Learning Energy-Based Models from Stochastic Interpolants using Spatiotemporal Differences

Hanlin Yu
University of Helsinki

RuiKang OuYang
University of Cambridge

Partha Kaushik
Carnegie Mellon University

Arto Klami
University of Helsinki

Michael U. Gutmann
University of Edinburgh

Omar Chehab
Carnegie Mellon University

Abstract

Learning an energy-based model from data samples is a central problem in machine learning. Many recent and popular methods, such as denoising score matching for training energy-based diffusion models, use stochastic interpolants to corrupt data samples at different noise levels indexed by a time variable. This defines a joint density over both the data space and time, and most methods learn its energy through either spatial or temporal differences. We identify distinct failure modes for both of these approaches. To solve them, we propose Spatiotemporal Noise-Contrastive Estimation (stNCE), a framework for learning the energy through joint spatiotemporal differences. stNCE unifies many existing methods and leads to new training objectives. Experiments on images and molecules demonstrate performance competitive with state-of-the-art density estimation methods. Code is available at `https://github.com/ksnxr/stnce`.

1 Introduction

A central problem in machine learning is to learn the underlying probability density of a data distribution from finite samples. Accurate density models are useful in a wide range of applications. In computational chemistry, for example, they can be used to correct samples from a proposal distribution to the equilibrium distribution [59]. They also enable sampling from compositions of densities (e.g., products or tempered variants) which is used to combine generative models in a modular way [24; 65]. To support these applications efficiently, recent work has focused on learning likelihoods that can be evaluated in *one shot*, i.e., with a single neural network evaluation [1; 34; 2; 61]. A natural approach is to parameterize the likelihood as an Energy-based Model (EBM), where a neural network defines an unnormalized density. While EBMs also support sampling, one can alternatively use a separate generative model to draw samples [1], and we focus on density estimation.

Over the past two decades, a wide range of methods has been proposed for training energy-based models [67; 36; 13; 38; 43; 64; 20; 27]. In this work we focus on contrastive estimation approaches, showing that many established methods like Noise-Contrastive Estimation, score matching and contrastive divergence can be unified under the same framework. Recent works also looked into training flows in conjunction with EBMs [27; 32] and there is an established literature on training EBMs based on likelihoods [42]. Most approaches proceed in two steps. First, they define corrupted versions of the data by interpolating between data samples and samples from a reference distribution (e.g., a Gaussian) [64; 3]. This introduces a *time* variable t indexing corruption levels and defines a joint density $p(x, t)$ over the data space and time. Second, the model is trained by estimating how this density varies, either across time or across space. This viewpoint yields our taxonomy. *Temporal methods* estimate variations across corruption levels. For instance, Noise-Contrastive Estimation (NCE) learns large temporal differences between data and reference distributions [36], while sub-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

sequent work improves performance by focusing on smaller temporal differences, as in telescoping density ratio estimation and time score matching [64; 20; 85; 17]. *Spatial methods*, in turn, estimate variations across data points. For instance, Conditional NCE [13] learns spatial difference between nearby data points. When these points are infinitesimally close, this becomes score matching [43] that is widely used to train diffusion-based EBMs [24; 75].

While both approaches for learning energy-based models have been widely applied [43; 69; 81; 64; 20; 85; 17], they are known to suffer from catastrophic failure modes that remain poorly understood. For example, spatial methods such as score matching [43] can incur large estimation errors in multi-modal settings [82; 48]. Conversely, temporal methods based on large differences, such as Noise-Contrastive Estimation, can fail when the high-density regions of the data distribution (its "support") do not overlap with those of the reference distribution. However, these failure modes are rarely stated explicitly or analyzed in a unified manner in the literature. Often the analysis is limited to behavior of a specific method, instead of the broader approach.

A widely held view is that these challenges can be alleviated by learning across multiple corruption levels (or times). The intuition is that some levels are inherently easier than others, and that *amortization*, through parameter sharing across levels, allows harder tasks to benefit from the easier ones. Empirically, this strategy has led to improvements in both temporal and spatial methods. On the temporal side, multi-level NCE [64] has been shown to improve sample efficiency, with some theoretical support [64; 14]. On the spatial side, multi-level denoising score matching for energy-based diffusion models [75] enjoys emerging theoretical justification [63; 19]. Yet, this perspective is not fully satisfactory. Recent work indicates that such approaches do not fundamentally resolve the underlying issues: Srivastava et al. [73] show that multi-level NCE does not eliminate support mismatch, while Yu et al. [84] provides empirical evidence that multi-level score matching continues to struggle in very simple multimodal settings.

Overall, our understanding of the failure modes of existing methods remains limited. Progress in the field has largely emphasized the development of new techniques, rather than the analysis of the limitations they aim to address. Although recent approaches report strong empirical gains [37; 34; 62; 1; 60], we still lack a clear understanding of when and why they outperform prior methods.

Contributions We revisit the problem of learning EBMs through the lens of spatial and temporal variations. We first introduce a unifying taxonomy that organizes existing methods along these two axes. We then explicitly identify their limitations when used in isolation: temporal methods struggle under support mismatch, while spatial methods fail in multi-modal settings, *even when learning across different times* (corruption levels), contrary to popular belief.

Motivated by these observations, we propose to jointly learn spatial and temporal variations. We formalize this as *Spatiotemporal Noise-Contrastive Estimation* (stNCE), a general framework that unifies existing approaches and addresses the previous limitations. While some very recent works empirically combine spatial and temporal variations, they lack a clear conceptual justification and a unified framework [37; 34; 62; 1; 60]. Our framework unifies known methods and shows, perhaps remarkably, that they are *all* instances of a *single* binary classification task. Our framework leads also to new training objectives that are competitive with the state-of-the-art across synthetic, image, and molecular benchmarks, while providing a clearer understanding of when and why they succeed.

2 Background

Stochastic Interpolants Stochastic interpolants (SIs) [64; 3] can be used to sample from a *joint* density $p_d(x, t)$ over data points x and their corruption levels t (0 is fully corrupted, 1 is clean), where we use $p_d(x|t)$ to denote the time-varying density induced by the SIs given the data distribution. SIs interpolate between clean data samples $x_1 \sim p_1$ and samples $x_0 \sim p_0$ from a reference distribution p_0, which is often a standard Gaussian. A simple interpolation is [53]

$$x_t = (1 - t)x_0 + tx_1, \tag{1}$$

but other choices are possible too [64]. These interpolation schemes can be used to learn score fields, velocity fields and densities. In this work, we focus on learning the densities as time-varying energy-based models. The joint density can be modelled as $p_\theta(x, t) = p(t)p_\theta(x|t)$, where $p(t)$ is a distribution over corruption levels chosen by the user, for example uniform between 0 (fully corrupt

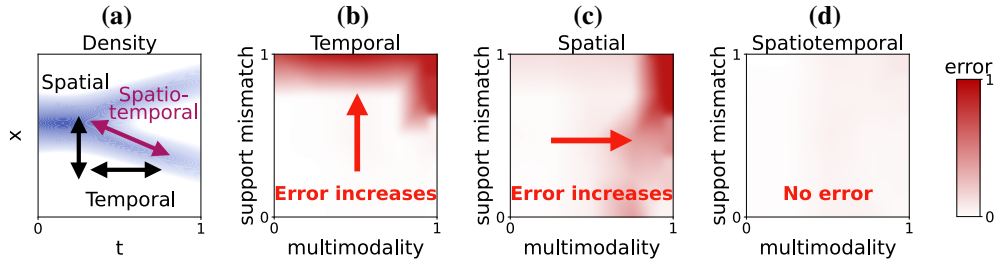


Figure 1: *(a): interpolant density* $p_d(x,t)$. We learn it via spatial differences which vary x but fix t, temporal differences which vary t but fix x, or spatiotemporal differences that vary both x and t. *(b), (c), (d): empirical error of different learning methods* when trained to model the stochastic interpolant between standard normal noise and data given by a one-dimensional Gaussian mixture with two components, where the means vary and the standard deviation is fixed. Temporal methods incur high estimation error when the data distribution has high density in low density region of the noise distribution (the means of data components are far from high density region of standard noral noise, abusively, "support mismatch"). Spatial methods incur high estimation error when the data distribution is multi-modal (the two components are far apart). Our spatiotemporal method achieves here virtually zero error in both setups. Further details are provided in Section 6.1 and Appendix E.4.

data) and 1 (clean data). The time-dependent density $p_\theta(x|t)$ is an energy-based model

$$p_\theta(x|t) = \exp(-E_\theta(x,t) - \log Z_t), \tag{2}$$

where $E_\theta(x,t)$ is typically a neural network and $Z_t = \int \exp(-E_\theta(x,t))\,dx$ is the normalizing constant, which is generally intractable. The goal is to learn a parameter θ such that the model of the clean data density $p_\theta(x \mid 1)$ is accurate. A popular way to estimate θ without having to compute the intractable Z_t is to model variations of the joint density.

Binary classification for learning variations A seminal approach for learning the difference between two log densities $\log p_A(x)$ and $\log p_B(x)$ is to solve a binary classification task. This is core to many algorithms [36; 13; 30], including the ones in this paper. Consider data from two classes $p_A(x)$ and $p_B(x)$ and a binary classification task minimizing the logistic loss

$$\mathcal{L}_{\text{logistic}}(F) = -\mathbb{E}_{x\sim p_A}\left[\log \text{sigmoid}(F(x))\right] - \mathbb{E}_{x\sim p_B}\left[\log(1-\text{sigmoid}(F(x)))\right]. \tag{3}$$

The optimal classifier is the *difference* between log densities $F(x) = \log p_A(x) - \log p_B(x)$ [74].

Learning the energy by temporal differences. One class of methods learns the energy by estimating *temporal* differences $E(x,t') - E(x,t) + \log Z(t') - \log Z(t)$. A canonical approach is Noise-Contrastive Estimation (NCE) [36], using binary classification as above. First, generate data x from two classes A and B with equal probability, where the classes correspond to the densities of x under different times of the stochastic interpolant induced by the data distribution, t and t'

$$p_A(x) = p_d(x|t), \quad p_B(x) = p_d(x|t'). \tag{4}$$

For example, generating data from class A is done by corrupting a clean data sample x_1 with level t following Eq. 1. Then, minimizing the logistic loss yields the optimal classifier

$$F(x,t,t') = -E(x,t) + E(x,t') - \log Z(t) + \log Z(t'), \tag{5}$$

that equals the temporal energy difference. In practice, we jointly parameterize $E_\theta(\cdot,\cdot)$ and $\log Z(\cdot)$ using a single neural network and learn them. Rhodes et al. [64] observed that the performances become stronger when t and t' are sampled between 0 and 1 and close to each other; this was later generalized to continuous time [1; 60], where one define a specific $p(t)$ and sample $t' \sim p(t'|t)$, where $p(t'|t)$ can be close to $\mathcal{N}(t';t,\sigma^2)$ for some small σ. In the limit where t and t' are infinitesimally close, that energy difference approaches a quantity determined by the *time score* $\partial_t \log p_t(x)$ [20; 85]. We provide further details on $p(t)$ and $p(t'|t)$ in Appendix D.1. We formalize this algorithm family as temporal NCE (tNCE) and present the resulting loss as follows

$$\mathcal{L}_{\text{tNCE}}(\theta) = -\mathbb{E}_{p_d(x,t)p_n(t'|t)}\left[\log \text{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x|t') + c(t,t'))\right] \tag{6}$$

$$- \mathbb{E}_{p_d(x,t')p_n(t|t')}\left[\log\left(1-\text{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x|t') + c(t,t'))\right)\right], \tag{7}$$

where $c(t,t') = \log p(t) + \log p_n(t' \mid t) - \log p(t') - \log p_n(t \mid t')$.

