf{x})$ by constructing an *infinite* number of bridge distributions $p_t(\mathbf{x})$—indexed by a *continuous* "time" variable

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

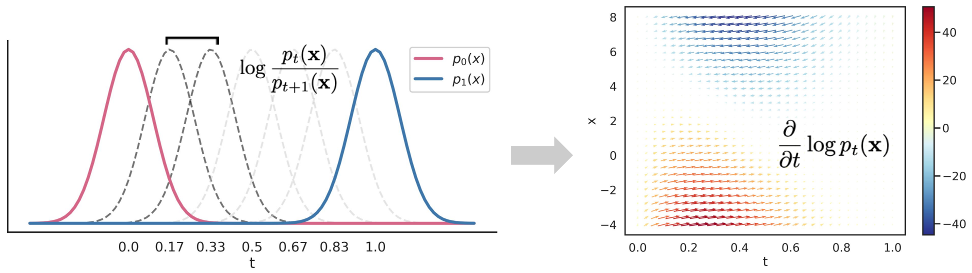


Figure 1: An overview of DRE-∞'s time score matching framework. Instead of bridging $p_0(\mathbf{x}) \equiv q(\mathbf{x})$ and $p_1(\mathbf{x}) \equiv p(\mathbf{x})$ with a finite number of bridges, we smoothly interpolate and estimate the instantaneous rate of change of each intermediate distribution $\frac{\partial}{\partial t} \log p_t(\mathbf{x})$. The x-coordinates of the vector field on the right denote the time scores, while the y-coordinates denote data scores. Arrows are colored by the time score values.

$t \in [0, 1]$—via an interpolation mechanism. This gives our method DRE-∞ its namesake. The key to our approach is to estimate for each location $\mathbf{x}$ the instantaneous rate of change of the intermediate log-density $p_t(\mathbf{x})$ along this path of bridging distributions. The rate of change $\frac{\partial}{\partial t} \log p_t(\mathbf{x})$ measures how each intermediate density is locally changing along a prescribed trajectory in distribution space over time. As this is in direct analogy to the traditional (Stein) score or *data score* ($\nabla_{\mathbf{x}} \log p(\mathbf{x})$), which measures how a density is changing over its input domain (Hyvärinen, 2005; Kingma and LeCun, 2010), we call this quantity $\frac{\partial}{\partial t} \log p_t(\mathbf{x})$ the *time score*. The intuition is that since $\frac{\partial}{\partial t} \log p_t(\mathbf{x}) \approx (\log p_{t+\Delta t}(\mathbf{x}) - \log p_t(\mathbf{x}))/\Delta t = (\log \frac{p_{t+\Delta t}(\mathbf{x})}{p_t(\mathbf{x})})/\Delta t$, the time score characterizes the log density ratio between two distributions with an infinitesimal gap (Δt). This allows us to compute the original density ratio $r(\mathbf{x})$ by *integrating* (rather than summing) the time scores over $t \in [0, 1]$. Figure 1 illustrates an overview of our DRE-∞ method.

Because the true underlying time scores are unknown, we introduce a framework to estimate them from data. We propose a new *time score matching* objective to efficiently train a neural network for learning the time scores. We additionally prove that this time score matching objective is equivalent to solving a series of "infinitesimal classification" tasks between two extremely close bridge distributions. Perhaps counterintuitively, DRE-∞ generalizes TRE to an *infinite number* T of bridge distributions while simultaneously overcoming its various computational limitations. We also show that this framework also naturally allows for incorporating auxiliary information from the data scores ($\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$) of the bridge distributions that is helpful for estimating density ratios more accurately in practice. To do so, we introduce a hybrid training objective that jointly learns both the data and time scores, which allows for the construction of integration paths connecting regions of high data density for both $q(\mathbf{x})$ and $p(\mathbf{x})$. Empirically, we demonstrate the efficacy of our approach on downstream tasks which require access to accurate density ratios, such as mutual information estimation and energy-based modeling on complex, high-dimensional data.

In summary, the contributions of our work are:

1. We propose DRE-∞, a DRE technique that smoothly interpolates between two distributions and involves learning the rate of change of the intermediate log densities (time scores).
2. We introduce a novel framework to learn the time scores from data—time score matching— that allows for the scalable estimation of such time scores, and demonstrate how to leverage black-box numerical integrators to obtain density ratios.
3. We demonstrate how to leverage data scores via a hybrid objective to improve our density ratio estimates in practice, by connecting regions of high data density in both distributions.

2 PRELIMINARIES

Notation and Problem Setup. Let $p(\mathbf{x})$ and $q(\mathbf{x})$ be two unknown distributions over $\mathcal{X} \in \mathbb{R}^D$, for which we have access to i.i.d. samples $\mathcal{D}_P = \{\mathbf{x}_i\}_{i=1}^N \sim p(\mathbf{x})$ and $\mathcal{D}_Q = \{\mathbf{x}_i\}_{i=1}^N \sim q(\mathbf{x})$. The goal of density ratio estimation (DRE) is to accurately estimate $r(\mathbf{x}) = q(\mathbf{x})/p(\mathbf{x})$ given $\mathcal{D}_P$ and $\mathcal{D}_Q$.

DRE via Probabilistic Classification. A well-known technique for DRE is probabilistic classification, where a binary classifier $h_\theta : \mathcal{X} \to [0, 1]$ is trained to discriminate between two sets of samples $\mathcal{D}_P$ and $\mathcal{D}_Q$ (Sugiyama et al., 2008; Menon and Ong, 2016). Each "dataset" from a particular distribution is assigned a pseudolabel of either $y = 0$ or $y = 1$ depending on its source. Once the classifier has been trained, the corresponding ratios can be recovered from its class probabilities via Bayes Rule:

$$r(\mathbf{x}) = \frac{q(\mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid y=1)}{p(\mathbf{x} \mid y=0)} = \frac{h_\theta^*(\mathbf{x})}{1-h_\theta^*(\mathbf{x})} \qquad (1)$$

where h_θ^* denotes the Bayes optimal classifier. Despite its elegant simplicity, this method fails in settings where $q(\mathbf{x})$ and $p(\mathbf{x})$ are sufficiently different. The discriminative task becomes trivial, and the classifier can achieve perfect accuracy but fail to estimate the ratios correctly due to poorly calibrated output probabilities (Rhodes et al., 2020; Choi et al., 2021). Such a failure mode is illustrated on a simple DRE task for 2-dimensional Gaussians in Figure 2(a), where $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{4}, \boldsymbol{I})$. In particular, the classifier cannot capture the entire range of variation of the likelihood ratios (see Fig. 2(a)).

Improving Estimates with Bridges. To sidestep this challenge, Telescoping Density Ratio Estimation (TRE) (Rhodes et al., 2020) proposes a *divide-and-conquer* approach and partitions the binary classification problem in Eq. (1) into several easier subproblems. TRE constructs $T > 0$ bridge densities $\{p_t(\mathbf{x})\}_{t=1}^T$ by interpolating between $p_0(\mathbf{x}) \equiv q(\mathbf{x})$ and $p_1(\mathbf{x}) \equiv p(\mathbf{x})$, and trains T (conditional) classifiers to discriminate between samples from $p_t(\mathbf{x})$ and $p_{t+1}(\mathbf{x})$. Intuitively, each of the likelihood ratios $\log r_t(\mathbf{x}) = \log \frac{p_t(\mathbf{x})}{p_{t+1}(\mathbf{x})}$ are better behaved and thus easier to learn than those in the original problem, as in Figure 2(a). Then, a telescoping sum of the intermediate classifier outputs allows for the recovery of the desired likelihood ratio:

$$\begin{aligned} \log r(\mathbf{x}) = \log q(\mathbf{x}) - \log p(\mathbf{x}) &= \sum_{t=1}^T \log r_t(\mathbf{x}) \\ &= \sum_{t=1}^T \log p_t(\mathbf{x}) - \log p_{t+1}(\mathbf{x}) \end{aligned} \qquad (2)$$

Figures 2(b)-(c) demonstrate that TRE with 4 and 9 intermediate distributions respectively dramatically improves performance for our 2-D task, as the histograms of the learned ratios significantly overlap with those of ground truth. However, naively increasing T is undesirable for a number of reasons. Not only does the model size and complexity in TRE grow with T, but also the approach requires the evaluation of multiple classifiers at test time, which can be prohibitively expensive.

3 FROM DISCRETE BRIDGES TO CONTINUOUS PATHS

As an alternative to Eq. (2), we consider a continuous *path* of $T \to \infty$ bridge distributions connecting $p_0(\mathbf{x}) \equiv q(\mathbf{x})$ and $p_1(\mathbf{x}) \equiv p(\mathbf{x})$ in distribution space. Concretely, we denote this sequence of probability densities indexed by $t \in [0,1]$ as $\{p_t(\mathbf{x})\}_{t=0}^1$ and let $p(t) = \mathcal{U}[0,1]$ denote a uniform distribution over time steps. We note that there are several ways to construct the intermediate bridges $p_t(\mathbf{x})$. A benefit of DRE-∞ (which also holds for TRE) is that it only requires we be able to efficiently *sample* from $p_t(\mathbf{x})$ without necessarily knowing its analytical form. For example, we can define $\mathbf{x}(t) := \sqrt{1-\alpha(t)^2}\mathbf{x} + \alpha(t)\mathbf{y}$, where $\mathbf{x} \sim q$, $\mathbf{y} \sim p$, $\mathbf{x}(t) \sim p_t$, and $\alpha : [0,1] \to \mathbb{R}^+$ is a positive function that satisfies $\alpha(0) = 0$ and $\alpha(1) = 1$.

To build some intuition for DRE-∞, we observe the behavior of the log density ratio between two bridge distributions as the gap ($\Delta t = \frac{1}{T}$) between $p_t(\mathbf{x})$ and $p_{t+1}(\mathbf{x})$ becomes infinitesimal. Using finite differences, we can see that the intermediate densities $p_t(\mathbf{x})$ are changing at each timestep t by: $(\log p_{t+\Delta t}(\mathbf{x}) - \log p_t(\mathbf{x}))/\Delta t = \left(\log \frac{p_{t+\Delta t}(\mathbf{x})}{p_t(\mathbf{x})}\right)/\Delta t \approx \frac{\partial}{\partial t}\log p_t(\mathbf{x})$. As $T \to \infty$ (and therefore $\Delta t \to 0$), this demonstrates that the object of interest is now not the individual log-ratios $\log r_t(\mathbf{x})$ as in Eq. (2), but the instantaneous rate of change of the intermediate log densities $\frac{\partial}{\partial t}\log p_t(\mathbf{x})$, which we denote as the *time score*.

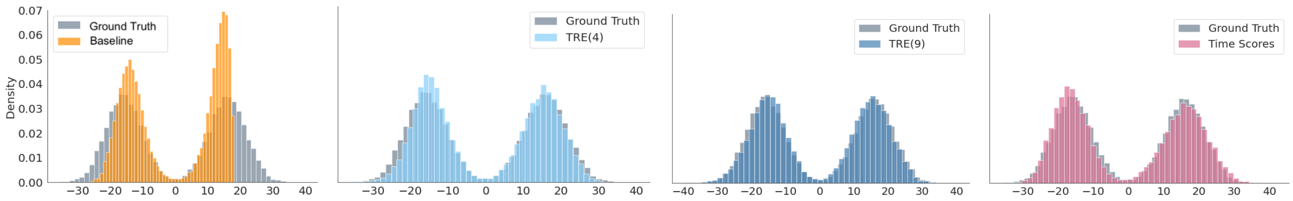
We formalize this intuition in the following proposition. The identity is well known in the path sampling literature and we include it here for completeness (Gelman and Meng, 1998; Owen, 2013; Yao et al., 2020).