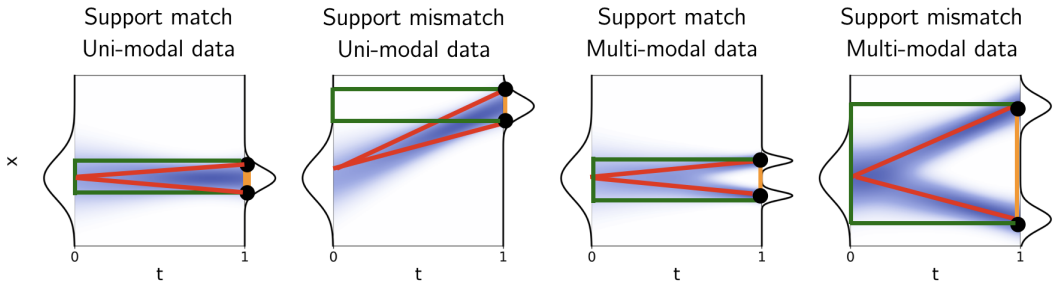


Figure 2: Ability to compute log-likelihood differences between two data points using learned spatial (orange), temporal (green), or spatiotemporal (red) differences. Methods based purely on spatial or temporal differences *fail when traversing low-density regions* (white), where these differences are poorly estimated. In contrast, spatiotemporal methods remain within high-density regions (blue), enabling accurate estimation.

Learning the energy by spatial differences. Another class of methods learns the energy by estimating *spatial* differences $E(x', t) - E(x, t) + \log Z(t) - \log Z(t)$. A fundamental example is Conditional Noise-Contrastive Estimation (CNCE) [13], which also solves a binary classification task. First, generate data (x, x') from two classes A and B

$$p_A(x, x') = p_d(x|t)p_n(x' \mid x), \quad p_B(x, x') = p_d(x'|t)p_n(x \mid x') \tag{8}$$

To generate data from class A, first generate a data point x with corruption level t following Eq. 1, and then perturb it into x', using a perturbation kernel p_n chosen by the user. Switching the positions of x and x' yields data from class B. Then, minimizing the logistic loss yields the optimal classifier

$$F(x, x', t) = -E(x, t) + E(x', t) + \log p_n(x' \mid x) - \log p_n(x \mid x'). \tag{9}$$

It equals the spatial energy difference up to known terms. We parameterize $E(\cdot, \cdot)$ using a neural network. As x and x' become infinitesimally close, the energy difference is determined by the *space score* $\nabla_x \log p_t(x)$ [43; 13]. Based on extensions of score matching to continuous time [66; 71], a natural continuous time extension of CNCE is to sample t between 0 and 1 and take expectation over $p(t)$. We formalize this algorithm as temporal CNCE (tCNCE), where the loss is

$$\mathcal{L}_{\text{tCNCE}}(\theta) = -\mathbb{E}_{p_d(x,t)p_n(x'|x)} \left[\log \text{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x'|t) + c(x, x'))\right] \tag{10}$$

$$- \mathbb{E}_{p_d(x',t)p_n(x|x')} \left[\log \left(1 - \text{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x'|t) + c(x, x'))\right)\right]. \tag{11}$$

We again pack the terms independent of θ into the notation $c(x, x') = \log p_n(x'|x) - \log p_n(x|x')$.

3 Limitations of current methods

Despite widespread success, current methods suffer from failure modes. It is well known that methods based on spatial differences [12; 81] and temporal differences [36] can fail even in simple settings. A common belief is that these issues can be mitigated by *learning across multiple corruption levels*, through temporal schemes [64; 20; 85; 17] or spatial ones [24; 75; 71]. Here we argue—and empirically demonstrate—that this is *not* the case: they still fail in the same simple regimes.

A unifying heuristic. To analyze these failures, we study the ability of a model to relate the log-density of two points. For any two points, (x, t) and (x', t'), we decompose the difference into spatial and/or temporal differences that need to be learnt by the model. This viewpoint captures both local properties (e.g., mode geometry for nearby points) and global properties (e.g., relative mode weights for distant points). Crucially, we will show that purely spatial or temporal decompositions inevitably require evaluating $p(x, t)$ in low-density regions, where samples are scarce and estimation is therefore worse. Figure 1 empirically validates these claims to hold even for a bivariate density.

Spatial differences struggle with multimodality. Spatial methods like tCNCE learn $p_\theta(x, t)$ via differences in x, where one interpolates between x and y and discretizes it as x_k such that $x_0 = y$

and $x_K = x$. Using our heuristic, we decompose the log-density difference between x and y as

$$\log p(x, 1) - \log p(y, 1) = \sum_{i=0}^{K-1} \left(\log p(x_{i+1}, 1) - \log p(x_i, 1)\right). \tag{12}$$

When p_1 is multimodal (Figure 2, two right panels), these spatial differences in orange necessarily traverse low-density regions (white). As a result, spatial methods—including energy-based diffusion models—struggle to capture global structure such as mode weights [84].

Temporal differences suffer from support mismatch. Temporal methods learn $p_\theta(x, t)$ via differences in t. For tNCE, using our heuristic, we discretize t into a grid in the form of t_k where $t_0 = 0$ and $t_K = 1$, and decompose the log-density difference between x and y as

$$\log p(x, 1) - \log p(y, 1) = \log p(x, 0) - \log p(y, 0) + \Delta, \tag{13}$$

where $p(\cdot, 0)$ is a known reference distribution, and

$$\Delta = \sum_{i=0}^{K-1} \big(\log p(x, t_{i+1}) - \log p(x, t_i)\big) - \sum_{i=0}^{K-1} \big(\log p(y, t_{i+1}) - \log p(y, t_i)\big). \tag{14}$$

When the supports of p_0 and p_1 differ (Figure 2, second and fourth panels), these temporal differences in green traverse low-density regions in white. As before, this degrades estimation.

Spatiotemporal differences avoid both issues. We now consider a joint discretization in space and time, combining the two discretizations above. Let $\{(x_i, y_i, t_i)\}_{i=0}^K$ with $x_i, y_i \sim p_{t_i}$. Then

$$\log p(x, 1) - \log p(y, 1) = \log p(x_0, 0) - \log p(y_0, 0) + \Delta, \tag{15}$$

where $\Delta = \sum_{i=0}^{K-1} \big(\log p(x_{i+1}, t_{i+1}) - \log p(x_i, t_i)\big) - \sum_{i=0}^{K-1} \big(\log p(y_{i+1}, t_{i+1}) - \log p(y_i, t_i)\big)$. As illustrated in Figure 2, these spatiotemporal differences in red can be chosen to remain within high-density regions in blue. Since such regions are well sampled, the corresponding spatiotemporal differences are accurately estimated. This observation may explain the success of recent spatiotemporal methods [34; 37; 1; 60; 62], as they simultaneously address multimodality and support mismatch. We empirically validate this conjecture in Figure 1. This motivates the following unifying framework for spatiotemporal methods. We discuss the relationship between our stNCE framework and related spatiotemporal methods in Appendix C.2.

4 Learning the energy by spatiotemporal differences

In this section, we introduce a framework for estimating the parameters of an EBM by learning its variation jointly in data space and time. Our approach reduces to a *single* binary classification problem that unifies many popular methods [43; 81; 36; 13; 64; 20; 85; 17; 34; 1; 60]; see Table 3.

Spatiotemporal Noise-Contrastive Estimation (stNCE) We propose to learn the energy by estimating its *spatiotemporal* differences, $E(x', t') - E(x, t)$ through binary classification. We term this Spatiotemporal Noise-Contrastive Estimation (stNCE). We start from two classes A and B

$$p_A(x, t, x', t') = p_d(x, t) p_n(x', t' | x, t), \quad p_B(x, t, x', t') = p_d(x', t') p_n(x, t | x', t'). \tag{16}$$

To generate data from class A, first sample a data point and its corruption level (x, t) following Eq. 1, and then perturb them into (x', t') using a perturbation kernel p_n that is chosen by the user. Switching the positions of (x, t) and (x', t') in the tuple yields data from class B. Strategies for sampling (x, t, x', t') are discussed in Appendix D.1. Minimizing the logistic loss yields the optimal classifier

$$F(x, t, x', t') = -E(x, t) + E(x', t') - \log Z(t) + \log Z(t') + \log p(t) - \log p(t') \tag{17}$$

$$+ \log p_n(x', t' \mid x, t) - \log p_n(x, t \mid x', t'), \tag{18}$$

which equals the spatiotemporal energy difference up to known terms. In practice, we parameterize $E(x, t)$ and $\log Z(t)$ using neural networks and learn them jointly. Note that this can be viewed as CNCE from Section 2 on the augmented space (x, t) . This leads to the following loss

$$\mathcal{L}_{\text{stNCE}}(\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)} \left[\log \text{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x'|t') + c(x, t, x', t'))\right] \tag{19}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')} \left[\log\left(1 - \text{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x'|t') + c(x, t, x', t'))\right)\right]. \tag{20}$$

For a fixed perturbation kernel, we again pack the terms independent of the optimized parameters into $c(x, t, x', t') = \log p_n(x', t'|x, t) - \log p_n(x, t|x', t') + \log p(t) - \log p(t')$.

Statistical guarantees. We next present statistical guarantees on the stNCE estimator under a fixed perturbation kernel: it is consistent and we can quantify its sample-efficiency. We remark that it applies to the forward-reverse kernel where the score estimates are potentially arbitrary but fixed, though the practical stNCE-s variant uses score estimates that are updated during optimization and hence is beyond the scope of our existing theory.

Theorem 1 (Non-parametric consistency) *Let $p_\theta(x|t) = \exp(-E_\theta(x,t))/Z_\theta(t)$ be a parametric model of the joint density where $p_\theta(x \mid 0)$ equals the known Gaussian noise, and suppose the data is generated with parameter θ^*. Assume that the fixed perturbation kernel p_n has full support for every conditioning point, i.e. $p_n(x', t'|x, t) > 0$ for all $(x, t), (x', t') \in \mathbb{R}^D \times [0, 1]$, where D is the dimensionality of x. Under standard regularity conditions ensuring convergence of empirical risk minimizers to population minimizers, the stNCE estimator $\hat{\theta}$, obtained by minimizing the empirical logistic loss with N samples, leads to a pointwise consistent estimator:*

$$p_{\hat{\theta}} \to p_{\theta^*} \quad \text{as } N \to \infty. \tag{21}$$

We prove this in Appendix A.2, and next characterize the estimator's asymptotic sample efficiency.

Theorem 2 (Sample efficiency) *(Based on Yu et al. [84]) Let*

$$F_\theta(x, t, x', t') = \log p_\theta(x, t) - \log p_\theta(x', t') + \log p_n(x', t' \mid x, t) - \log p_n(x, t \mid x', t'). \tag{22}$$

Then the estimator satisfies

$$\mathbb{E}\|\hat{\theta} - \theta^*\|^2 = \frac{1}{N_d} \operatorname{Tr}\big(C_1^{-1} C_2 C_1^{-1}\big) + o\Big(\frac{1}{N_d}\Big), \tag{23}$$

where C_1 and C_2 are given as follows, with expectations taken under $p_d(x, t)p_n(x', t' \mid x, t)$:

$$C_1 = \mathbb{E}\big[\nabla_\theta F_\theta \nabla_\theta F_\theta^\top \operatorname{sigmoid}(-F_{\theta^*}) \operatorname{sigmoid}(F_{\theta^*})\big], \quad C_2 = \mathbb{E}\big[\nabla_\theta F_\theta \nabla_\theta F_\theta^\top \operatorname{sigmoid}(-F_{\theta^*})^2\big]. \tag{24}$$

Design choices. The main design choice is the *perturbation kernel* p_n. While our stNCE estimator is consistent for any choice of p_n satisfying the assumption as in Theorem 1, its sample efficiency depends critically on this choice. As shown in Table 3, specific choices recover many existing methods based on temporal differences [36; 64; 20; 85], spatial differences [13; 43; 69], and spatiotemporal differences [34; 1; 37; 60]. In principle, one could optimize the asymptotic variance expression with respect to p_n. As is often the case, this is analytically difficult. Instead, we adopt an empirical approach and propose choices of p_n that are heuristically well suited to the problem.

5 Choices of perturbation kernels

In this section, we propose three choices for the perturbation kernels, called *mixture*, *white*, and *forward-reverse*. We argue that the first two are sample-efficient when the perturbation is "small enough", while the last is sample-efficient when a certain quantity called the "space score" is well-approximated. In both cases, the guiding principle is that the pairs (x, t) and (x', t') used to estimate energy difference should remain within high density region of $p_d(x, t)$; because one of them is sampled from $p_d(x, t)$ and is therefore likely to be on the data manifold, the perturbation kernel p_n used to obtain the second should avoid mapping samples *outside* of the data manifold, see Figure 3. We explain that certain recent methods in the literature [34; 37] can be viewed as special cases or variants on these perturbation kernels.

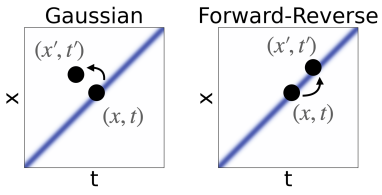


Figure 3: Choice of kernels. Left: departs from the data manifold (blue). Right: remains on the data manifold (blue).

Mixture kernel. This kernel, leading to stNCE-m variant, perturbs *either* x or t:

$$p_n(x', t' \mid x, t) = \tfrac{1}{2}\, p_n(t' \mid t)\, \delta_x(x') + \tfrac{1}{2}\, p_n(x' \mid x)\, \delta_t(t'). \tag{25}$$

It recovers a mixture of tNCE and tCNCE objectives, as we show in Appendix B.

White noise kernel. This kernel perturbs x using Gaussian noise and varies t

$$p_n(x', t' \mid x, t) = \mathcal{N}(x' \mid x, \sigma^2 I)\, p(t'|t). \tag{26}$$

We refer to the resulting algorithm as stNCE-w. Sufficiently small perturbations keep (x', t') in the data manifold, and as $\sigma \to 0$, this recovers Dual Score Matching [34], as we prove in Appendix A.4.

These two kernels suffer from the same pathology: for non-vanishing perturbations, (x', t') could naturally be generated off the high density region of $p_d(x, t)$. The manifold hypothesis [8] states that real world data lies on a low dimensional manifold, such that adding isotropic Gaussian noise often results in off-manifold samples. Furthermore, samples from Gaussian distribution concentrate near a sphere in high dimensions [80], resulting in pathologies even for noise-perturbed data. In contrast to the kernels above, the following kernel aims to stay within high density region.

Forward-reverse kernel. We propose perturbing (x, t) along trajectories consistent with $p_d(x, t)$:

$$p_n(x', t' \mid x, t) = p_n(x' \mid x, t, t')\, p_n(t'|t). \tag{27}$$

We first perturb t to t', then apply a kernel on x: a *noising* step if $t' < t$, and a *denoising* step if $t' > t$. The idealized versions of these kernels are Bayes inverses, as in diffusion-based methods [28; 22; 37]. An explicit construction based on the stochastic interpolant as in Eq. 1 is:

$$p_n(x' \mid x, t, t') = \begin{cases} \mathcal{N}\big(x' \big| (t'/t)x,\ \left((1-t')^2 - t'^2(1-t)^2/t^2\right) I\big), & t' < t, \\ \mathcal{N}\Big(x' \Big| (t'/t)x + \frac{2(1-t)(t'-t)}{t}\, \mathrm{sg}\left(\nabla_x \log p_t(x)\right),\ \frac{(t'-t)(t+t'-2tt')}{t^2} I\Big), & t' > t, \end{cases} \tag{28}$$

where sg denotes stop gradient, preventing gradient propagating through the score in the kernel, such that the kernel is indeed fixed during optimization, while refreshed after each step. The noising kernel (top) is tractable, while the denoising kernel, obtained using the exponential integrator formula [29], depends on the space score $\nabla_x \log p_t(x)$, which is typically unknown and is the key quantity that diffusion models approximate [71]. We use either: (i) `oracle` scores (when available), denoted as stNCE-o, or (ii) `self` scores, obtained from the current model, denoted as stNCE-s. The latter is reminiscent of consistency models and flow maps [72; 10]. The recent RNE [37] proposed to regularize energy-based diffusion models is related too. However, while stNCE is consistent by itself already for suitable kernels as in Theorem 1, given imperfect scores RNE is no longer consistent and cannot be used independently. See Appendix C.3 for further discussion.

Infinitesimal limit. We now characterize the behavior of stNCE for certain *infinitesimal* perturbation kernels, specifically perturbation kernels that are transition probabilities along a trajectory of a Stochastic Differential Equation (SDE) as in Eq. 28 or of an Ordinary Differential Equation (ODE), assuming oracle access to the ground truth scores. We prove the following results in Appendix A.3.

Proposition 3 (Transition probability of an SDE) *Consider an SI whose marginal laws p_t are simulated by the time-reversal of an SDE, $\mathrm{d}X_t = b(X_t, t)\mathrm{d}t + g(t)\mathrm{d}W_t$, where b and g are SDE parameters. Define the perturbation kernel to be the transition probability of that SDE. In the limit $\Delta t \to 0$, stNCE recovers the space score matching loss with maximum-likelihood weighting [70]:*

$$\mathcal{L}_{\mathrm{stNCE}}(\theta) = 2\log 2 + \frac{\Delta t}{4}\, \mathbb{E}_{p_t}\left[g(t)^2 \|\nabla_x \log p_t(x) - \nabla_x \log p_t^\theta(x)\|^2\right] + o(\Delta t). \tag{29}$$

Proposition 4 (Transition probability of an ODE) *Consider an SI whose marginal laws p_t are simulated by an ODE, $\mathrm{d}X_t = v_t(X_t)\mathrm{d}t$, where v_t is the velocity field. Define the perturbation kernel to be the transition probability of that ODE. Under this scenario, the transition kernels are Dirac deltas subject to Jacobian scalings. In the limit $\Delta t \to 0$, stNCE recovers a loss for matching the trajectory-wise rate of change of the true and model log densities:*

$$\mathcal{L}_{\mathrm{stNCE}}(\theta) = 2\log 2 + \frac{\Delta t^2}{4}\, \mathbb{E}_{p_t}\left[\|\mathrm{d}_t \log p_t(x) - \mathrm{d}_t \log p_t^\theta(x)\|^2\right] + o(\Delta t^2). \tag{30}$$

where $\mathrm{d}_t \log p_t(x) := \partial_t \log p_t(x) + \nabla_x \log p_t(x) \cdot v_t(x)$.

6 Experiments

We empirically evaluate our framework for learning densities of synthetic data, images and molecules. In particular, we report results of the different variants of our Spatiotemporal Noise-Contrastive Estimation (stNCE-m, stNCE-w, stNCE-s, stNCE-o) that are detailed in Section 5. We

provide a compact glossary of the different algorithms in Table 4. We may also combine tNCE or stNCE-s with a loss function on the space score estimates induced by the energy network, in which case we explicitly indicate it by writing +DSM; see Appendix E.2. These space score estimates can be reused for stNCE-s loss. After the models are trained, we perform an approximate normalization step following Guth et al. [34]; see Appendix D.5. Our implementation of stNCE-m naively adds up tNCE loss and tCNCE loss, leading to slower running time, as indicated using *. Across datasets, our stNCE-s and variants are most promising: they are competitive with the state of the art.

6.1 Failure modes

Many methods for learning energy-based models, including denoising score matching used in diffusion models [75; 23], are typically evaluated on complex datasets. Yet, Figure 1 shows that they can already fail in a simple one-dimensional setting. This experiment highlights two key properties of the data distribution that drive these failures. First, multimodality increases the error of methods based on learning spatial differences. Second, a mismatch between the high-density regions ("supports") of the data and reference distributions increases the error of methods based on learning temporal differences. Methods based on learning spatiotemporal difference are robust to both sources of error. Details on the simulations are provided in Appendix E.4.

6.2 Gaussian mixtures

Image data are high-dimensional and sparse. To emulate these properties, we build a Gaussian mixture from the MNIST dataset [50] by sampling 100 images (10 per digit class) and using them as centroids with diagonal covariance $0.01I$. This synthetic yet realistic distribution exhibits the two pathologies identified in Section 3: strong multimodality and minimal overlap with a standard Gaussian reference distribution due to its thin support. We therefore expect purely spatial and purely temporal methods to struggle, while spatiotemporal methods with density-aware kernels should perform well. We use LogMSE, LogRatio, LogNormMSE and NormNLL as evaluation metrics and benchmark different EBM training algorithms, measuring the quality of the learned energies, either with or without using a separately estimated normalizing constant using importance sampling, where always lower is better. Further details on the evaluation metrics are provided in Appendix E.6.2.

The results are shown in Table 1. stNCE-w results match the intuition of Figure 3: finite Gaussian perturbations perform poorly because they push samples off the data manifold, whereas infinitesimal perturbations (Dual SM [34]) preserve proximity to the manifold and yield accurate estimates, albeit at higher computational cost due to double differentiation through the network. Results for the forward–reverse kernel further support this interpretation. Even with finite corruption, stNCE-s remains competitive, possibly because it stays close to the data manifold. By contrast, as discussed in Section 3, RNE [37] is consistent only when given access to the true scores $\nabla_{x_t} \log p_t(x_t)$ (RNE-o), which are unavailable in practice; using estimated scores (RNE-s) can lead to substantial degradation. Overall, stNCE-s provides a favorable trade-off between accuracy and speed, and is the only NCE-based method that successfully learns this challenging Gaussian mixture.

We additionally perform a fine-grained analysis to validate that the error analysis of different methods in Figure 1 and Figure 2 holds for the trained models. As presented in Appendix F.2.1, our unifying heuristics as discussed in Section 3 provides a reasonable prediction for the performance of the different methods, where the estimated density ratios have higher errors in low density region.