Proposition 1. *Let* $\log r(\mathbf{x})$ *denote the log density ratio between the two densities* $p_0(\mathbf{x})$ *and* $p_1(\mathbf{x})$. *When* $T \to \infty$, *we have the following:*

$$\log r(\mathbf{x}) = \log \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})} = \sum_{t=1}^T \log\left(\frac{p_{(t-1)/T}(\mathbf{x})}{p_{t/T}(\mathbf{x})}\right) = \int_1^0 \frac{\partial}{\partial \lambda}\log p_\lambda(\mathbf{x})d\lambda. \qquad (3)$$

We provide a more detailed derivation in Appendix A.1.

There are two key takeaways from Proposition 1. First, as the number of bridge distributions increase to infinity, the telescoping sum in Eq. (2) becomes an integral. The integration is important because it can be computed using any off-the-shelf numerical integrator—the fact that it is one-dimensional also means that it will be very efficient to compute. This eliminates the need to evaluate all T intermediate classifiers at inference time as in Eq. (2). Instead, we have an additional degree of freedom where we can choose how accurately we want to estimate $\int_1^0 \frac{\partial}{\partial t}\log p_t(\mathbf{x})dt$ by specifying the error tolerance of the numerical integrator. This will adaptively determine the total number of necessary function (intermediate classifier) evaluations, making DRE-∞'s inference procedure more efficient than that of TRE. In fact, a surprising insight of DRE-∞ is that taking the number of bridges to the infinite limit



(a) Baseline MSE: 952.8 (b) TRE(4) MSE: 1.1 (c) TRE(9) MSE: 0.63 (d) Time MSE (Ours): 0.58

Figure 2: Motivating example on a synthetic 2-D Gaussian dataset, with learned density ratio estimates by method relative to the ground truth values for (a-d). The performance of TRE improves with more intermediate bridge distributions, while DRE-∞ outperforms the rest. The x-axis denotes the log-ratios.

actually confers significant computational benefits over the finite regime of TRE.

Next, we see that the log ratios $\log r_t(\mathbf{x})$ in Eq. (2) become the *time scores* $\frac{\partial}{\partial t}\log p_t(\mathbf{x})$ in Proposition 1. The time score captures—for each input $\mathbf{x}$—the instantaneous rate of change of the intermediate $p_t(\mathbf{x})$ along the prescribed path in distribution space. But as we rarely have access to the true time scores in most practical scenarios, we must *learn* them from data.

4 TIME SCORE MATCHING VIA INFINITESIMAL CLASSIFICATION

We propose to train a time score model $s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)$ to estimate the true time scores $\frac{\partial}{\partial t}\log p_t(\mathbf{x})$ via the following objective:

$$\mathcal{J}_{\text{time}}(\theta) = \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\lambda(t)\left(\frac{\partial}{\partial t}\log p_t(\mathbf{x}) - s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)\right)^2\right] \tag{4}$$

where $\lambda(t) : [0, 1] \to \mathbb{R}_+$ is a positive weighting function and $p(t)$ is the uniform distribution over timescales. While this objective is clearly minimized when $s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t) = \frac{\partial}{\partial t}\log p_t(\mathbf{x})$, it might not seem possible to evaluate and optimize it since $\frac{\partial}{\partial t}\log p_t(\mathbf{x})$ is unknown. However, using integration by parts as in (Hyvärinen, 2005), this objective can be simplified to the following expression in Proposition 2. As used in traditional score matching techniques (Kingma and LeCun, 2010; Song et al., 2020; Song and Ermon, 2019b), integration by parts allows us to obtain a practical objective function for training $s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)$ that does not depend on the true time scores $\frac{\partial}{\partial t}\log p_t(\mathbf{x})$.

Proposition 2 (Informal)**.** *Under certain regularity conditions, the optimal solution θ^* of Eq. (4) is the same as the optimal solution of:*

$$\mathcal{L}_{time}(\theta) = 2\mathbb{E}_{q(\mathbf{x})}[\lambda(0)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, 0)] - 2\mathbb{E}_{p(\mathbf{x})}[\lambda(1)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, 1)]$$
$$+ \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[2\lambda(t)\frac{\partial}{\partial t}s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t) + 2\lambda'(t)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t) + \lambda(t)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t)\right] \tag{5}$$

where the first two terms of Proposition 2 denote the boundary conditions for $t = \{0, 1\}$, and the expectations can be approximated via Monte Carlo. Our ability to estimate the objective via Monte Carlo samples makes training DRE-∞ much more efficient than TRE, which requires the explicit construction of T additional batches per gradient step for all intermediate classifiers. Even with a fixed batch size B that is divided among T intermediate classifiers, TRE requires that B grow with T (since it is necessary that $B/T \geqslant 1$ to train each classifier). We defer the exact assumptions and proof to Appendix A.4, and provide pseudocode in Appendix B.

The optimal time score model, denoted by $s_{\boldsymbol{\theta*}}^{\text{time}}(\mathbf{x}, t)$, satisfies $s_{\boldsymbol{\theta*}}^{\text{time}}(\mathbf{x}, t) \approx \frac{\partial}{\partial t}\log p_t(\mathbf{x})$. Therefore after training, the log-density-ratio can be estimated by:

$$\log r(\mathbf{x}) \approx \int_1^0 s_{\boldsymbol{\theta*}}^{\text{time}}(\mathbf{x}, t)\mathrm{d}t. \tag{6}$$

4.1 Joint Training Objective

We can also incorporate helpful auxiliary information from the data scores into the training objective in Eq. (4). Specifically, we can define a vector-valued score model $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x}, t)$ and train it with a hybrid objective that seeks to *jointly learn* $\nabla_{[\mathbf{x};t]}\log p(\mathbf{x}, t)$:

$$\mathcal{J}_{\text{joint}}(\theta) = \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\tfrac{1}{2}\lambda(t)\left\|\nabla_{[\mathbf{x};t]}\log p(\mathbf{x}, t) - \boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x}, t)\right\|_2^2\right]. \tag{7}$$

The data score component in Eq. (7) can be obtained by DSM (Vincent, 2011) or SSM (Song et al., 2020). For the SSM variant, (Hyvärinen, 2005; Song et al., 2020) shows that optimizing Eq. (7) is equivalent to optimizing the following objective.

Theorem 1 (Informal)**.** *Under certain regularity conditions, the solution to the optimization problem in Eq. (7) can be written as follows:*

$$\begin{aligned}&\theta^* = \arg\min_\theta \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\mathbb{E}_{p(\mathbf{v})}\Big[\\ &\frac{1}{2}\lambda(t)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[\mathbf{x}]\right\|_2^2 + \lambda(t)\mathbf{v}^\mathsf{T}\nabla_\mathbf{x}\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[\mathbf{x}]\mathbf{v}\\ &+\lambda(t)\frac{\partial}{\partial t}\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[t] + \lambda'(t)\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[t]\Big]\\ &+\mathbb{E}_{p_0(\mathbf{x})}[\lambda(0)\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},0)[t]] - \mathbb{E}_{p_1(\mathbf{x})}[\lambda(1)\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},1)[t]].\end{aligned} \tag{8}$$

where $\mathbf{v} \sim p(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[\mathbf{x}]$ denotes the data score component of $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}$, and $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[t]$ denotes the time-score component of $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}$. We defer the proof and detailed assumptions to Appendix A.5. In practice, the expectation in Eq. (8) can be approximated via Monte Carlo sampling. We can leverage DSM when the data follows a known stochastic differential equation (SDE) and $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ since the analytical form of $p_t(\mathbf{x})$ is tractable.

4.2 Link to Infinitesimal Classification

Recall that our motivation was to generalize TRE by taking the number of intermediate bridge distributions T to the infinite limit. With an infinite number of bridges, each intermediate classifier is tasked with distinguishing samples from two bridge distributions $p_{t+\Delta t}(\mathbf{x})$ and $p_t(\mathbf{x})$. In fact, the following proposition states that the optimal form of this infinitesimal classifier is given by the time score $\frac{\partial}{\partial t}\log p_t(\mathbf{x})$.

Proposition 3. *When $T \to \infty$, the Bayes-optimal classifier between two adjacent bridge distributions $p_{t/T}(\mathbf{x})$ and $p_{(t+1)/T}(\mathbf{x})$ for any $t \in [0,1]$ is:*

$$\boldsymbol{h}_{\boldsymbol{\theta}*}(\mathbf{x},t) = \frac{1}{2} + \frac{1}{4}\left(\frac{\partial}{\partial t}\log p_t(\mathbf{x})\right)\Delta t + o(\Delta t). \tag{9}$$

where $\Delta t = \frac{1}{T}$, and $\boldsymbol{h}_{\boldsymbol{\theta}*}(\mathbf{x},t) \in [0,1]$ is a conditional probabilistic classifier.

While the above result is instructive, it does not provide us with a practical algorithm for time score estimation—we cannot train an infinite number of such binary classifiers. To tackle this challenge, we consider the limit of the binary cross-entropy loss function for the optimal infinitesimal classifier (Proposition 3) when $T \to \infty$.

Proposition 4. *Let $\Delta t = 1/T$ and parameterize the binary classifier as $\boldsymbol{h}_{\boldsymbol{\theta}}(\mathbf{x},t) = \frac{1}{2} + \frac{1}{4}\boldsymbol{s}_{\boldsymbol{\theta}}^{time}(\mathbf{x},t)\Delta t$, where $\boldsymbol{s}_{\boldsymbol{\theta}}^{time}(\mathbf{x},t) \approx \frac{\partial}{\partial t}\log p_t(\mathbf{x})$ denotes a time score model. Then from the binary cross-entropy objective:*

$$\begin{aligned}&\arg\max_\theta \mathbb{E}_{p_t(\mathbf{x})}[\log(1-\boldsymbol{h}_{\boldsymbol{\theta}}(\mathbf{x},t))] + \mathbb{E}_{p_{t+\Delta t}(\mathbf{x})}[\log \boldsymbol{h}_{\boldsymbol{\theta}}(\mathbf{x},t)]\\ &= \arg\max_\theta -\frac{1}{4}(\Delta t)^2\mathbb{E}_{p_t(\mathbf{x})}\left[\left(\boldsymbol{s}_{\boldsymbol{\theta}}^{time}(\mathbf{x},t) - \frac{\partial}{\partial t}\log p_t(\mathbf{x})\right)^2\right] + o((\Delta t)^2)\end{aligned} \tag{10}$$

We defer the proof to Appendix A.3. Notably, the form of the objective function in Proposition 4 exactly mirrors that of Eq. (4), drawing the equivalence between solving an infinite number of "infinitesimal" classification problems and time score matching. This extends the previous connection between infinitesimal classification problems and score matching as mentioned in (Gutmann and Hirayama, 2012) and (Ceylan and Gutmann, 2018) in the context of estimating unnormalized probability models.