6.3 Images

In this section, we follow previous related works [34; 85] and report the Negative Log-Likelihood (NLL) in bits per dimension, defined as $\mathrm{NLL}_{\mathrm{bpd}} := \frac{-\mathbb{E}_{x\sim p_d(\cdot|1)}[\log p_\theta(x|1)]}{D\log 2}$, where D is the data dimension (the number of pixels times channels). Here the DSM loss is a conditional flow matching loss on the velocity estimates recovered from energy network; see Appendix E.2 for details.

MNIST We next train the different algorithms to learn the data density of images from the MNIST dataset of handwritten digits [50]. Results are shown in Table 11: our stNCE-s is the best among NCE algorithms while being fast to train. In this table the algorithms were trained for $100,000$ steps; we verify in Table 12 that stNCE-s continues to improve when optimized longer. Our stNCE-

Table 1: Results on MNIST mixture (784 dim). Time means average time per step in seconds. For methods not in Oracle category best results are in **bold**, second ones are underlined. Under oracle category best results are in **bold**.

Category	Method	LogMSE ↓	LogRatio ↓	LogNormMSE ↓	NormNLL ↓	Time ↓
Temporal	tNCE	12.37 ± 0.02	6.85 ± 0.14	8.86 ± 0.19	-606.76 ± 7.83	0.39
	CTSM	12.44 ± 0.45	7.14 ± 0.41	8.96 ± 0.24	-602.92 ± 9.59	0.68
Spatial	tCNCE	12.28 ± 0.26	7.9 ± 0.22	9.13 ± 0.18	-598.73 ± 7.82	0.4
	DSM	16.39 ± 0.07	9.31 ± 0.43	10.24 ± 0.39	-535.54 ± 28.81	0.7
Spatial-temporal	stNCE-m	10.31 ± 0.12	6.29 ± 0.45	7.64 ± 0.42	-644.8 ± 8.51	0.45*
	stNCE-w	9.96 ± 0.05	5.96 ± 0.33	7.27 ± 0.29	-652.43 ± 5.22	0.4
	stNCE-s	4.8 ± 0.1	3.34 ± 0.07	4.17 ± 0.12	-680.99 ± 0.53	**0.36**
	RNE-s	14.86 ± 0.31	7.75 ± 0.39	9.62 ± 0.05	-570.51 ± 1.09	0.44
	Dual SM	**3.73 ± 0.18**	**2.88 ± 0.06**	**3.25 ± 0.11**	**-683.96 ± 0.33**	0.7
Oracle	stNCE-o	4.99 ± 0.11	3.69 ± 0.06	4.95 ± 0.03	-677.07 ± 0.12	**0.32**
	RNE-o	**2.92 ± 0.08**	**2.06 ± 0.02**	**2.24 ± 0.07**	**-685.75 ± 0.09**	0.4

s+DSM reaches results competitive with strong baselines, by jointly training the spatial scores of the density. We additionally verify the sampling capabilities of the trained model; see Appendix F.3.1.

Table 2: Negative log likelihood (in bits/dimension) on ImageNet64 test set as in Guth et al. [34].

Method	Anti-aliasing	Horizontal flips	Discreteness	Type	Single NFE	NLL
Glow [47]	✗	✗	Continuous	Normalized	✓	3.81
PixelCNN [78]	✗	✗	Discrete	Normalized	✗	3.57
I-DDPM [58]	✗	✗	Continuous	Upper bound	✗	3.54
VDM [45]	✗	✗	Discrete	Upper bound	✗	3.40
FM [53]	✓	✗	Uniform*	Normalized	✗	3.31
NFDM [5]	✗	✓	Uniform*	Normalized	✗	3.20
TarFlow [86]	✗	✓	Uniform	Normalized	✓	2.99
Dual SM [34]	✓	✓	Continuous	Estimate	✓	3.36
Ours (stNCE-s+DSM)	✓	✓	Continuous	Estimate	✓	2.94



Figure 4: ImageNet test set images with the highest densities (left) and the lowest densities (right) as identified using our model. Higher likelihood images appear to have higher frequencies manifested by sharper contours and shapes, while lower likelihood images appear to have lower frequencies in the form of regular geometric patterns. This is consistent with the literature [34].

ImageNet As a comparison against the state-of-the-art in larger scale, we repeat the experiment on ImageNet64 [21] using our strongest method as identified based on the previous results, stNCE-s+DSM. In Table 2, we can see that our stNCE-s+DSM is competitive with Flow Matching [53] and outperforms approaches such Dual Score-Matching [34], which similarly models the data density as an energy-based model. We verify in Figure 4 that our learnt model assigns high probability to sharper images and lower probability to blurrier images.

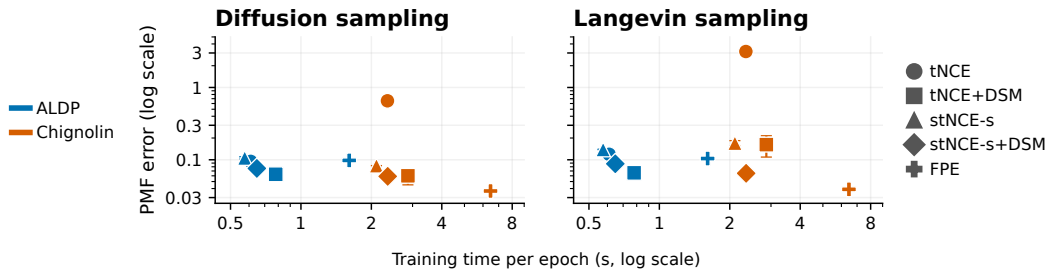


Figure 5: Diffusion and Langevin PMF versus training time. On ALDP dataset, both tNCE and stNCE-s yields strong performance compared to FPE. On the higher dimensional Chignolin dataset, stNCE-s yields better performance than tNCE, either with or without DSM. Notably, stNCE-s+DSM provides comparable accuracy at roughly half the training cost.

6.4 Molecules

We wish to model a target distribution (*a.k.a.* equilibrium distribution) over configurations of a molecule using an EBM parameterized with a neural network. Each data point in that space is the collection of three-dimensional positions of the atoms, or rather groups of atoms called chains, that make up the molecule. We consider two "coarse-grained" molecular systems: Alanine dipeptide (ALDP) and Chignolin. ALDP is represented by 5 chains and Chignolin by 10 chains; because each chain has three-dimensional spatial coordinates, this leads to 15-dimensional and 30-dimensional data, respectively. We use the datasets provided by Plainer et al. [62] and Lindorff-Larsen et al. [52], largely adapting the same neural network and experimental settings as Plainer et al. [62].

We compare tNCE and stNCE-s, possibly adding DSM loss, against another spatiotemporal method from Plainer et al. [62] based on the Fokker-Planck Equation (FPE). FPE consists in learning *spatial* differences of the energy using DSM, and *temporal* differences of the energy by minimizing a PINN loss that effectively forces the energy to follow a certain Partial Differential Equation. Although technically sound, FPE is computationally expensive due to multiple additional network evaluations. For evaluation, we compare ground-truth samples $x \sim p_d(x|1)$ from the the dataset and samples from our model $x \sim p_\theta(x|1)$ obtained through two score-based samplers, the Langevin sampler which uses the final score $\nabla \log p_\theta(x|1)$ [26] and the diffusion sampler which uses the intermediate scores $\nabla \log p_\theta(x|t)$ [71]. We then project the ground-truth and model samples onto the 2-dimensional subspace, and estimate the densities of the empirical samples to compute the Potential of Mean Force (PMF) errors. We additionally estimate the running times of the different methods.

The results are reported in Figure 5. Jensen-Shannon divergence, another metric employed by Plainer et al. [62], largely follows the same trend as PMF; we report detailed results in Appendix F.4.1. In general, training jointly with DSM consistently improves performance while introducing limited computational overhead possibly due to the smaller scale of these problems. On ALDP, both tNCE+DSM and stNCE-s+DSM achieve better results than FPE. On Chignolin, they achieve comparable accuracies when compared against FPE at roughly half the training cost.

7 Discussion

In conclusion, we show that despite their diversity, most energy-based model estimation methods fall into three categories: spatial, temporal, and spatiotemporal. We identify fundamental failure modes of purely spatial and temporal approaches, and show how spatiotemporal methods overcome them. Our unified binary classification framework recovers many existing methods and yields new objectives with near state-of-the-art performance across synthetic, image, and molecular data.

Limitations and future directions While models trained using NCE loss functions can accurately recover the energies, the scores may not be learned well compared to the score matching variants. Future work could explore better samplers for models trained with NCE losses that either do not rely on gradient information or could better handle noisy gradients. The perturbation kernel is an important design choice but is currently left as heuristics. The strongest results typically rely on a separate DSM loss, and the best stand-alone variant does not reach those numbers on its own.

Acknowledgments and Disclosure of Funding

The authors wish to acknowledge Florentin Guth for interesting discussions on Dual Score Matching [34]. Hanlin Yu and Arto Klami acknowledge the research environment provided by ELLIS Institute Finland, and were supported by the Research Council of Finland Flagship programme: Finnish Center for Artificial Intelligence FCAI. RuiKang OuYang acknowledges the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L016516/1 for the University of Cambridge Centre for Doctoral Training, the Cambridge Centre for Analysis. Michael U. Gutmann was supported in part by Gen AI - the AI Hub for Generative Models, funded by EPSRC (EP/Y028805/1). The authors wish to acknowledge CSC - IT Center for Science, Finland, for computational resources, and acknowledge CSC - IT Center for Science, Finland for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC(Finland) and the LUMI consortium through CSC - IT Center for Science, Finland. The authors declare no competing interests.

References

[1] R. Aggarwal, J. Chen, N. M. Boffi, and D. Koes. BoltzNCE: Learning likelihoods for boltzmann generation with stochastic interpolants and noise contrastive estimation. In *Conference on neural information processing systems*, 2025.

[2] X. Ai, Y. He, A. Gu, R. Salakhutdinov, J. Kolter, N. Boffi, and M. Simchowitz. Joint distillation for fast likelihood evaluation and sampling in flow-based models. In *International Conference on Learning Representations*, 2026.

[3] M. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

[4] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *ACM international conference on architectural support for programming languages and operating systems, volume 2 (ASPLOS '24)*. ACM, Apr. 2024.

[5] G. Bartosh, D. Vetrov, and C. A. Naesseth. Neural flow diffusion models: Learnable forward process for improved diffusion modelling. In *Annual Conference on Neural Information Processing Systems*, 2024.

[6] I. Batatia, D. P. Kovacs, G. Simm, C. Ortner, and G. Csanyi. MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. In *Advances in Neural Information Processing Systems*, volume 35, pages 11423–11436. Curran Associates, Inc., 2022.

[7] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, volume 97, pages 573–582. PMLR, 09–15 Jun 2019.

[8] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[9] L. Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`.

[10] N. Boffi, M. Albergo, and E. Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 33346–33382. Curran Associates, Inc., 2025.