5 LEARNING TIME SCORES IN PRACTICE

5.1 Variance Reduction via Importance Weighting

In our preliminary experiments, we found that a naive implementation of Proposition 2 led to unstable training due to the high variance in the objective across the different timescales t. This finding is in accordance with recent work on diffusion probabilistic models (Nichol and Dhariwal, 2021; Kingma et al., 2021; Song et al., 2021a), which emphasize the critical role of applying the proper weighting function $\lambda(t)$ rather than randomly sampling $t \sim \mathcal{U}[0,1]$. This problem is also exacerbated by the fact that our training objective requires backpropagating through the score network $s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)$ with respect to t, which can cause training to diverge or progress extremely slowly for certain design choices.

Drawing inspiration from Nichol and Dhariwal (2021); Song et al. (2021a); Kingma et al. (2021), we learn an importance weighting scheme of the distribution over timescales $p(t)$. We optimize the following importance-weighted objective rather than Proposition 2:

$$\mathcal{J}_{\text{rw-time}}(\theta) = \mathbb{E}_{p_{\text{iw}}(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\frac{p(t)\lambda(t)}{p_{\text{iw}}(t)}\left(\frac{\partial}{\partial t}\log p_t(\mathbf{x}) - s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\right)^2\right] \tag{11}$$

where $p(t) \sim \mathcal{U}[0,1]$ and $p_{\text{iw}}(t)$ is a *learned* proposal distribution. We approximate the importance weighting distribution by maintaining a history buffer of the B most recent loss values in Proposition 2 (excluding the boundary conditions which are constant w.r.t. t). Then, we use this buffer to estimate an importance sampling distribution $p_{\text{iw}}(t)$ over t designed to reduce the variance of the loss. We report specific implementation details in Appendix D.1.

5.2 Incorporating Auxiliary Information via Data Scores

Another advantage of DRE-∞ is that it allows for considerable flexibility in the way that the likelihood ratios $r(\mathbf{x})$ are computed. Recall that in Eq. (6), the time

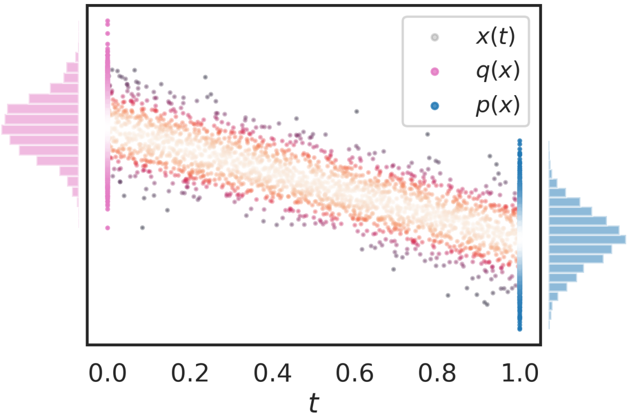


Figure 3: An example of the simple line path $\mathbf{y}(t) = \mathbf{x} + t \cdot (\mathbf{z} - \mathbf{x})$ bridging the high-density regions of $q(\mathbf{x})$ and $p(\mathbf{x})$, where $\mathbf{z}$ is sampled from $p(\mathbf{x})$. Brighter color indicates higher density.

scores $\frac{\partial}{\partial t} \log p_t(\mathbf{x})$ are integrated over $t \in [0, 1]$ while $\mathbf{x}$ is fixed along a horizontal path. However, we are not required to stick to this simple integral. We can actually construct *arbitrary* paths—that is, vary both $\mathbf{x}$ and t in the integral—with a theoretical guarantee that we will recover the same density ratios as before. We find that this approach often helps improve performance in practice, and call it the "pathwise method."

The pathwise method aims to compute a variant of Eq. (6) by evaluating our time score model at various points $\mathbf{y}(t)$ where its estimates will be the most accurate. To do so, we prescribe a path $\mathbf{y}(t)$ such that it connects $\mathbf{x}$ in the high data density region of $p_0(\mathbf{x}) \equiv q(\mathbf{x})$ to $\mathbf{z}$ in the high data density region of $p_1(\mathbf{x}) \equiv p(\mathbf{x})$. This trajectory can be described by an ordinary differential equation (ODE):

$$\begin{cases} \mathbf{y}'(t) = \boldsymbol{d}(\mathbf{y}, t) \\ \mathbf{y}(0) = \mathbf{x}. \end{cases}$$

where $\boldsymbol{d}$ is any function that captures the relationship between $\mathbf{y}$ and t. A simple example of such a path is the line $\mathbf{y}(t) = \mathbf{x} + t \cdot (\mathbf{z} - \mathbf{x})$ as shown in Figure 3, where $\mathbf{y}'(t) = (\mathbf{z} - \mathbf{x})$. A reasonable choice for $\mathbf{z}$ in this case are samples from $p(\mathbf{x})$, but we note that there are several possible choices for the path connecting $p(\mathbf{x})$ and $q(\mathbf{x})$ (Gelman and Meng, 1998).

Using this path, the difference between $\log q(\mathbf{x})$ and $\log p(\mathbf{y})$ can be obtained by integration:

$$\log q(\mathbf{x}) - \log p(\mathbf{y}) = \int_1^0 \frac{\partial}{\partial t} \log p_t(\mathbf{y}(t)) + \boldsymbol{d}(\mathbf{y}(t), t)^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{y}(t)) \mathrm{d}t.$$

Finally, we can compute the density ratio via the following expression:

$$\begin{aligned} r(\mathbf{x}) &= (\log q(\mathbf{x}) - \log p(\mathbf{y})) + (\log p(\mathbf{y}) - \log p(\mathbf{x})) \\ &= \underbrace{\left(\int_1^0 \frac{\partial}{\partial t} \log p_t(\mathbf{y}(t)) + \boldsymbol{d}(\mathbf{y}(t), t)^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{y}(t)) \mathrm{d}t \right)}_{\text{Term 1}} \\ &+ \underbrace{\left(\int_1^0 \nabla_{\mathbf{x}} \log p(\mathbf{y} + t(\mathbf{x} - \mathbf{y}))^\intercal (\mathbf{x} - \mathbf{y}) \mathrm{d}t \right)}_{\text{Term 2}}. \end{aligned} \tag{12}$$

Eq. (12) decomposes the density ratio into 2 terms, where the first term depends on both the time score and data score, while the second term only depends on the data score. The integrals in Eq. (12) can be approximated using off-the-shelf ODE solvers. Note that when the density of p is tractable (e.g. a Gaussian distribution, as in energy based modeling), the second term $\log p(\mathbf{y}) - \log p(\mathbf{x})$ can be computed in closed form. This property of the pathwise method makes it a particularly attractive alternative to Eq. (6). We note that this method can be trained with the joint objective function in Eq. (7).

6 EXPERIMENTAL RESULTS

In this section, we are interested in empirically investigating the following questions:

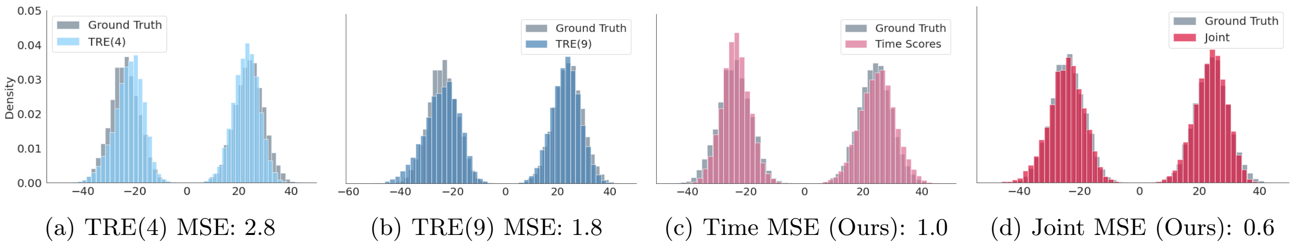
1. Does DRE-∞ lead to more accurate density ratio estimation than existing baselines?

2. Does incorporating auxiliary information for DRE (e.g. learning the data scores) help to learn more accurate ratios?

6.1 Synthetic Gaussian Experiments

Our running example with the 2-D synthetic dataset of Gaussian mixtures is comprised of 10K samples each from $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{4}, \boldsymbol{I})$. We use the Variance Preserving SDE (VPSDE) noise schedule as in (Ho et al., 2020; Song et al., 2021b) for the construction of $\mathbf{x}(t)$. Thus $\mathbf{x}(t) \sim q(\mathbf{x})$ when $t = 0$ and $\mathbf{x}(t) \sim p(\mathbf{x})$ when $t = 1$. Our score networks are fully-connected MLPs with ELU activation functions, with the time conditioning signal concatenated to the inputs before feeding them into the network. As shown in Figure 2, our DRE-∞ outperforms all baselines with a finite number (0, 4, 9) of intermediate distributions.

Additionally, the benefits of the pathwise approach (Section 5.2) is shown in Figure 4(d), where all models trained on $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{4}, \boldsymbol{I})$ are evaluated on 10K samples drawn from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and



(a) TRE(4) MSE: 2.8 (b) TRE(9) MSE: 1.8 (c) Time MSE (Ours): 1.0 (d) Joint MSE (Ours): 0.6

Figure 4: Additional results on the synthetic 2-D Gaussian dataset on a more challenging evaluation task, where half the samples are shifted by 1. While all models' performance slightly degrade, our joint score matching objective still accurately recovers the density ratio estimates. The x-axis denotes the log ratios.

$\mathcal{N}(\mathbf{5}, \boldsymbol{I})$ each. Although all models perform worse than in Figure 2 due to the slight mismatch in train and test conditions (the baseline binary classifier is not shown, as it performed extremely poorly), the additional integration path helps the jointly trained score model to more accurately recover the density ratios relative to other methods. We also note that the pathwise approach yielded the lowest MSE of 0.35 among all methods shown in Figure 2. We refer the reader to Appendix D for more details on the model architecture and hyperparameter settings, as well as Appendix F.1 for additional synthetic experiments on 1-D problems.

6.2 Mutual Information Estimation for High-Dimensional Gaussians

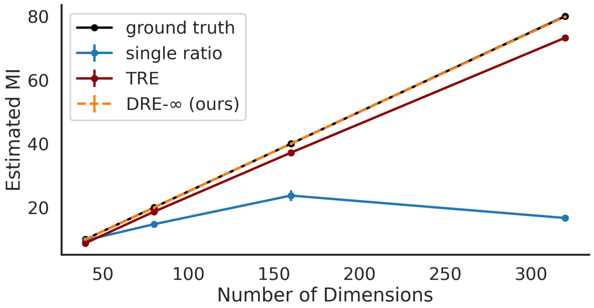


Figure 5: Estimated MI between two correlated high-dimensional Gaussian random variables, where our joint score matching objective outperforms TRE in all settings. Results are averaged over 3 runs.

Next, we evaluate our approach on a mutual information (MI) estimation task between two correlated, high-dimensional Gaussians. MI estimation between two random variables is a direct application of DRE, as the problem can be reduced to estimating average density ratios between their joint density and the product of their marginals: $I(\mathbf{x};\mathbf{y}) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{x})p(\mathbf{y})}\right]$. We adapt the experimental setting of (Belghazi et al., 2018; Poole et al., 2019; Rhodes et al., 2020), where we sweep over the dimensions $d = \{40, 80, 160, 320\}$, and fix the correlation coefficient to be $\rho = 0.8$.