[11] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, Y. Katariya, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

[12] C. Ceylan. Conditional noise-contrastive estimation: With application to natural image statistics. Master's thesis, KTH, School of Computer Science and Communication (CSC), Stockholm, Sweden, 2017. Independent thesis, advanced level (20 credits / 30 HE credits).

[13] C. Ceylan and M. U. Gutmann. Conditional Noise-Contrastive Estimation of Unnormalised Models. In *International Conference on Machine Learning*, volume 80, pages 726–734. PMLR, July 2018.

[14] O. Chehab, A. Hyvarinen, and A. Risteski. Provable benefits of annealing for estimating normalizing constants: Importance sampling, noise-contrastive estimation, and beyond. In *Conference on Neural Information Processing Systems*, 2023.

[15] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[16] W. Chen, S. Li, J. Li, J. Xu, Z. Lin, J. Yang, D. Zeng, J. Paisley, and Q. Zhao. Diffusion Secant Alignment for Score-Based Density Ratio Estimation, 2025. arXiv: 2509.04852 [stat.ML].

[17] W. Chen, S. Li, J. Li, J. Yang, J. Paisley, and D. Zeng. Dequantified diffusion-schrödinger bridge for density ratio estimation. In *International Conference on Machine Learning*, volume 267, pages 8427–8452. PMLR, 13–19 Jul 2025.

[18] W. Chen, J. Li, S. Li, Z. Lin, J. Yang, J. Paisley, and D. Zeng. A minimum variance path principle for accurate and stable score-based density ratio estimation. In *International Conference on Learning Representations*, 2026.

[19] S. Chewi, A. Kalavasis, A. Mehrotra, and O. Montasser. DDPM score matching is asymptotically efficient. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025.

[20] K. Choi, C. Meng, Y. Song, and S. Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, volume 151, pages 2552–2573. PMLR, 28–30 Mar 2022.

[21] P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

[22] A. Doucet, W. Grathwohl, A. G. Matthews, and H. Strathmann. Score-based diffusion meets annealed importance sampling. In *Advances in Neural Information Processing Systems*, volume 35, pages 21482–21494. Curran Associates, Inc., 2022.

[23] Y. Du and I. Mordatch. Implicit generation and modeling with energy-based models. In *international conference on neural information processing systems*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[24] Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. S. Grathwohl. Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC. In *International Conference on Machine Learning*, volume 202, pages 8489–8510. PMLR, July 2023.

[25] A. E. P. Durumeric, Y. Chen, F. Noé, and C. Clementi. Learning data efficient coarse-grained molecular dynamics from forces and noise, 2024. arXiv: 2407.01286 [physics.bio-ph].

[26] P. Eastman, R. Galvelis, R. P. Peláez, C. R. A. Abreu, S. E. Farr, E. Gallicchio, A. Gorenko, M. M. Henry, F. Hu, J. Huang, A. Krämer, J. Michel, J. A. Mitchell, V. S. Pande, J. P. Rodrigues, J. Rodriguez-Guerra, A. C. Simmonett, S. Singh, J. Swails, P. Turner, Y. Wang, I. Zhang, J. D. Chodera, G. De Fabritiis, and T. E. Markland. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *The Journal of Physical Chemistry B*, 128(1):109–116, Jan. 2024. ISSN 1520-6106.

[27] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. Flow contrastive estimation of energy-based models. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 7515–7525, 2020.

[28] R. Gao, Y. Song, B. Poole, Y. N. Wu, and D. P. Kingma. Learning energy-based models by diffusion recovery likelihood. In *International conference on learning representations*, 2021.

[29] M. Gonzalez, N. Fernandez Pinto, T. Tran, E. Gherbi, H. Hajri, and N. Masmoudi. SEEDS: Exponential SDE solvers for fast high-quality sampling from diffusion models. In *Advances in neural information processing systems*, volume 36, pages 68061–68120. Curran Associates, Inc., 2023.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *Commun. ACM*, 63(11):139–144, Oct. 2020. ISSN 0001-0782. Place: New York, NY, USA.

[31] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *International Conference on Learning Representations*, 2019.

[32] L. Grenioux, E. Moulines, and M. Gabrié. Balanced training of energy-based models with adaptive flow sampling. In *ICML 2023 workshop on structured probabilistic inference & generative modeling*, 2023.

[33] S. Gugger, L. Debut, T. Wolf, P. Schmid, Z. Mueller, S. Mangrulkar, M. Sun, and B. Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable., 2022. URL https://github.com/huggingface/accelerate.

[34] F. Guth, Z. Kadkhodaie, and E. Simoncelli. Learning normalized image densities via dual score matching. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 89796–89826. Curran Associates, Inc., 2025.

[35] M. U. Gutmann and J.-i. Hirayama. Bregman divergence as general framework to estimate unnormalized statistical models. In *conference on uncertainty in artificial intelligence*, UAI'11, pages 283–290, Barcelona, Spain, 2011. AUAI Press. ISBN 978-0-9749039-7-2.

[36] M. U. Gutmann and A. Hyvärinen. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012.

[37] J. He, J. M. Hernández-Lobato, Y. Du, and F. Vargas. RNE: plug-and-play diffusion inference-time control and energy-based training. In *International Conference on Learning Representations*, 2026.

[38] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[39] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[40] P. Holderrieth, U. Singer, T. Jaakkola, R. T. Q. Chen, Y. Lipman, and B. Karrer. GLASS flows: Efficient inference for reward alignment of flow and diffusion models. In *International conference on learning representations*, 2026.

[41] B. E. Husic, N. E. Charron, D. Lemm, J. Wang, A. Pérez, M. Majewski, A. Krämer, Y. Chen, S. Olsson, G. de Fabritiis, F. Noé, and C. Clementi. Coarse graining molecular dynamics with graph neural networks. *The Journal of Chemical Physics*, 153(19), Nov. 2020. ISSN 1089-7690.

[42] A. Hyvarinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on Neural Networks*, 18(5):1529–1531, 2007.

[43] A. Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[44] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 26565–26577. Curran Associates, Inc., 2022.

[45] D. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707. Curran Associates, Inc., 2021.

[46] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[47] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[48] F. Koehler, A. Heckett, and A. Risteski. Statistical efficiency of score matching: The view from isoperimetry. In *The Eleventh International Conference on Learning Representations*, 2023.

[49] C.-H. Lai, Y. Takida, N. Murata, T. Uesaka, Y. Mitsufuji, and S. Ermon. FP-diffusion: Improving score-based diffusion models by enforcing the underlying score fokker-planck equation. In *International Conference on Machine Learning*, volume 202, pages 18365–18398. PMLR, 23–29 Jul 2023.

[50] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[51] T. Li and K. He. Back to Basics: Let Denoising Generative Models Denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 36115–36125, June 2026.

[52] K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

[53] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[54] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Q. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat. Flow matching guide and code, 2024. arXiv: 2412.06264 [cs.LG].

[55] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International conference on learning representations*, 2019.

[56] C. Lu, Y. Zhou, F. Bao, J. Chen, C. LI, and J. Zhu. DPM-solver: a fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in neural information processing systems*, volume 35, pages 5775–5787. Curran Associates, Inc., 2022.

[57] B. Máté, S. Klein, T. Golling, and F. Fleuret. Flowification: Everything is a normalizing flow. In *Advances in Neural Information Processing Systems*, 2022.

[58] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, volume 139, pages 8162–8171. PMLR, 18–24 Jul 2021.

[59] F. Noé, S. Olsson, J. Köhler, and H. Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[60] R. Ouyang, L. Grenioux, and J. M. Hernández-Lobato. A Diffusive Classification Loss for Learning Energy-based Generative Models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 95197–95252. PMLR, July 2026.

[61] R. OuYang, H. Yu, X. Ai, Y. He, N. M. Boffi, P. Ravikumar, J. M. Hernández-Lobato, M. Simchowitz, B. K. Miller, and O. Chehab. Few-Step Boltzmann Generators via Scalable Likelihood Flow Maps. In *ICML 2026 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2026.

[62] M. Plainer, H. Wu, L. Klein, S. Günnemann, and F. Noe. Consistent Sampling and Simulation: Molecular Dynamics with Energy-Based Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 24460–24505. Curran Associates, Inc., 2025.

[63] Y. Qin and A. Risteski. Fit like you sample: Sample-efficient generalized score matching from fast mixing diffusions. In *Conference on Learning Theory*, volume 247, pages 4413–4457. PMLR, 30 Jun–03 Jul 2024.

[64] B. Rhodes, K. Xu, and M. U. Gutmann. Telescoping density-ratio estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 4905–4916. Curran Associates, Inc., 2020.

[65] M. Skreta, T. Akhound-Sadegh, V. Ohanesian, R. Bondesan, A. Aspuru-Guzik, A. Doucet, R. Brekelmans, A. Tong, and K. Neklyudov. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. In *International Conference on Machine Learning*, 2025.

[66] Y. Song and S. Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[67] Y. Song and D. P. Kingma. How to train your energy-based models, 2021. arXiv: 2101.03288 [cs.LG].

[68] Y. Song, C. Meng, and S. Ermon. Mintnet: Building invertible neural networks with masked convolutions. In *Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[69] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In *Uncertainty in Artificial Intelligence Conference*, volume 115, pages 574–584. PMLR, July 2020.

[70] Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum Likelihood Training of Score-Based Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 34, pages 1415–1428. Curran Associates, Inc., 2021.

[71] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021.

[72] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency Models. In *International Conference on Machine Learning*, ICML'23. JMLR.org, 2023. Place: Honolulu, Hawaii, USA.

[73] A. Srivastava, S. Han, K. Xu, B. Rhodes, and M. U. Gutmann. Estimating the Density Ratio between Distributions with High Discrepancy using Multinomial Logistic Regression. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[74] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[75] J. Thornton, L. Bethune, R. Zhang, A. Bradley, P. Nakkiran, and S. Zhai. Composition and control with distilled energy diffusion models and sequential monte carlo. In *AISTATS 2025*, 2025.

[76] A. Y. Tong, N. Malkin, K. Fatras, L. Atanackovic, Y. Zhang, G. Huguet, G. Wolf, and Y. Bengio. Simulation-Free Schrödinger Bridges via Score and Flow Matching. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 1279–1287. PMLR, May 2024.

[77] M. Uehara, T. Matsuda, and F. Komaki. Analysis of Noise Contrastive Estimation from the Perspective of Asymptotic Variance, 2018. arXiv: 1808.07983 [stat.ML].

[78] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[79] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, UK, 1998.

[80] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

[81] P. Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[82] L. K. Wenliang and H. Kanagawa. Blindness of score-based methods to isolated components and mixing proportions, 2021. _eprint: 2008.10087.

[83] O. Yair and T. Michaeli. Contrastive Divergence Learning is a Time Reversal Adversarial Game. In *International Conference on Learning Representations*, 2021.

[84] H. Yu, M. U. Gutmann, A. Klami, and O. Chehab. Conditional noise-contrastive estimation of energy-based models by jumping between modes. In *EurIPS 2025 workshop on principles of generative modeling (PriGM)*, 2025.

[85] H. Yu, A. Klami, A. Hyvarinen, A. Korba, and O. Chehab. Density Ratio Estimation with Conditional Probability Paths. In *Forty-second International Conference on Machine Learning*, 2025.