For the TRE baseline, we use the default hyperparameter settings in (Rhodes et al., 2020). We use the joint score matching objective for our method, where we use the same interpolation procedure as in the 2-D Gaussian experiment in Section 6.1. We find that our method's estimated MI values overlap with the ground truth in all settings as shown in Figure 5. DRE-∞ outperforms TRE in all cases and the performance gap between the two methods increase in higher dimensions. A single binary classifier, on the other hand, fails completely for dimensions greater than $d = 40$. For additional details on the experimental setup, we refer the reader to Appendix F.2.

6.3 Energy-based Modeling with MNIST

In this experiment, we train an energy-based model (EBM) of the MNIST dataset (LeCun, 1998) using timewise score matching. Specifically, we let $q(\mathbf{x})$ denote the distribution over MNIST digits and experiment with three different settings for $p(\mathbf{x})$ as in (Rhodes et al., 2020): a Gaussian noise model, a Gaussian copula, and a Rational Quadratic Neural Spline Flow (RQ-NSF) (Durkan et al., 2019). After obtaining our likelihood ratio estimates, we can estimate the likelihood of our data by computing $\log p_{\text{data}}(\mathbf{x}) \approx \log q(\mathbf{x}) = \log r(\mathbf{x}) + \log p(\mathbf{x})$. We construct our bridge distributions in the latent space of our normalizing flow via the VPSDE interpolation schedule.

We report the likelihoods we obtain via DRE-∞ in bits per dimension (bpd). Additionally, we compare our bpds with both a lower bound estimated via Annealed Importance Sampling (AIS) (Neal, 2001) and a conservative upper bound estimated via the Reverse Annealed Importance Sampling Estimator (RAISE) (Burda et al., 2015). Such comparisons with AIS/RAISE are important because $\log q(\mathbf{x})$ is only an estimate of $\log p_{\text{data}}$ obtained via DRE-∞'s approximate normalizing constant. If DRE-∞ fails to estimate the likelihood ratio

Method	**Interpolation**	Noise	Direct (↓)	RAISE (↓)	AIS (↓)
NCE	Gaussian	2.01	1.96	1.99	2.01
TRE	Gaussian	2.01	1.39	1.35	1.35
DRE-∞	Gaussian	2.01	**1.33**	**1.33**	**1.33**
NCE	Copula	1.40	1.33	1.48	1.45
TRE	Copula	1.40	1.24	1.23	1.22
DRE-∞	Copula	1.40	**1.21**	**1.21**	**1.21**
NCE	RQ-NSF	1.12	1.09	1.10	1.10
TRE	RQ-NSF	1.12	1.09	1.09	1.09
DRE-∞	RQ-NSF	1.12	1.09	**1.08**	**1.08**

Table 1: Estimated log-likelihood results on the energy-based modeling task for MNIST, reported in bits per dimension (bpd). Lower is better. Results for NCE and TRE are from (Rhodes et al., 2020). We note that DRE-∞'s time score matching framework leads to performance improvements over relevant baselines in all settings.

$\log r(\mathbf{x})$ accurately, then $\log q(\mathbf{x})$ may be a poor approximation to $\log p_{\text{data}}$. AIS and RAISE allow us to obtain a more accurate estimate of the intractable normalizing constant by constructing (another) sequence of intermediate distributions between our estimated target distribution $q(\mathbf{x})$ and another proposal distribution $p_1(\mathbf{x})$, which we set to be the flow $p(\mathbf{x})$.

As shown in Table 1, we note that using an infinite number of bridge distributions improves performance on the bpds. More importantly, our bpd estimates directly obtained by the output of the score network are very close to those of AIS and RAISE, indicating that our density ratio estimates are accurate even for high dimensional datasets such as MNIST. This is not necessarily the case for other methods such as TRE. We refer the reader to Appendix F.3 for additional details on the experimental setup and likelihood evaluations.

7 RELATED WORK

Score-Based Generative Modeling. Our work builds on the growing body of work on score matching (Hyvärinen, 2005; Vincent, 2011) and score-based generative models (Song et al., 2020; Song and Ermon, 2019b, 2020; Song et al., 2021b). Given empirical samples, the goal of score-based generative modeling is to accurately model the data density $p(\mathbf{x})$ by learning its (Stein) *score* $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ (Hyvärinen, 2005; Liu et al., 2016). We notably build upon (Song and Ermon, 2019b) to estimate the time scores of the data in addition to the data scores, which allows for accurate DRE. We additionally establish an interesting connection between time score matching and solving an infinite number of infinitesimal classification problems, which extends the work of (Gutmann and Hirayama, 2012) and (Ceylan and Gutmann, 2018) for (Stein) score matching.

DRE and Importance Sampling. DRE has its roots in *importance sampling*, which has numerous applications in Bayesian statistics and the approximation of intractable normalizing constants (Meng and Wong, 1996; Gelman and Meng, 1998; Neal, 2001; Fishman, 2013). In particular, bridge sampling (Bennett, 1976; Meng and Wong, 1996) was introduced as a variance reduction technique in MCMC to “shorten the path" between two densities. Modern versions of bridge sampling include (Rhodes et al., 2020; Sinha et al., 2020), which also incorporate an element of warping (Hoffman et al., 2019) via a normalizing flow to further improve performance. *Path sampling* bears the closest resemblance to our method (Gelman and Meng, 1998), in which the discrete bridges of (Geyer, 1994; Meng and Wong, 1996) are relaxed to an infinite number as indexed by a continuous value $t \in [0, 1]$. However, path sampling estimators are typically not evaluated on a fixed point $\mathbf{x}$ as in our use case for DRE. Another way in which our work differs from traditional parametric importance sampling methods such as AIS is that we do not require explicit parametric forms of the intermediate distributions — we only require the ability to sample from them. Our work most closely mirrors (Rhodes et al., 2020), where we take the number of intermediate distributions to the limit. This approach eliminates the need to train multiple classifiers, and makes it easier to incorporate auxiliary information via the data scores to improve DRE in practice.

8 CONCLUSION

We introduced DRE-∞, a novel time score matching framework for DRE. We proposed to smoothly interpolate between two densities by specifying an infinite number of bridge distributions, and trained a neural network to estimate the instantaneous rate of change of the log densities (“time scores”) along this path. After training, we demonstrated that we can leverage black-box numerical integration techniques to efficiently obtain likelihood ratios. We provide a reference implementation in PyTorch (Paszke et al., 2019), and the codebase for this work is open-sourced at https://github.com/ermongroup/time-score-dre.

However, this work is not without limitations. Although the method depends on the specification of an interpolation scheme, it is not clear whether there is an optimal way to bridge the two densities together. Additionally, DRE-∞ takes longer to converge as $q(\mathbf{x})$ becomes further apart from $p(\mathbf{x})$, though this speaks to the challenging nature of the DRE problem as a whole. It would be interesting to investigate whether there is a time-dependent function $\lambda(t)$ such that the time score matching loss corresponds to the maximum likelihood training of a binary classifier (Song et al., 2021a; Kingma et al., 2021). Additionally, exploring optimal integration paths between $p(\mathbf{x})$ and $q(\mathbf{x})$ would be exciting future work.

Author Contributions

Kristy Choi wrote the code, ran the experiments, and wrote the paper. Chenlin Meng helped run the experiments and write the paper. Yang Song designed the project, proposed the theoretical results, and wrote the proofs. Stefano Ermon supervised the project, provided valuable feedback, and helped edit the paper.

Acknowledgements

We are thankful to Rui Shu and Chris Cundy for providing helpful feedback on the paper. KC is supported by the NSF GRFP, Stanford Graduate Fellowship, and Two Sigma Diversity PhD Fellowship. YS is supported by the Apple PhD Fellowship in AI/ML. This research was supported by NSF (#1651565, #1522054, #1733686), ONR (N00014-19-1-2145), AFOSR (FA9550-19-1-0024), ARO (W911NF2110125), and Amazon AWS.

References

Abadie, A. and Imbens, G. W. (2016). Matching on the estimated propensity score. *Econometrica*, 84(2):781–807.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. (2018). Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR.

Bennett, C. H. (1976). Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, 22(2):245–268.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Artificial Intelligence and Statistics*, pages 102–110. PMLR.

Ceylan, C. and Gutmann, M. U. (2018). Conditional noise-contrastive estimation of unnormalised models. In *International Conference on Machine Learning*, pages 726–734. PMLR.

Chatterjee, S. and Diaconis, P. (2018). The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135.

Choi, K., Liao, M., and Ermon, S. (2021). Featurized density ratio estimation. *arXiv preprint arXiv:2107.02212*.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. *Advances in Neural Information Processing Systems*, 32:7511–7522.

Fishman, G. (2013). *Monte Carlo: concepts, algorithms, and applications*. Springer Science & Business Media.

Gelman, A. and Meng, X.-L. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185.

Geyer, C. J. (1994). Estimating normalizing constants and reweighting mixtures.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

Gutmann, M. and Hirayama, J.-i. (2012). Bregman divergence as general framework to estimate unnormalized statistical models. *arXiv preprint arXiv:1202.3727*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

Johansson, F. D., Kallus, N., Shalit, U., and Sontag, D. (2018). Learning weighted representations for generalization across designs. *arXiv preprint arXiv:1802.08598*.

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

Kingma, D. P. and LeCun, Y. (2010). Regularized estimation of image statistics by score matching. In *NIPS*, volume 509, page 618.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *arXiv preprint arXiv:2107.00630*.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

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

McAllester, D. and Stratos, K. (2020). Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR.

Meng, X.-L. and Wong, W. H. (1996). Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, pages 831–860.

Menon, A. and Ong, C. S. (2016). Linking losses for density ratio and class-probability estimation. In *International Conference on Machine Learning*, pages 304–313. PMLR.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods.* Department of Computer Science, University of Toronto Toronto, ON, Canada.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing*, 11(2):125–139.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2007). Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *NIPS*, pages 1089–1096.

Nichol, A. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. *arXiv preprint arXiv:1606.00709*.

Øksendal, B. (2003). Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Owen, A. B. (2013). Monte carlo theory, methods and examples.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., and Tucker, G. (2019). On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Rhodes, B., Xu, K., and Gutmann, M. U. (2020). Telescoping density-ratio estimation. *Advances in Neural Information Processing Systems*, 33.

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

Salimans, T. and Ho, J. (2021). Should ebms model the energy or the score? In *Energy Based Models Workshop-ICLR 2021*.

Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pages 3076–3085. PMLR.

Sinha, A., O'Kelly, M., Tedrake, R., and Duchi, J. C. (2020). Neural bridge sampling for evaluating safety-critical autonomous systems. *Advances in Neural Information Processing Systems*, 33.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Song, J. and Ermon, S. (2019a). Understanding the limitations of variational mutual information estimators. *arXiv preprint arXiv:1910.06222*.

Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021a). Maximum likelihood training of score-based diffusion models. *arXiv e-prints*, pages arXiv–2101.

Song, Y. and Ermon, S. (2019b). Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*.

Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2020). Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., and Kawanabe, M. (2008). Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. (2013). Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370.

Yao, Y., Cademartori, C., Vehtari, A., and Gelman, A. (2020). Adaptive path sampling in metastable posterior distributions. *arXiv preprint arXiv:2009.00471*.

Supplementary Material:
Density Ratio Estimation via Infinitesimal Classification

A Detailed Derivations of Theoretical Results

In this section, we provide a more careful treatment of the relevant derivations in the main text.

A.1 Bridge Sampling to Path Sampling

The identity for converting bridge sampling to path sampling in Proposition 1 is well known (Gelman and Meng, 1998; Owen, 2013; Yao et al., 2020), and we include it here for completeness.

Proposition 1. *Let $\log r(\mathbf{x})$ denote the log density ratio between the two densities $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$. When $T \to \infty$, we have the following:*

$$\log r(\mathbf{x}) = \log \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})} = \sum_{t=1}^{T} \log \left(\frac{p_{(t-1)/T}(\mathbf{x})}{p_{t/T}(\mathbf{x})} \right) = \int_1^0 \frac{\partial}{\partial \lambda} \log p_\lambda(\mathbf{x}) d\lambda \tag{13}$$

Proof.

$$\begin{aligned}
\log \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})} &= \log p_0(\mathbf{x}) - \log p_1(\mathbf{x}) \\
&= \big(\log p_0(\mathbf{x}) - \log p_{1/T}(\mathbf{x})\big) + \big(\log p_{1/T}(\mathbf{x}) - \log p_{2/T}(\mathbf{x})\big) + \cdots + \big(\log p_{(T-1)/T}(\mathbf{x}) - \log p_1(\mathbf{x})\big) \\
&= \sum_{t=1}^{T} \log \left(\frac{p_{(t-1)/T}(\mathbf{x})}{p_{t/T}(\mathbf{x})} \right) \\
&= \sum_{t=1}^{T} \log \left(1 + \frac{p_{(t-1)/T}(\mathbf{x}) - p_{t/T}(\mathbf{x})}{p_{t/T}(\mathbf{x})} \right) \\
&\approx \sum_{t=1}^{T} \left(\frac{p_{(t-1)/T}(\mathbf{x}) - p_{t/T}(\mathbf{x})}{p_{t/T}(\mathbf{x})} \right) \\
&= \lim_{T \to \infty} \sum_{t=1}^{T} \frac{d}{d\lambda} \log p_\lambda(\mathbf{x})|_{\lambda = t/T} \\
&= \int_1^0 \log p_\lambda(\mathbf{x}) d\lambda
\end{aligned}$$

□

A.2 Form of Optimal Infinitesimal Classifier

For completeness, we restate Proposition 3 prior to providing the proof.

Proposition 3. *When $T \to \infty$, the form of the Bayes-optimal classifier between two adjacent bridge distributions $p_{t/T}(\mathbf{x})$ and $p_{(t+1)/T}(\mathbf{x})$ for any $t \in [0, 1]$ becomes:*

$$\boldsymbol{h}_{\boldsymbol{\theta}*}(\mathbf{x}, t) = \frac{1}{2} + \frac{1}{4} \left(\frac{\partial}{\partial t} \log p_t(\mathbf{x}) \right) \Delta t + o(\Delta t). \tag{14}$$

Proof. Recall that it is trained by optimizing the following cross-entropy loss:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p_{(t-1)/T}(\mathbf{x})}[\log(1 - \boldsymbol{h}_{\boldsymbol{\theta}}(\mathbf{x}, t/T))] + \mathbb{E}_{p_{t/T}(\mathbf{x})}[\log \boldsymbol{h}_{\boldsymbol{\theta}}(\mathbf{x}, t/T)],$$

where $\boldsymbol{h_\theta}(\mathbf{x}, t/T) \in [0, 1]$ is a binary classifier. By calculus of variations, we can derive that for the optimal model parameter $\boldsymbol{\theta}^*$,

$$\boldsymbol{h}_{\boldsymbol{\theta}^*}(\mathbf{x}, t/T) = \sigma(\log p_{t/T}(\mathbf{x}) - \log p_{(t-1)/T}(\mathbf{x})),$$

where $\sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$ is the sigmoid function. Thus when $T \to \infty$, we clearly have that:

$$\boldsymbol{h}_{\boldsymbol{\theta}^*}(\mathbf{x}, t) = \frac{1}{2} + \frac{1}{4}\left(\frac{\partial}{\partial t}\log p_t(\mathbf{x})\right)\Delta t + o(\Delta t).$$

where $\Delta t = \frac{1}{T}$. □

A.3 Derivation of the Time Score from Infinitesimal Binary Classification

For completeness, we restate Proposition 4 prior to providing the proof.

Proposition 4. *Let $\Delta t = 1/T$ and parameterize the binary classifier as $\boldsymbol{h_\theta}(\mathbf{x}, t) = \frac{1}{2} + \frac{1}{4}\boldsymbol{s}_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t)\Delta t$. Then from the binary cross-entropy objective, we can derive:*

$$\arg\max_{\theta} \mathbb{E}_{p_t(\mathbf{x})}[\log(1 - \boldsymbol{h_\theta}(\mathbf{x}, t))] + \mathbb{E}_{p_{t+\Delta t}(\mathbf{x})}[\log \boldsymbol{h_\theta}(\mathbf{x}, t)] = \arg\max_{\theta} -\frac{1}{4}(\Delta t)^2 \mathbb{E}_{p_t(\mathbf{x})}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t) - \frac{\partial}{\partial t}\log p_t(\mathbf{x})\right\|_2^2\right] + o((\Delta t)^2) \tag{15}$$

Proof. From the definition of the binary cross entropy loss, we have:

$$\begin{aligned}
\arg\max_{\theta} \mathbb{E}_{p_t(\mathbf{x})}[\log(1 - \boldsymbol{h_\theta}(\mathbf{x}, t))] + \mathbb{E}_{p_{t+\Delta t}(\mathbf{x})}[\log \boldsymbol{h_\theta}(\mathbf{x}, t)] &= \arg\max_{\theta} -2\log 2 + \frac{1}{2}\Delta t \int (p_{t+\Delta t}(\mathbf{x}) - p_t(\mathbf{x}))\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)\mathrm{d}\mathbf{x} \\
&\quad - \frac{1}{4}(\Delta t)^2 \mathbb{E}_{p_t(\mathbf{x})}[\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)^2] + o((\Delta t)^2) \\
&= \arg\max_{\theta} -2\log 2 + \frac{1}{2}(\Delta t)^2 \mathbb{E}_{p_t(\mathbf{x})}\left[\frac{\partial}{\partial t}\log p_t(\mathbf{x})\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)\right] \\
&\quad - \frac{1}{4}(\Delta t)^2 \mathbb{E}_{p_t(\mathbf{x})}[\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)^2] + o((\Delta t)^2) \\
&= \arg\max_{\theta} -\frac{1}{4}(\Delta t)^2 \mathbb{E}_{p_t(\mathbf{x})}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t) - \frac{\partial}{\partial t}\log p_t(\mathbf{x})\right\|_2^2\right] \\
&\quad + o((\Delta t)^2)
\end{aligned}$$

□

A.4 Time score matching objective

We provide a more detailed derivation of the time-wise score matching objective in Eq. (4) below.

Proposition 2. *Under certain regularity conditions, the optimal solution θ^* of Eq. (4) is the same as the optimal solution of:*

$$\mathcal{L}_{time}(\theta) = 2\mathbb{E}_{q(\mathbf{x})}[\lambda(0)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, 0)] - 2\mathbb{E}_{p(\mathbf{x})}[\lambda(1)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, 1)] + \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[2\lambda(t)\frac{\partial}{\partial t}s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t) + 2\lambda'(t)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t) + \lambda(t)s_{\boldsymbol{\theta}}^{time}(\mathbf{x}, t)^2\right] \tag{16}$$

Proof. To see this, we expand out the square and use the Leibniz integral rule.

$$
\begin{aligned}
&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\lambda(t)\left(\frac{\partial}{\partial t}\log p_t(\mathbf{x}) - s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\right)^2\right] \\
=&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\lambda(t)\left(\frac{\partial}{\partial t}\log p_t(\mathbf{x})\right)^2 - 2\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\frac{\partial}{\partial t}\log p_t(\mathbf{x}) + \lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2\right] \\
=&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[-2\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\frac{\partial}{\partial t}\log p_t(\mathbf{x}) + \lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2\right] + \text{const.} \\
=&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[-2\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\frac{\partial}{\partial t}\log p_t(\mathbf{x})\right] + \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2] + \text{const.} \\
=&\int_0^1\int -2\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\frac{\partial}{\partial t}\log p_t(\mathbf{x})p_t(\mathbf{x})\mathrm{d}\mathbf{x}\mathrm{d}t + \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2] + \text{const.} \\
=&-2\int_0^1\int \lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\frac{\partial p_t(\mathbf{x})}{\partial t}\mathrm{d}\mathbf{x}\mathrm{d}t + \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2] + \text{const.} \\
=&2\int[\lambda(0)p_0(\mathbf{x})s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},0) - \lambda(t)2p_1(\mathbf{x})s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},1)]\mathrm{d}\mathbf{x} \\
&+2\int_0^1\int p_t(\mathbf{x})\left[\lambda(t)\frac{\partial}{\partial t}s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t) + \lambda'(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)\right]\mathrm{d}\mathbf{x}\mathrm{d}t + \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2] + \text{const.} \\
=&2\mathbb{E}_{q(\mathbf{x})}[\lambda(0)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},0)] - 2\mathbb{E}_{p(\mathbf{x})}[\lambda(1)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},1)] \\
&+\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[2\lambda(t)\frac{\partial}{\partial t}s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t) + 2\lambda'(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t) + \lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)^2\right] + \text{const.}
\end{aligned}
$$

□

The optimal time score model, denoted by $s_{\boldsymbol{\theta}*}^{\text{time}}(\mathbf{x},t)$, satisfies $s_{\boldsymbol{\theta}*}^{\text{time}}(\mathbf{x},t) \approx \frac{\partial}{\partial t}\log p_t(\mathbf{x})$. Therefore, the log-density-ratio can be estimated by

$$\log r(\mathbf{x}) \approx \int_1^0 s_{\boldsymbol{\theta}*}^{\text{time}}(\mathbf{x},t)\mathrm{d}t.$$

A.5 Joint Score Matching Objective

The assumptions needed for this proof are largely adapted from (Song et al., 2020).