[86] S. Zhai, R. Zhang, P. Nakkiran, D. Berthelot, J. Gu, H. Zheng, T. Chen, M. Á. Bautista, N. Jaitly, and J. M. Susskind. Normalizing flows are capable generative models. In *International Conference on Machine Learning*, 2025.

Appendix

The appendix is organized as follows.

A Theoretical results **19**

A.1 Useful lemma on the logistic loss . . . 19

A.2 Non-parametric consistency (Theorem 1) . . . 20

A.3 Forward-reverse kernel: Infinitesimal Limit (Propositions 3 and Proposition 4) . . . 20

A.4 White noise kernel: infinitesimal limit . . . 22

B Special cases of our method **25**

B.1 Summary . . . 25

B.2 Noise-Contrastive Estimation (NCE) . . . 25

B.3 Flow-Contrastive Estimation (FCE) . . . 26

B.4 Temporal Noise-Contrastive Estimation (tNCE) . . . 27

B.5 Conditional Noise-Contrastive Estimation (CNCE) . . . 27

B.6 Contrastive Divergence (CD) . . . 28

B.7 Temporal Conditional Noise-Contrastive Estimation (tCNCE) . . . 28

B.8 stNCE-m and sum of tNCE and tCNCE . . . 29

B.9 Time Score Matching . . . 30

B.10 Space Score Matching . . . 30

B.11 Energy-based Diffusion . . . 31

C Additional discussion **32**

C.1 Compute burden of the likelihood: training-time *versus* inference-time . . . 32

C.2 Discussion on related spatiotemporal methods . . . 32

C.3 Discussion on RNE . . . 32

C.4 Comparison against Dual Score Matching . . . 33

C.5 Broader impacts . . . 33

D Implementation considerations **34**

D.1 Sampling mechanisms . . . 34

D.2 Implementing forward-reverse kernel . . . 36

D.3 Parameterization of energy-based diffusion models . . . 37

D.4 Score and velocity . . . 39

D.5 Obtaining approximately normalized energies . . . 40

E Experimental details **41**

E.1 Glossary of methods . . . 41

E.2 Losses on the scores . . . 41

E.3 Network architecture . . . 42

E.4 Failure modes . . . 43

E.5 Toy mixture 44
E.6 MNIST mixture 44
E.7 Images 45
E.8 Molecules 45
E.9 Compute resources 46
E.10 License information 46

F Additional experimental results **47**
F.1 Synthetic Gaussian mixtures 47
F.2 MNIST mixture 47
F.3 Images 49
F.4 Molecules 51

A Theoretical results

In this section, we prove the consistency of the estimator obtained by minimizing the empirical stNCE loss. We also derive the simplified expressions of stNCE loss under specific perturbation kernels.

A.1 Useful lemma on the logistic loss

The following are useful lemma on how to rewrite the logistic classification loss.

Lemma 5 (Logistic loss as a Bregman divergence) *Consider the logistic loss*

$$\mathcal{L}_{\text{logistic}}(F) = -\mathbb{E}_{x\sim p_A}\left[\log \text{sigmoid}(F(x))\right] - \mathbb{E}_{x\sim p_B}\left[\log(1-\text{sigmoid}(F(x)))\right], \tag{31}$$

which is minimized by

$$F^*(x) = \log p_A(x) - \log p_B(x). \tag{32}$$

Then, up to an additive constant independent of F, the logistic loss can be written as a Bregman divergence between e^{F^*} and e^F.

Proof. This is inspired by Gutmann and Hirayama [35]; Uehara et al. [77].

Let $\Phi(u) = u\log u - (1+u)\log(1+u)$. Consider the pointwise Bregman divergence

$$D_\Phi\big(e^{F^*(x)}, e^{F(x)}\big) = \Phi\big(e^{F^*(x)}\big) - \Phi\big(e^{F(x)}\big) - \Phi'\big(e^{F(x)}\big)\big(e^{F^*(x)} - e^{F(x)}\big). \tag{33}$$

Integrating against $p_B(x)\,dx$ yields a divergence between functions:

$$D_\Phi\big(e^{F^*}, e^F\big) = \int D_\Phi\big(e^{F^*(x)}, e^{F(x)}\big)\, p_B(x)\,dx. \tag{34}$$

Substituting $F^*(x) = \log p_A(x) - \log p_B(x)$, a straightforward calculation shows

$$D_\Phi\big(e^{F^*}, e^F\big) = \mathcal{L}_{\text{logistic}}(F) - \mathcal{L}_{\text{logistic}}(F^*), \tag{35}$$

which proves the claim. □

Lemma 6 (Logistic loss simplified when the model is near zero) *Consider the logistic loss*

$$\mathcal{L}(F_\theta) := -\mathbb{E}_{p^+(x)}\big[\log \text{sigmoid}(F_\theta(x))\big] - \mathbb{E}_{p^-(x)}\big[\log\big(1-\text{sigmoid}(F_\theta(x))\big)\big]. \tag{36}$$

If $F_\theta(x)$ is close to zero, then

$$\mathcal{L}(F_\theta) = \mathcal{L}(0) - \frac{1}{2}\mathbb{E}_{p^+(x)-p^-(x)}\left[F_\theta(x)\right] + \frac{1}{8}\mathbb{E}_{p^+(x)+p^-(x)}\left[F_\theta(x)^2\right] \tag{37}$$

$$+ \mathbb{E}_{p^+(x)+p^-(x)}\left[o(F_\theta(x)^2)\right]. \tag{38}$$

Proof. For fixed x, define the pointwise loss

$$\ell(F_\theta) := -p^+\log \text{sigmoid}(F_\theta) - p^-\log\big(1-\text{sigmoid}(F_\theta)\big). \tag{39}$$

A second-order Taylor expansion around $F_\theta = 0$ gives

$$\ell(F_\theta) = \ell(0) - \frac{1}{2}\big(p^+ - p^-\big)F_\theta + \frac{1}{8}\big(p^+ + p^-\big)F_\theta^2 + \big(p^+ + p^-\big)o(F_\theta^2). \tag{40}$$

Integrating over x yields the result. □

A.2 Non-parametric consistency (Theorem 1)

Recall that the stNCE population loss is the logistic classification objective

$$\mathcal{L}_{\text{stNCE}}(F) = -\mathbb{E}_{x\sim p_A}\big[\log \text{sigmoid}(F(x))\big] - \mathbb{E}_{x\sim p_B}\big[\log(1-\text{sigmoid}(F(x)))\big]. \tag{41}$$

By Lemma 5, this can be written (up to an additive constant) as

$$\mathcal{L}_{\text{stNCE}}(F) = D_\Phi\big(e^{F^*}, e^F\big) + \text{const.} \tag{42}$$

Hence any minimizer $F^{\theta_{\min}}$ satisfies

$$F^{\theta_{\min}} = F^*. \tag{43}$$

Using the definition of the stNCE classifier Eq. 17, this yields, for all (x,t) and (x',t') in the support,

$$\log p_{\theta_{\min}}(x \mid t) - \log p_{\theta_{\min}}(x' \mid t') = \log p_d(x \mid t) - \log p_d(x' \mid t'). \tag{44}$$

The equality is well-defined whenever the classification mixture has positive density.

Since the perturbation kernel has full support, applying Equation 44 along these transitions shows that

$$\log p_{\theta_{\min}}(x \mid t) = \log p_d(x \mid t) + C, \tag{45}$$

where C does not depend on (x,t). Enforcing the constraint $p_\theta(x \mid t=0) = p_0(x)$ fixes C, yielding

$$p_{\theta_{\min}}(x \mid t) = p_d(x \mid t). \tag{46}$$

This establishes Fisher consistency of the population objective.

For the empirical estimator, under standard regularity conditions, empirical consistency follows from standard M-estimation theory [79, Theorem 5.7]. If the model is identifiable, we further obtain parametric Fisher consistency, i.e., $\theta_{\min} = \theta^*$.

A.3 Forward-reverse kernel: Infinitesimal Limit (Propositions 3 and Proposition 4)

Simplifying the loss around zero Let

$$p_+(x,t,x',t') := p_d(x,t)p_n(x',t' \mid x,t), \tag{47}$$

$$p_-(x,t,x',t') := p_d(x',t')p_n(x,t \mid x',t'). \tag{48}$$

The logistic loss used in stNCE is

$$\mathcal{L}(F_\theta) = -\mathbb{E}_{p_+(x,t,x',t')}[\log \text{sigmoid}(F_\theta(x,t,x',t'))] \tag{49}$$

$$- \mathbb{E}_{p_-(x,t,x',t')}[\log(1-\text{sigmoid}(F_\theta(x,t,x',t')))]. \tag{50}$$

Using Lemma 6, we have, whenever F_θ is close to zero,

$$\mathcal{L}(F_\theta) = \mathcal{L}(0) - \frac{1}{2}\mathbb{E}_{p_+-p_-}\left[F_\theta\right] + \frac{1}{8}\mathbb{E}_{p_++p_-}\left[F_\theta^2\right] + \mathbb{E}_{p_++p_-}\left[o(F_\theta^2)\right]. \tag{51}$$

In contrast with the expansion around the optimum, this expansion generally contains a linear term. However, this term vanishes in the forward-reverse case considered below.

Plugging in the forward-reverse kernel Recall that the forward-reverse kernel satisfies

$$p_d(x,t)p_n(x',t' \mid x,t) = p_d(x',t')p_n(x,t \mid x',t'), \tag{52}$$

due to the Bayes rule. Equivalently,

$$p_+(x,t,x',t') = p_-(x,t,x',t'). \tag{53}$$

Therefore, the linear term vanishes:

$$\mathbb{E}_{p_+-p_-}\left[F_\theta\right] = 0. \tag{54}$$

Moreover, by taking logs in the forward-reverse identity,

$$\log p_n(x',t' \mid x,t) - \log p_n(x,t \mid x',t') = \log p_d(x',t') - \log p_d(x,t). \tag{55}$$

Hence

$$F_\theta(x,t,x',t') = \log p_\theta(x,t) - \log p_\theta(x',t') \tag{56}$$
$$+ \log p_n(x',t' \mid x,t) - \log p_n(x,t \mid x',t') \tag{57}$$
$$= \log p_\theta(x,t) - \log p_\theta(x',t') + \log p_d(x',t') - \log p_d(x,t) \tag{58}$$
$$= \big(\log p_d(x',t') - \log p_\theta(x',t')\big) - \big(\log p_d(x,t) - \log p_\theta(x,t)\big). \tag{59}$$

Defining

$$e_\theta(x,t) := \log p_d(x,t) - \log p_\theta(x,t), \tag{60}$$

we obtain

$$F_\theta(x,t,x',t') = e_\theta(x',t') - e_\theta(x,t). \tag{61}$$

Since $p_+ = p_- = p_d(x,t)p_n(x',t' \mid x,t)$, the expansion becomes

$$\mathcal{L}(F_\theta) = \mathcal{L}(0) + \frac{1}{4}\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\big(e_\theta(x',t') - e_\theta(x,t)\big)^2\right] \tag{62}$$
$$+ \mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[o\Big(\big(e_\theta(x',t') - e_\theta(x,t)\big)^2\Big)\right]. \tag{63}$$

This expansion is valid when F_θ is close to zero. In the infinitesimal-time regime, this follows from the fact that (x',t') is close to (x,t).

Case of the SDE Assume that x_t follows the SDE

$$\mathrm{d}x_t = b(x_t,t)\mathrm{d}t + g(t)\mathrm{d}W_t. \tag{64}$$

Let $t' = t + \Delta t$. Conditional on $(x_t,t) = (x,t)$, the next point satisfies

$$x' = x + b(x,t)\Delta t + g(t)\Delta W + R_t, \qquad \Delta W \sim \mathcal{N}(0, \Delta t\, I), \tag{65}$$

where

$$R_t = \int_t^{t+\Delta t} \big(b(x_s,s) - b(x,t)\big)\, ds + \int_t^{t+\Delta t} \big(g(s) - g(t)\big)\, dW_s. \tag{66}$$

Therefore, the expectation with respect to the transition kernel $p_n(x',t' \mid x,t)$ may be written as an expectation with respect to the Brownian increment ΔW.

By Itô's lemma,

$$e_\theta(x',t') - e_\theta(x,t) = \partial_t e_\theta(x,t)\Delta t + \nabla_x e_\theta(x,t)^\top b(x,t)\Delta t \tag{67}$$
$$+ g(t)\nabla_x e_\theta(x,t)^\top \Delta W + \frac{1}{2}g(t)^2 \operatorname{Tr}\left(\nabla_x^2 e_\theta(x,t)\right)\Delta t + o(\Delta t). \tag{68}$$

Since $\Delta W = O(\sqrt{\Delta t})$, the leading term is

$$e_\theta(x',t') - e_\theta(x,t) = g(t)\nabla_x e_\theta(x,t)^\top \Delta W + O(\Delta t). \tag{69}$$

Thus, conditionally on (x,t),

$$\mathbb{E}_{\Delta W}\left[\big(e_\theta(x',t') - e_\theta(x,t)\big)^2 \mid x,t\right] = g(t)^2 \mathbb{E}_{\Delta W}\left[\big(\nabla_x e_\theta(x,t)^\top \Delta W\big)^2\right] + o(\Delta t) \tag{70}$$
$$= g(t)^2 \left\|\nabla_x e_\theta(x,t)\right\|^2 \Delta t + o(\Delta t). \tag{71}$$

Substituting into the loss expansion gives

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{1}{4}\mathbb{E}_{p_d(x,t)}\mathbb{E}_{\Delta W}\left[\big(e_\theta(x',t') - e_\theta(x,t)\big)^2 \mid x,t\right] + o(\Delta t) \tag{72}$$
$$= \frac{\Delta t}{4}\mathbb{E}_{p_d(x,t)}\left[g(t)^2 \left\|\nabla_x e_\theta(x,t)\right\|^2\right] + o(\Delta t). \tag{73}$$

Finally, since

$$\nabla_x e_\theta(x,t) = \nabla_x \log p_d(x,t) - \nabla_x \log p_\theta(x,t), \tag{74}$$

we obtain

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{\Delta t}{4}\mathbb{E}_{p_d(x,t)}\left[g(t)^2 \left\|\nabla_x \log p_d(x,t) - \nabla_x \log p_\theta(x,t)\right\|^2\right] + o(\Delta t). \tag{75}$$

Case of the ODE Assume now that x_t follows the ODE

$$\mathrm{d}x_t = b(x_t, t)\mathrm{d}t. \tag{76}$$

Let $t' = t + \Delta t$. Then, conditional on $(x_t, t) = (x, t)$,

$$x' = x + b(x, t)\Delta t + o(\Delta t). \tag{77}$$

Therefore,

$$e_\theta(x', t') - e_\theta(x, t) = \left(\partial_t e_\theta(x, t) + b(x, t)^\top \nabla_x e_\theta(x, t)\right) \Delta t + o(\Delta t). \tag{78}$$

Defining the total derivative

$$\mathrm{d}_t e_\theta(x, t) := \partial_t e_\theta(x, t) + b(x, t)^\top \nabla_x e_\theta(x, t), \tag{79}$$

we get

$$e_\theta(x', t') - e_\theta(x, t) = \mathrm{d}_t e_\theta(x, t)\Delta t + o(\Delta t). \tag{80}$$

Using the Taylor expansion of the logistic loss around zero, and using the forward-reverse identity $p_+ = p_-$, the linear term vanishes and

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{1}{4}\mathbb{E}_{p_d(x,t)}\left[\left(e_\theta(x', t') - e_\theta(x, t)\right)^2\right] + \mathbb{E}_{p_d(x,t)}\left[o\Big(\left(e_\theta(x', t') - e_\theta(x, t)\right)^2\Big)\right]. \tag{81}$$

Substituting the ODE expansion gives

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{\Delta t^2}{4}\mathbb{E}_{p_d(x,t)}\left[\left(\mathrm{d}_t e_\theta(x, t)\right)^2\right] + o(\Delta t^2). \tag{82}$$

Finally, since

$$e_\theta(x, t) = \log p_d(x, t) - \log p_\theta(x, t), \tag{83}$$

we have

$$\mathrm{d}_t e_\theta(x, t) = \mathrm{d}_t \log p_d(x, t) - \mathrm{d}_t \log p_\theta(x, t). \tag{84}$$

Therefore,

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{\Delta t^2}{4}\mathbb{E}_{p_d(x,t)}\left[\left(\mathrm{d}_t \log p_d(x, t) - \mathrm{d}_t \log p_\theta(x, t)\right)^2\right] + o(\Delta t^2). \tag{85}$$

We remark that the SDE interpretation is closer to the practical implementation of stNCE with forward-reverse kernel. With the ODE interpretation, one would need to know the velocity field $b(x, t)$ in advance, and need to keep track of the ground truth $\mathrm{d}_t \log p_t(x)$.

A.4 White noise kernel: infinitesimal limit

Simplifying the loss around zero Let

$$p_+(x, t, x', t') := p_d(x, t)p_n(x', t' \mid x, t), \tag{86}$$

$$p_-(x, t, x', t') := p_d(x', t')p_n(x, t \mid x', t'). \tag{87}$$

The logistic loss used in stNCE is

$$\mathcal{L}(F_\theta) = -\mathbb{E}_{p_+}\left[\log \mathrm{sigmoid}(F_\theta(x, t, x', t'))\right] - \mathbb{E}_{p_-}\left[\log\big(1 - \mathrm{sigmoid}(F_\theta(x, t, x', t'))\big)\right]. \tag{88}$$

Using Lemma 6, whenever F_θ is close to zero,

$$\mathcal{L}(F_\theta) = \mathcal{L}(0) - \frac{1}{2}\mathbb{E}_{p_+ - p_-}\left[F_\theta\right] + \frac{1}{8}\mathbb{E}_{p_+ + p_-}\left[F_\theta^2\right] + \mathbb{E}_{p_+ + p_-}\left[o(F_\theta^2)\right]. \tag{89}$$

Plugging in the white noise kernel Assume that the perturbation kernel is an infinitesimal Gaussian kernel in both space and time:

$$x' = x + \sqrt{\varepsilon_x}\,\xi, \qquad \xi \sim \mathcal{N}(0, I), \tag{90}$$

$$t' = t + \sqrt{\varepsilon_t}\,\tau, \qquad \tau \sim \mathcal{N}(0, 1), \tag{91}$$

where ξ and τ are independent. Since the Gaussian kernel is symmetric,

$$p_n(x', t' \mid x, t) = p_n(x, t \mid x', t'). \tag{92}$$

Thus

$$F_\theta(x, t, x', t') = \log p_\theta(x, t) - \log p_\theta(x', t'). \tag{93}$$

A first-order Taylor expansion gives

$$F_\theta(x, t, x', t') = -\sqrt{\varepsilon_x}\nabla_x \log p_\theta(x, t)^\top \xi - \sqrt{\varepsilon_t}\partial_t \log p_\theta(x, t)\tau + o(\sqrt{\varepsilon_x} + \sqrt{\varepsilon_t}). \tag{94}$$

Therefore F_θ is close to zero in the infinitesimal perturbation limit, justifying the Taylor expansion around zero.

The linear term Using symmetry of p_c, we have

$$p_+(x, t, x', t') - p_-(x, t, x', t') = \big(p_d(x, t) - p_d(x', t')\big)p_n(x', t' \mid x, t). \tag{95}$$

Moreover,

$$p_d(x', t') = p_d(x, t) + \sqrt{\varepsilon_x}\nabla_x p_d(x, t)^\top \xi + \sqrt{\varepsilon_t}\partial_t p_d(x, t)\tau + o(\sqrt{\varepsilon_x} + \sqrt{\varepsilon_t}). \tag{96}$$

Hence

$$p_d(x, t) - p_d(x', t') = -p_d(x, t)\left[\sqrt{\varepsilon_x}\nabla_x \log p_d(x, t)^\top \xi + \sqrt{\varepsilon_t}\partial_t \log p_d(x, t)\tau\right] \tag{97}$$

$$+ o(\sqrt{\varepsilon_x} + \sqrt{\varepsilon_t}). \tag{98}$$

Combining this with the expansion of F_θ, and averaging over ξ, τ, gives

$$\mathbb{E}_{p_+ - p_-}\left[F_\theta\right] = \mathbb{E}_{p_d(x,t)}\Big[\varepsilon_x \nabla_x \log p_d(x, t)^\top \nabla_x \log p_\theta(x, t) + \varepsilon_t \partial_t \log p_d(x, t)\partial_t \log p_\theta(x, t)\Big] \tag{99}$$

$$+ o(\varepsilon_x + \varepsilon_t), \tag{100}$$

where the mixed term vanishes because ξ and τ are independent and centered.