Theorem 1. *Assume that the vector-valued score function learned by the joint score network* $\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)$ *and the true data scores* $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ *are differentiable, and satisfy* $\mathbb{E}[\|\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)\|_2^2] < \infty$ *and* $\mathbb{E}[\|\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\|_2^2] < \infty$. *Additionally, we assume: (1) identifiability—the model family* $\{p_m(\mathbf{x};\theta)|\theta \in \Theta\}$ *is well-specified; and (2) that the score model satisfies some boundary conditions, e.g.* $\forall\theta \in \Theta, \lim_{\|\mathbf{x}\|\to\infty} s_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)p_{\text{data}} = 0$. *We also assume that the projection vectors* $\mathbf{v} \sim p(\mathbf{v}) = \mathcal{N}(\mathbf{0},\boldsymbol{I})$. *Then, the solution to the optimization problem in Eq. (7) can be written as follows:*

$$
\begin{aligned}
\theta^* =& \arg\min_\theta \mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\mathbb{E}_{p(\mathbf{v})}\left[\frac{1}{2}\lambda(t)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[\mathbf{x}]\right\|_2^2 + \lambda(t)\mathbf{v}^\mathsf{T}\nabla_{\mathbf{x}}\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[\mathbf{x}]\mathbf{v}\right. \\
&\left. + \lambda(t)\frac{\partial}{\partial t}\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[t] + \lambda'(t)\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},t)[t]\right] \\
&+ \mathbb{E}_{p_0(\mathbf{x})}[\lambda(0)\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},0)[t]] - \mathbb{E}_{p_1(\mathbf{x})}[\lambda(1)\boldsymbol{s}_{\boldsymbol{\theta}}^{joint}(\mathbf{x},1)[t]].
\end{aligned}
\tag{17}
$$

Proof. The proof involves expanding out the square and using integration by parts as in (Hyvärinen, 2005).

$$\begin{aligned}\mathcal{L}_{\text{joint}}(\theta) =&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\frac{1}{2}\lambda(t)\left\|\nabla_{[\mathbf{x};t]}\log p(\mathbf{x},t) - \boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)\right\|_2^2\right]\\ =&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\left[\frac{1}{2}\lambda(t)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[\mathbf{x}]\right\|_2^2 + \lambda(t)\operatorname{tr}(\boldsymbol{J}_{\boldsymbol{s}_{\boldsymbol{\theta}}}(\mathbf{x},t)) + \lambda'(t)\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[t]\right]\\ &+\mathbb{E}_{p_0(\mathbf{x})}[\lambda(0)\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},0)[t]] - \mathbb{E}_{p_1(\mathbf{x})}[\lambda(1)\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},1)[t]] + \text{const.}\\ =&\mathbb{E}_{p(t)}\mathbb{E}_{p_t(\mathbf{x})}\mathbb{E}_{p(\mathbf{v})}\left[\frac{1}{2}\lambda(t)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[\mathbf{x}]\right\|_2^2 + \lambda(t)\mathbf{v}^{\mathsf{T}}\nabla_{\mathbf{x}}\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[\mathbf{x}]\mathbf{v} + \lambda(t)\frac{\partial}{\partial t}\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[t] + \lambda'(t)\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},t)[t]\right]\\ &+\mathbb{E}_{p_0(\mathbf{x})}[\lambda(0)\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},0)[t]] - \mathbb{E}_{p_1(\mathbf{x})}[\lambda(1)\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{joint}}(\mathbf{x},1)[t]] + \text{const.}\end{aligned} \tag{18}$$

□

We note that for the joint training objective, the optimal score model satisfies $\boldsymbol{s}_{\boldsymbol{\theta}*}^{\text{joint}}(\mathbf{x},t) \approx [\nabla_{\mathbf{x}}\log p_t(\mathbf{x}); \frac{\partial}{\partial t}\log p_t(\mathbf{x})]$. This is because $\boldsymbol{s}_{\boldsymbol{\theta}*}^{\text{joint}}(\mathbf{x},t) \approx \nabla_{[\mathbf{x};t]}\log p(\mathbf{x},t)$ and

$$\begin{aligned}\nabla_{\mathbf{x}}\log p(\mathbf{x},t) &= \nabla_{\mathbf{x}}\log p_t(\mathbf{x}) + \nabla_{\mathbf{x}}\log p(t) = \nabla_{\mathbf{x}}\log p_t(\mathbf{x})\\ \frac{\partial}{\partial t}\log p(\mathbf{x},t) &= \frac{\partial}{\partial t}\log p_t(\mathbf{x}) + \frac{\partial}{\partial t}\log p(t) = \frac{\partial}{\partial t}\log p_t(\mathbf{x}),\end{aligned}$$

since $p(t)$ does not depend on $\mathbf{x}$ and is a uniform distribution.

B Pseudocode for Training and Inference

We provide pseudocode for training the time score model using Eq. (4).

Algorithm 1 Time Score Matching

Input: Datasets $\{\mathcal{D}_P, \mathcal{D}_Q\}$, time score model $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},t)$, interpolation procedure `interpolate`, weighting function $\lambda : [0,1] \rightarrow \mathbb{R}_+$

```
Sample $t \sim \mathcal{U}[0,1]$
Sample $\mathbf{x} \sim \mathcal{D}_Q$, $\mathbf{y} \sim \mathcal{D}_P$
Interpolate $\mathbf{x}_t \leftarrow \texttt{interpolate}(\mathbf{x}, \mathbf{y}, t)$
$\hat{\mathcal{L}}(\theta) \leftarrow \lambda(t)\frac{\partial}{\partial t}s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}_t,t) + 2\lambda'(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}_t,t) + \lambda(t)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}_t,t) + 2\lambda(0)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x},0) - 2\lambda(1)s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{y},1)$
return $\hat{\mathcal{L}}(\theta)$
```

Next, we provide pseudocode for computing the density ratios via any black-box numerical integration method.

Algorithm 2 Density Ratio Estimation

Input: Time score model $\boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}$, minibatch of samples $\mathbf{x}$, start time $t_0 = 1$, end time $t_1 = 0$, initial condition $\mathbf{y}_0 = \mathbf{0}$

```
$s_{\boldsymbol{\theta}}^{\mathbf{x}} \leftarrow s_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, \cdot)$
$\log r(\mathbf{x}) \leftarrow \texttt{integrate}(s_{\boldsymbol{\theta}}^{\mathbf{x}}, (t_0, t_1), \mathbf{y}_0)$
return $\log r(\mathbf{x})$
```

C Structured Interpolations via Stochastic Differential Equations (SDEs)

We briefly mention a special case of DRE-∞'s interpolation mechanism where the analytical form of $p_t(\mathbf{x})$ is tractable. One such example is a diffusion process, where the data generating process of $q(\mathbf{x})$ is represented as a Markov chain that transforms a simple distribution $p_T(\mathbf{x}) \equiv p(\mathbf{x})$ into a target distribution $p_0(\mathbf{x}) \equiv q(\mathbf{x})$

(Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b). This sequential procedure can be described as the solution to an Itô stochastic differential equation (SDE):

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

where $\mathbf{w}$ represents Brownian motion, $\mathbf{f}(\cdot, t) : \mathbb{R}^D \to \mathbb{R}^D$ is the drift coefficient of $\mathbf{x}(t)$, and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient of $\mathbf{x}(t)$, where $\mathbf{x}(t) \sim p_t(\mathbf{x})$.

For learning settings where our data follows a known SDE, we can leverage the Fokker-Planck equation (Jordan et al., 1998; Øksendal, 2003) to transform the data scores $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ into time scores *without training an additional model*. The Fokker-Planck equation describes the time-evolution of the probability density $p_t(\mathbf{x})$ associated with the SDE:

$$\begin{aligned} \frac{\partial}{\partial t} \log p_t(\mathbf{x}) = & -\nabla \cdot \mathbf{f} - \mathbf{f}^{\mathsf{T}} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\ & + g^2(t) \left[||\nabla_{\mathbf{x}} \log p_t(\mathbf{x})||_2^2 + \mathrm{tr}(\nabla_{\mathbf{x}}^2 \log p_t(\mathbf{x})) \right] \end{aligned} \tag{19}$$

where $\mathbf{f}$ and g are well-defined for tractable SDEs. We can train a data score network $s_{\boldsymbol{\theta}}^{\text{data}}(\mathbf{x}, t)$ to approximate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ via sliced score matching (SSM) (Song et al., 2020) or denoising score matching (DSM) (Vincent, 2011; Song and Ermon, 2019b).

D Architecture and Implementation Details

We provide further details on the importance weighting scheme and the time score architecture discussed in Section 5.1.

D.1 Variance Reduction via Polynomial Interpolation with a Loss History Buffer

In our empirical evaluations, we experimented with various approaches for learning the proper weighting function $p_{\text{iw}}(t) \approx \lambda(t) : [0, 1] \to \mathbb{R}_+$ to reduce the variance in our training objectives. We found that estimating the weights by maintaining a history of the $B > 0$ most recent loss values (Nichol and Dhariwal, 2021) led to the most significant performance improvements. We used this loss history approach for the energy-based modeling experiments in MNIST, it was not necessary for our synthetic experiments.

We experimented with buffer sizes of $B = \{10, 100\}$, and batch sizes of $\{64, 128, 256, 500\}$. For each minibatch $t \sim \mathcal{U}[0, 1]$ sampled during training, we sorted the timescales in ascending order before computing the loss and storing the corresponding values into the buffer. This made the batch size very important, as it served as a discrete approximation to the way in the history was used to compute the weights in (Nichol and Dhariwal, 2021). After obtaining an initial estimate of the weights as in (Nichol and Dhariwal, 2021), we fit a ridge regression model using the stored time and weight values with the default settings for `PolynomialFeatures` in `scikit-learn` (polynomial degree 4 and a regularization coefficient of 0.001). This regression model was then used as an interpolation mechanism for returning the corresponding weight values for new values of t seen during training.

To avoid settings where the loss weighting would return negative values for $\alpha(t)$, we returned the absolute values of the interpolated weights before applying them in our loss function. For our MNIST experiments, we found that a buffer size of $B = 100$ and a batch size of 500 worked the best for the Gaussian noise and Gaussian copula noise models. For the RQ-NSF noise model, this interpolation mechanism did not improve performance (and thus we used the original VPSDE weighting scheme instead).

D.2 Time Score Network Architecture

When designing the time score network architecture for more complex datasets, we found that both sinusoidal positional embeddings (Ho et al., 2020) and Fourier embeddings (Tancik et al., 2020; Song et al., 2021b; Kingma et al., 2021) commonly used in the literature led to training instabilities when computing $\frac{\partial}{\partial t} \boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t)$. We hypothesize that this is due to the periodic nature of the sine and cosine functions, causing gradient information with respect to t to oscillate during training. Therefore, we fed the time-conditioning signal into a single hidden-layer Multilayer Perception (MLP) with Tanh activation functions prior to combining it with the input features. We composed this time embedding module with a convolutional U-Net architecture (Ronneberger et al., 2015), which gave us the biggest performance boost, for our experiments. We defer additional details to Appendix D.1.

In both the time-wise and joint score matching objectives in Eq. (4) and Eq. (7), we must backpropagate through the network with respect to the time-conditioning signal t. This requires care in designing the embedding mechanism as well as the network architecture to avoid training instabilities. Below, we list additional details as well as some empirical observations that led to good performance in practice:

1. For the time embeddings, we used a single-hidden layer MLP of the following form: `Linear(1, 256)` → `Tanh` → `Linear(256, 256)` → `Tanh` → `Linear(256, 256)`.

2. The `Swish` activation function (Ramachandran et al., 2017) and Group Normalization (Wu and He, 2018) led to the most stable training in our score networks for the MNIST experiments. For all other synthetic experiments, we used the `ELU` activation function. In general, we found that commonly used activation functions such as `ReLU` and `LeakyReLU` hurt performance, as backpropagating through the network during training would zero out gradients.

3. We found that architecture backbones based on ResNets (He et al., 2016) led to unstable training.

4. For our model architecture, we used a standard convolutional U-Net with channels of increasing resolution [64, 128, 256, 512]. The details of this architecture can be found in Table 2. We note that after each convolution, the Dense activation block is applied to the time embedding and added to the convolved input feature. Then, the output of this operation is passed through a Group Normalization layer and the `Swish` activation function.

5. As in a standard U-Net: the output of the 3rd convolutional block (combined with the time embedding, plus normalization/activation) is concatenated with the input to `tconv3`, the output of the 2nd convolutional block is concatenated with the previous output into into `tconv2`, etc.

6. In our convolutional U-Net score network, we incorporated the outputs of the time-conditioning MLP module via `Dense activation` blocks. This block is also a single-hidden layer MLP with with `Tanh` activations of the following structure: `Linear(256, 32)` → `Tanh` → `Linear(32, 32)` → `Tanh` → `Linear(32, U-Net channel)`, where 256 corresponds to the output size of the time-embedding module.

E Leveraging the Numerical Integrator for Density Ratio Estimation

After training a time-conditioned score network with Eq. (4), it is straightforward to see that $\log r(\mathbf{x})$ can be obtained via the following formula:

$$\log r(\mathbf{x}) = \int_1^0 \frac{\partial}{\partial t} \log p_t(\mathbf{x}) \mathrm{d}t \approx \int_1^0 \boldsymbol{s}_{\boldsymbol{\theta}}^{\text{time}}(\mathbf{x}, t) \mathrm{d}t.$$

The integration over all intermediate time scores in Eq. (6) can be computed using any existing numerical integrator. In our experiments, we leverage a black-box ODE solver to perform the integration, though we emphasize that using an ODE solver is not strictly necessary. The ODE solver determines the timesteps t we should query along the trajectory as we obtain our density ratio estimates, which eliminates the need to hand-tune T as in Eq. (2). For computing the likelihood ratios as in Eq. (6) in all our experiments, we follow (Grathwohl et al., 2018; Song et al., 2021b) and use the RK45 ODE solver (Dormand and Prince, 1980) in `scipy.integrate.solve_ivp` with `atol=1e-5` and `rtol=1e-5`. To avoid numerical issues in practice, we set the limits of integration to be $(1, 1\mathrm{e}{-5})$.

Name	Component
Encoding Block	
conv1	3×3 conv, 64 filters, stride 1, bias=False
Dense Activation Block 1	input dim=256, output dim=64
Group Normalization 1	num groups=4, num channels=64
conv2	3×3 conv, 128 filters, stride 2, bias=False
Dense Activation Block 2	input dim=256, output dim=128
Group Normalization 2	num groups=32, num channels=128
conv3	3×3 conv, 256 filters, stride 2, bias=False
Dense Activation Block 3	input dim=256, output dim=256
Group Normalization 3	num groups=32, num channels=256
conv4	3×3 conv, 512 filters, stride 2, bias=False
Dense Activation Block 4	input dim=256, output dim=512
Group Normalization 4	num groups=32, num channels=512
Decoding Block	
tconv4	3×3 2d convtranspose, 128 filters, stride 2, bias=False
Dense Activation Block 5	input dim=256, output dim=256
Group Normalization 5	num groups=32, num channels=256
tconv3	3×3 2d convtranspose, 128 filters, stride 2, bias=False
Dense Activation Block 6	input dim=256, output dim=128
Group Normalization 6	num groups=32, num channels=128
tconv2	3×3 2d convtranspose, 64 filters, stride 2, bias=False
Dense Activation Block 7	input dim=256, output dim=256
Group Normalization 7	num groups=32, num channels=64
tconv1	3×3 2d convtranspose, 1 filter, stride 1
Fully Connected Layer	input dim=784, output dim =1

Table 2: Convolutional U-Net architecture used for the energy-based modeling experiments for MNIST.

F Additional Experimental Results

F.1 Synthetic Experiments

1-D Gaussians. As a warm-up, we evaluated whether the joint score matching objective in Section 5.2 is able to recover the true log-ratios of two 1-dimensional Gaussian distributions. We experimented with $p(\mathbf{x}) = \mathcal{N}(0,1)$ and $q(\mathbf{x}) = \mathcal{N}(0,\sigma^2)$ on two tasks of increasing difficulty, where $\sigma^2 = 1$ and $\sigma^2 = 1\mathrm{e}{-6}$ respectively. We use the arithmetic interpolation scheme of $\mathbf{x}(t) = \sqrt{1-t^2}\mathbf{y} + t \cdot \mathbf{x}$ for $\mathbf{x} \sim q(\mathbf{x})$ and $\mathbf{y} \sim p(\mathbf{x})$, and use SSM to learn the data scores. Note that because both p and q are Gaussian, the form of the intermediate densities $p_t(\mathbf{x}) = \mathcal{N}(0, 1-(1-\sigma^2)t^2)$ can be obtained analytically.

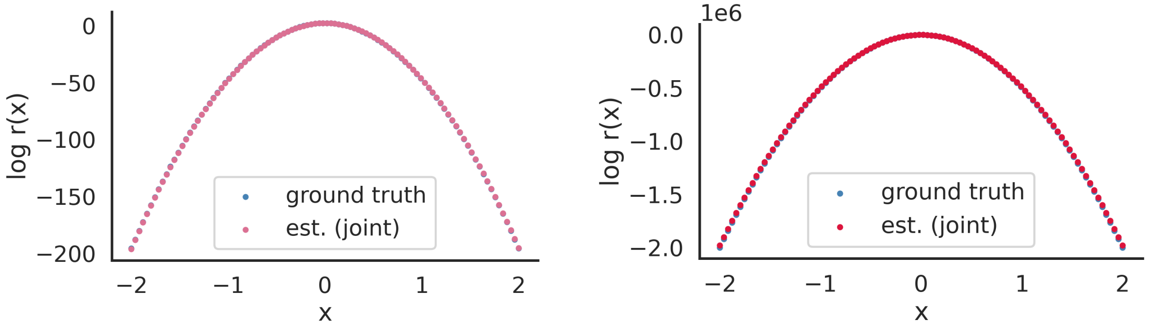
Because the discrepancy between p and q is extremely large in these settings, we follow (Rhodes et al., 2020) and endow the score network with the true parametric forms of $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ and $\frac{\partial}{\partial t} \log p_t(\mathbf{x})$. This way, the score network has to recover a single scalar parameter $\theta \in \mathbb{R}$. Specifically, we have:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = \frac{-(\mathbf{x}-\theta)}{(1-[1-\sigma^2]\cdot t^2)}$$

$$\frac{\partial}{\partial t} \log p_t(\mathbf{x}) = \frac{\left[-(\theta - x)^2 - (1-\sigma^2)\cdot t^2 + 1\right]\cdot t(1-\sigma^2)}{(1-[1-\sigma^2]\cdot t^2)^2}$$

Using this parameterization, we train the score model with the Adam optimizer with a learning rate of 0.001 for 10,000 steps using a batch size of 128. As shown in Figure 6, we find that the joint score network is able to recover the true θ*.

2-D Gaussians. In this setup, we remove the parameterization of the score network and directly learn the scores from data. We use fully-connected MLPs for all methods, including: (a) `NCE` (a single binary classifier);



(a) $p(\mathbf{x}) = \mathcal{N}(0, 1)$ and $q(\mathbf{x}) = \mathcal{N}(0, 0.01)$

(b) $p(\mathbf{x}) = \mathcal{N}(0, 1)$ and $q(\mathbf{x}) = \mathcal{N}(0, 1\text{e}{-6})$

Figure 6: Synthetic 1-D Gaussian example, demonstrating that the parameterized joint score network trained with SSM is able to recover the ground truth density ratios.

(b) TRE with 4 bridge distributions (`TRE(4)`); (c) TRE with 9 bridge distributions (`TRE(9)`); (d) the Time-only score network; and (e) the Joint score network. We use the VPSDE interpolation schedule for all methods.

For both TRE models, we found that naively using a deeper network hurt performance. Therefore, we used a single input layer and single hidden layer both of size `z_dim`=256 that were first used to transform all input features (and thus were shared across all bridges). This kind of parameter sharing was reported to be helpful in (Rhodes et al., 2020). We then used 1 linear classification head per intermediate density. We found that `LeakyRELU` activations with coefficient 0.3 worked the best.

For the time score model, we used an MLP with `ELU` activations with 2 hidden layers. (input dimension + 1 because we concatenate the minibatch of sampled times to the data). For the joint score network, we used an MLP with a single input layer and a single hidden layer that output twice the number of usual output features, then then split the output features into 2 blocks (one for the time scores, and another for the data scores). The time score-specific head was an MLP with 2 hidden layers, and the data-score specific head was an MLP with 2 hidden layers. The baseline classifier was an MLP with 2 hidden layers, the same as the time score network.

In terms of hyperparameters, we swept through batch size={128,256}, learning rate={2e-4,5e-4,1e-3}, `z_dim`={128,256} and used the best model configurations for all methods.

We summarize the method-specific model architectures below:

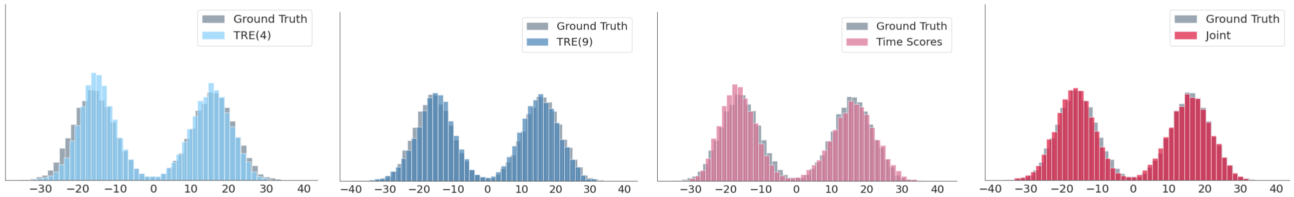
1. **NCE:** `Linear(2, 256)` → `ELU` → `Linear(256, 256)` → `ELU` → `Linear(256, 256)` → `ELU` → `Linear(256, 1)`.

2. **TRE:** `Linear(3, 256)` → `LeakyReLU(0.3)` → `Linear(256, 256)` → `LeakyReLU(0.3)` → `[Linear(256, 1) for _ in range(num_bridges)]`.

3. **Time:** `Linear(3, 256)` → `ELU` → `Linear(256, 256)` → `ELU` → `Linear(256, 256)` → `ELU` → `Linear(256, 1)`.

4. **Joint (Shared):** `Linear(3, 256)` → `ELU` → `Linear(256, 512)` → `chunk(2)` →
   (a) **Time Module:** `Linear(256, 256)` → `ELU` → `Linear(256, 256)` → `ELU` → `Linear(256, 1)`
   (b) **Data Module:** `Linear(256, 256)` → `ELU` → `Linear(256, 256)` → `ELU` → `Linear(256, 2)`

Additionally, we incorporate results for the pathwise method on the original $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{4}, \boldsymbol{I})$ setup as in Figure 2, rather than the more challenging evaluation setup in the main text with $q(\mathbf{x}) = \mathcal{N}(\mathbf{5}, \boldsymbol{I})$. We omit results for the naive baseline for clarity (and its poor performance). As shown in Figure 7(d), we find that the pathwise method outperforms all other methods as expected.



(a) TRE(4) MSE: 1.1 (b) TRE(9) MSE: 0.63 (c) Time MSE (Ours): 0.58 (d) Joint MSE (Ours): 0.35

Figure 7: Motivating example on a synthetic 2-D Gaussian dataset, with learned density ratio estimates by method relative to the ground truth values for (a-d). The performance of TRE improves with more intermediate bridge distributions, while our score matching method outperforms the rest. The x-axis denotes the log-ratios.

F.2 Mutual Information (MI) Estimation

For MI estimation between two high-dimensional correlated Gaussians, we follow the setup of (Rhodes et al., 2020) and parameterize the score network such that it only needs to learn a single $d \times d$ matrix (corresponding to $\boldsymbol{S}$ below). Similar to the 1-D Gaussians experiment, we use the arithmetic interpolation scheme to sample data points from all the intermediate densities. Concretely, we note that $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. Due to the way that we interpolate, the intermediate distributions are $p_t(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I} + t^2(\boldsymbol{\Sigma} - \boldsymbol{I})) = \mathcal{N}(\mathbf{0}, \boldsymbol{I} + \boldsymbol{S}t^2)$, where we let $\boldsymbol{S} = (\boldsymbol{\Sigma} - \boldsymbol{I}) \in \mathbb{R}^{d \times d}$ for notational convenience. Then, our joint score network is trained to output:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = -\boldsymbol{M}\mathbf{x}$$

$$\frac{\partial}{\partial t} \log p_t(\mathbf{x}) = -t \cdot \mathrm{tr}(\boldsymbol{S} \cdot \boldsymbol{M}) + \mathbf{x}^\top \boldsymbol{M}\boldsymbol{S}\boldsymbol{M}\mathbf{x}$$

where $\boldsymbol{M} = (1 + t^2\boldsymbol{S})^{-1}$.

For $d = \{40, 80, 160\}$, we use batch sizes of 512 and use a batch size of 256 for $d = 320$. We use the Adam optimizer with learning rate 0.001 with weight decay of 0.0005 for all settings. We train for $\{30K, 50K, 200K, 200K\}$ steps for $d = \{40, 80, 160, 320\}$ respectively. We note that we used a heuristic weighting function that led to good performance for this experiment. Specifically, we let $\lambda(t) = (1 - t^2)$ due to the way that we construct the intermediate samples $\mathbf{x}(t)$.

For TRE, we used their default hyperparameter settings and architecture details and refer the reader to (Rhodes et al., 2020) for more details on the exact experimental setup. In terms of the number of intermediate densities, we used $\{2, 4, 6, 8\}$ bridge distributions for $d = \{40, 80, 160, 320\}$ respectively.

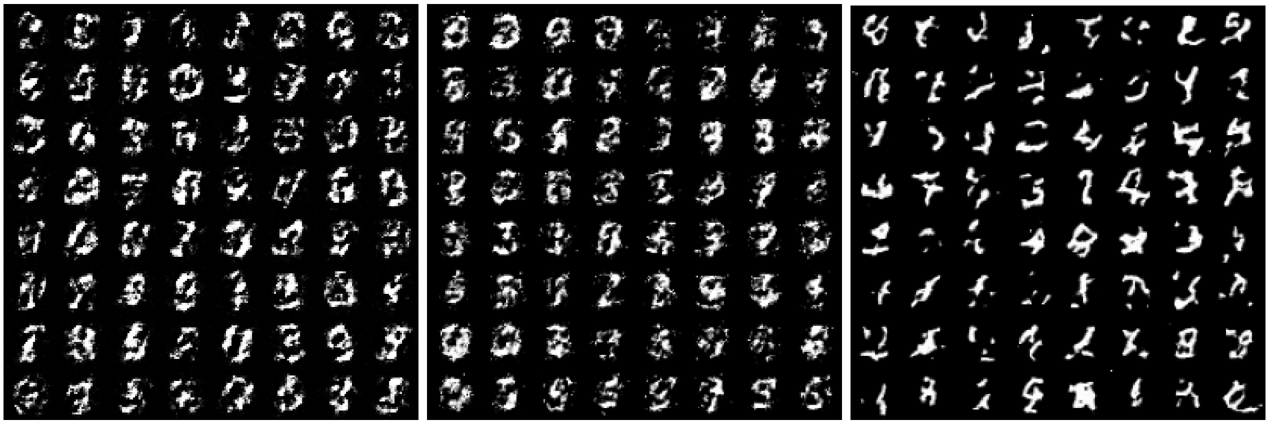
F.3 Energy-Based Modeling with MNIST

Our experimental setup largely mirrors that of (Rhodes et al., 2020) and refer the reader to their paper for additional details.

Data Preprocessing. To match the experimental setting as closely as possible, we first rescale the pixels to lie in $[0, 1]$, apply uniform dequantizatiation, and logit-transform the dequantized pixel values. Then, we whiten the transformed dataset by subtracting off the mean before training our flow models. For training the score network on the data distribution $q(\mathbf{x})$, we rescale the pixel values to lie between $[-1, 1]$.

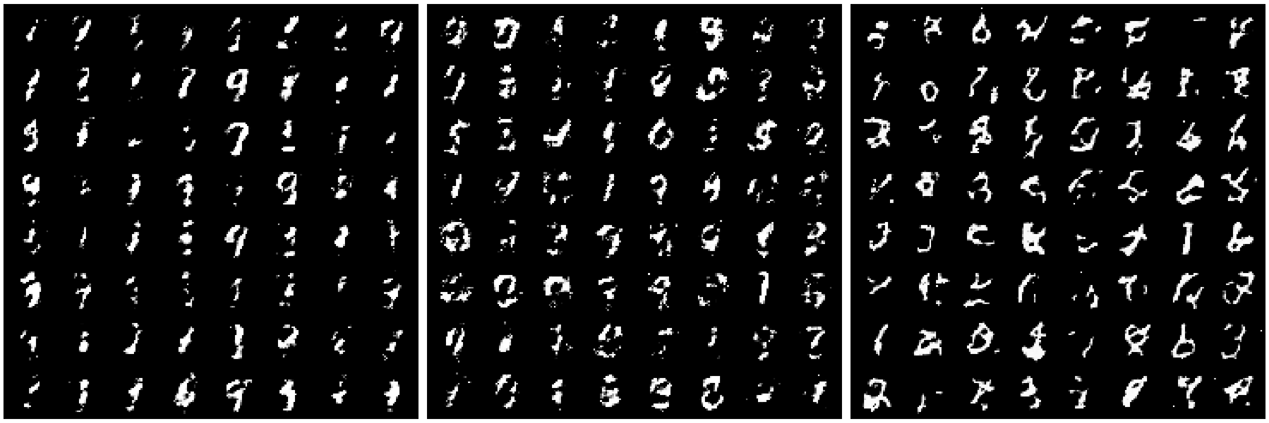
Fitting noise distributions. We experiment with three different interpolation schemes, which first require training a "normalizing flow" (invertible transformation) for each setting on the MNIST (LeCun, 1998) dataset. That is, our prior distribution $p_1(\mathbf{x})$ is the density captured by a pretrained flow on MNIST, and we can utilize the flow mapping to interpolate in the latent space $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We found this latent interpolation procedure to be critical for the success of all methods involved in this experiment.

For the copula, we use a batch size of 512 and train for 40K iterations. For both the copula and the RQ-NSF, we use a multi-scale convolutional neural network (CNN) with 2 levels, where each level contains 8 steps. The coupling transofrms use 64 feature maps and the spline functions use 8 bins with the interval width between



(a) Gaussian: 2.01 bpd (b) Gaussian Copula: 1.44 bpd (c) RQ-NSF: 1.12 bpd

Figure 8: Samples from the transformed noise distributions $p(\mathbf{x})$ in the energy-based modeling experiments for MNIST: (a) Gaussian model (an affine transformation); (b) Gaussian copula parameterized by the Rational Quadratic Spline building block; (c) the RQ-NSF flow.



(a) Gaussian: 1.33 bpd (b) Gaussian Copula: 1.21 bpd (c) RQ-NSF: 1.08 bpd

Figure 9: Samples obtained from running AIS with 100 parallel chains for 1000 steps for the energy-based modeling experiments for MNIST: (a) Gaussian model (an affine transformation); (b) Gaussian copula parameterized by the Rational Quadratic Spline building block; (c) the RQ-NSF flow.

[-3, 3]. We use a learning rate of 0.0001 and use a cosine annealing decay schedule. For the RQ-NSF, we use a learning rate of 0.0005 and train for 200,000 steps with a batch size of 256. This allowed us to match the initial Noise Distribution bpds of (Rhodes et al., 2020) with the exception of the Gaussian copula, where we were unable to improve upon a bpd of 1.44. However, we note that our method was still able to outperform the TRE baseline in Section 6.3. Samples from all noise distributions are shown in Figure 8.

Training the score networks. For training the score networks, we perform a hyperparameter sweep where `lr`={2e-4, 5e-4, 1e-3}, `batch_size`={128, 256, 500}, and we use the loss history as described in Section D.1. For the interpolation mechanism, we use the VPSDE noise schedule in latent space. For evaluating likelihoods, we use EMA with rate=0.999. For the score network architecture, we use a convolutional U-Net (Ronneberger et al., 2015) with a smaller MLP for the time embeddings. For the RQ-NSF setting, the powerful flow network made it challenging for the score network to make additional improvements on the likelihoods. For this model, we did not use the loss history, and directly reweighted the loss using the VPSDE reweighting scheme, which performed the best out of all configurations.

AIS and likelihood evaluation For likelihood evaluation, we computed the bpds directly from the score network and also via AIS. We ran AIS with 100 parallel chains for 1000 steps. We used Hamiltonian Monte Carlo (HMC) for MCMC, where we conducted the sampling in z-space and mapped the results back to x-space. Samples obtained from running AIS are shown in Figure 9.

F.4 Exploration of DRE-∞'s computational gains

We note that TRE's ResNet architecture did not perform well for score estimation (and vice versa), so we used smaller U-Nets (Salimans and Ho, 2021). On MNIST with the original TRE codebase, TRE with 10 bridges ("TRE-10") has 7.4M trainable parameters, while our time score network has 3.7M parameters. TRE-30, on the other hand, has 19.2M parameters. However, we do need to train longer to converge – while TRE converges in about a day, our models took roughly 2 days. We expect improvements in optimization as well as variance reduction to accelerate training.

We explore how many time score evaluations typically occur while performing the numerical integration at test time. As expected, the average number of function evaluations varies with the difficulty of the task. For example on the MNIST dataset, the score network trained with the Gaussian noise model requires 266.6 ± 13.3 function 61 evaluations at an error tolerance of $1e5$, 199.4 ± 5.7 evaluations for the copula noise model, and 118.4 ± 2.9 evaluations for the RQ-NSF.

In terms of wall-clock time, this helps our approach perform favorably against TRE when evaluating the log-ratios after training. For the RQ-NSF noise model for a batch of 100 examples, TRE-10 took $1.26 \pm .02$s, TRE-15 took $1.42 \pm .07$s, TRE-30 took 2.01 ± 0.27s, and ours took 0.70 ± 0.02s at error tolerance $1e5$.

G Societal Impact

Ultimately, the goal of this work is to provide more accurate density ratio estimates for a wide variety of machine learning applications. While DRE-∞ in itself does not have any direct social implications, its use in downstream applications such as domain adaptation, anomaly detection, and propensity score matching in causal inference, etc. may have consequences depending on their use case.