The quadratic term Similarly,

$$F_\theta^2 = \varepsilon_x \left(\nabla_x \log p_\theta(x, t)^\top \xi\right)^2 + \varepsilon_t \left(\partial_t \log p_\theta(x, t)\tau\right)^2 \tag{101}$$

$$+ 2\sqrt{\varepsilon_x \varepsilon_t}\left(\nabla_x \log p_\theta(x, t)^\top \xi\right)\left(\partial_t \log p_\theta(x, t)\tau\right) \tag{102}$$

$$+ o(\varepsilon_x + \varepsilon_t). \tag{103}$$

Averaging over ξ, τ yields

$$\mathbb{E}_{\xi,\tau}\left[F_\theta^2\right] = \varepsilon_x \left\|\nabla_x \log p_\theta(x, t)\right\|^2 + \varepsilon_t \left(\partial_t \log p_\theta(x, t)\right)^2 + o(\varepsilon_x + \varepsilon_t). \tag{104}$$

Since

$$p_+(x, t, x', t') + p_-(x, t, x', t') = 2p_d(x, t)p_n(x', t' \mid x, t) + o(1), \tag{105}$$

we get

$$\frac{1}{8}\mathbb{E}_{p_+ + p_-}\left[F_\theta^2\right] = \frac{1}{4}\mathbb{E}_{p_d(x,t)}\left[\varepsilon_x \left\|\nabla_x \log p_\theta(x, t)\right\|^2 + \varepsilon_t \left(\partial_t \log p_\theta(x, t)\right)^2\right] + o(\varepsilon_x + \varepsilon_t). \tag{106}$$

Combining the terms Substituting the linear and quadratic terms into the expansion gives

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{1}{4}\mathbb{E}_{p_d(x,t)}\Bigg[\varepsilon_x \left\|\nabla_x \log p_\theta(x,t)\right\|^2 + \varepsilon_t \left(\partial_t \log p_\theta(x,t)\right)^2\Bigg] \tag{107}$$

$$- \frac{1}{2}\mathbb{E}_{p_d(x,t)}\Bigg[\varepsilon_x \nabla_x \log p_d(x,t)^\top \nabla_x \log p_\theta(x,t) \tag{108}$$

$$+ \varepsilon_t \partial_t \log p_d(x,t) \partial_t \log p_\theta(x,t)\Bigg] \tag{109}$$

$$+ o(\varepsilon_x + \varepsilon_t). \tag{110}$$

Completing the square, we obtain

$$\mathcal{L}(F_\theta) - \mathcal{L}(0) = \frac{1}{4}\mathbb{E}_{p_d(x,t)}\Bigg[\varepsilon_x \left\|\nabla_x \log p_\theta(x,t) - \nabla_x \log p_d(x,t)\right\|^2 \tag{111}$$

$$+ \varepsilon_t \left(\partial_t \log p_\theta(x,t) - \partial_t \log p_d(x,t)\right)^2\Bigg] \tag{112}$$

$$- \frac{1}{4}\mathbb{E}_{p_d(x,t)}\Bigg[\varepsilon_x \left\|\nabla_x \log p_d(x,t)\right\|^2 + \varepsilon_t \left(\partial_t \log p_d(x,t)\right)^2\Bigg] \tag{113}$$

$$+ o(\varepsilon_x + \varepsilon_t). \tag{114}$$

Therefore, up to additive terms independent of θ,

$$\mathcal{L}(F_\theta) \equiv \frac{1}{4}\mathbb{E}_{p_d(x,t)}\Bigg[\varepsilon_x \left\|\nabla_x \log p_\theta(x,t) - \nabla_x \log p_d(x,t)\right\|^2 \tag{115}$$

$$+ \varepsilon_t \left(\partial_t \log p_\theta(x,t) - \partial_t \log p_d(x,t)\right)^2\Bigg] + o(\varepsilon_x + \varepsilon_t). \tag{116}$$

Thus, infinitesimal Gaussian perturbations recover the dual score matching objective [34]: one term matches the spatial score $\nabla_x \log p_d(x,t)$, while the other matches the temporal score $\partial_t \log p_d(x,t)$.

B Special cases of our method

In this section, we detail how many popular losses for learning energy-based models are in fact special cases of our loss.

First, we express the stNCE loss function in a similar form as Ceylan and Gutmann [13]. Recall the stNCE loss function:

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_+}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right] - \mathbb{E}_{p_-}\left[\log\big(1-\mathrm{sigmoid}(F_\theta(x,t,x',t'))\big)\right]. \tag{117}$$

Since $p_+(x,t,x',t') = p_d(x,t)p_n(x',t'|x,t)$ and $p_-(x,t,x',t') = p_d(x',t')p_n(x,t|x',t')$, due to symmetry one can switch x,t and x',t' without affecting the correctness. As such, we can simply reuse x,t,x',t' as x',t',x,t and write it using a single term, given by

$$\begin{aligned}
\mathcal{L}_{\mathrm{stNCE}}(F_\theta) &= -2\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right] && (118)\\
&= -2\mathbb{E}_{p_d(x|t)p(t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right], && (119)\\
F_\theta(x,t,x',t') &= \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) && (120)\\
&\quad - \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). && (121)
\end{aligned}$$

Observe that it concerns two choices: the time prior $p(t)$ and the perturbation kernel $p_n(x',t'|x,t)$. We next show how to recover exisiting methods using the above formulation.

B.1 Summary

We summarize in Table 3 how many methods in the literature can be viewed as special cases under our stNCE framework. In particular, the proof for Trajectory Score Matching can be found in Section A.3, and the proof for Dual Score Matching can be found in Section A.4. In the following we show how the previous methods fall under our stNCE framework.

We next write some additional precisions for the above table. The FCE parameterizes the the reference distribution as a flow-based model $p_d(x|0) = p_\beta(x)$ that is updated during training. Similarly, the CD updates the perturbation kernel during training as $p_{\mathrm{MCMC}}(x'|x)$, that is a MCMC kernel with stationary distribution $\mathrm{stopgrad}(p_\theta)$.

B.2 Noise-Contrastive Estimation (NCE)

Original loss The standard Noise-Contrastive Estimation (NCE) objective [36] is given by

$$\begin{aligned}
\mathcal{L}_{\mathrm{NCE}}(\theta) &= -\mathbb{E}_{p_d(x|1)}\left[\log \mathrm{sigmoid}(\log p_\theta(x) - \log p_d(x|0))\right] && (122)\\
&\quad - \mathbb{E}_{p_d(x|0)}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x) - \log p_d(x|0))\right)\right]. && (123)
\end{aligned}$$

Our loss Recall that the generic stNCE loss is

$$\begin{aligned}
\mathcal{L}_{\mathrm{stNCE}}(F_\theta) &= -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right] && (124)\\
&\quad - \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right)\right], && (125)
\end{aligned}$$

where the logit is

$$\begin{aligned}
F_\theta(x,t,x',t') &= \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) && (126)\\
&\quad - \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). && (127)
\end{aligned}$$

and the model distribution at time zero is known, $p_\theta(x,0) := p_0(x)$. Recall that $p_d(x,t) = p(t)p_d(x|t)$. Choosing the prior on time as $p(t) = \frac{1}{2}\delta_1(t) + \frac{1}{2}\delta_0(t)$, and the perturbation kernel as $p_n(x',t'|x,t) = \delta_x(x')\delta_{1-t}(t')$ simplifies the loss. In particular, the perturbation leaves the spatial variable unchanged and swaps the temporal index. We now obtain

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_1(x)}\left[\log \mathrm{sigmoid}(F_\theta(x,1,x,0))\right] - \mathbb{E}_{p_0(x)}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,0,x,1))\right)\right]. \tag{128}$$

where

$$\begin{aligned}
F_\theta(x,1,x,0) &= \log p_\theta(x|1) + \log p(1) + \log \delta_x(x) + \log \delta_0(0) && (129)\\
&\quad - \log p_\theta(x|0) - \log p(0) - \log \delta_x(x) - \log \delta_1(1), && (130)\\
&= \log p_\theta(x|1) - \log p_0(x) && (131)\\
F_\theta(x,0,x,1) &= \log p_0(x) - \log p_\theta(x|1). && (132)
\end{aligned}$$

Table 3: stNCE recovers known methods, obtained by specific choices of the time prior and perturbation kernel.

Method	**Time prior**	**Perturbation kernel** $x', t'\|x, t$
	Temporal variation	
Noise-Contrastive Estimation (NCE) [36]	$\frac{1}{2}\delta_0(t) + \frac{1}{2}\delta_1(t)$	$\delta_x(x')\,\delta_{1-t}(t')$
Flow-Contrastive Estimation (FCE) [27]	$\frac{1}{2}\delta_0(t) + \frac{1}{2}\delta_1(t)$	$\delta_x(x')\,\delta_{1-t}(t')$
Temporal NCE (tNCE)[64; 60; 1]	$p_t(t)$	$\delta_x(x')\,p_n(t' \mid t)$
Time Score Matching [20]	$p_t(t)$	$\delta_x(x')\,\delta_{t+\epsilon t}(t'), \quad \epsilon \to 0$
	Spatial variation	
Conditional NCE (CNCE) [13]	$\delta_1(t)$	$p_n(x' \mid x)\,\delta_t(t')$
Contrastive Divergence (CD) [38]	$\delta_1(t)$	$p_{\mathrm{MCMC}}(x' \mid x)\,\delta_t(t')$
Temporal CNCE (tCNCE)	$p_t(t)$	$p_n(x' \mid x)\,\delta_t(t')$
Space Score Matching [43; 81]	$\delta_1(t)$	$\mathcal{N}(x'; x, \epsilon I)\,\delta_t(t'), \quad \epsilon \to 0$
Energy-based Diffusion [71; 75]	$p_t(t)$	$\mathcal{N}(x'; x, \epsilon I)\,\delta_t(t'), \quad \epsilon \to 0$
	Spatiotemporal variation	
Spatiotemporal NCE (stNCE)	$p_t(t)$	$p_n(x' \mid x, t, t')\,p_n(t' \mid t)$
stNCE-m	$p_t(t)$	$\frac{1}{2}\delta_t(t')p_n(x'\|x) + \frac{1}{2}\delta_x(x')p_n(t'\|t)$
stNCE-w	$p_t(t)$	$\mathcal{N}\left(x' \mid x, \sigma^2 I\right)\,p_n(t' \mid t)$
stNCE-s	$p_t(t)$	Eq. 28 with $\nabla_x \log p_t(x)$ given by the model, along with $p_n(t' \mid t)$
stNCE-o	$p_t(t)$	Eq. 28 with oracle access to the ground truth $\nabla_x \log p_t(x)$, along with $p_n(t' \mid t)$
stNCE-o (forward-reverse, infinitesimal, SDE)	$p_t(t)$	see Theorem 3
stNCE-o (forward-reverse, infinitesimal, ODE)	$p_t(t)$	see Theorem 4
Dual Score Matching [34]	$p_t(t)$	$\mathcal{N}([x', t']; [x, t], \epsilon I) \quad \epsilon \to 0$

Substituting these expressions back into the stNCE loss yields

$$\mathcal{L}_{\mathrm{stNCE}} = -\mathbb{E}_{p_d(x|1)}\left[\log \mathrm{sigmoid}(\log p_\theta(x) - \log p_0(x))\right] \tag{133}$$

$$- \mathbb{E}_{p_d(x|0)}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x) - \log p_0(x))\right)\right], \tag{134}$$

which exactly recovers the standard NCE objective.

B.3 Flow-Contrastive Estimation (FCE)

This is a special case of NCE where the reference distribution $p_d(x|0)$ is not fixed beforehand but is instead updated adaptively during training. In particular, FCE parameterizes the reference distribution as a flow-based model $p_d(x|0) = p_\beta(x)$ that is updated during the optimization.

B.4 Temporal Noise-Contrastive Estimation (tNCE)

Original loss The temporal Noise-Contrastive Estimation (tNCE) objective which is heavily inspired by Rhodes et al. [64] is given by

$$\mathcal{L}_{\mathrm{tNCE}}(\theta) = -\mathbb{E}_{p_d(x,t)p_n(t'|t)}\left[\log \mathrm{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x|t') + \Delta(t,t'))\right] \tag{135}$$

$$- \mathbb{E}_{p_d(x,t')p_n(t|t')}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x|t') + \Delta(t,t'))\right)\right], \tag{136}$$

where

$$\Delta(t,t') = \log p(t) + \log p_n(t'|t) - \log p(t') - \log p_n(t|t') \tag{137}$$

is known to the user. For example, the user can choose $p(t)$ to be uniform over the interval $[0,1]$, and $p_n(t'|t)$ to be a Gaussian perturbation. In that case, $\Delta(t,t') = 0$.

Our loss Recall that the generic stNCE loss is

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right] \tag{138}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right)\right], \tag{139}$$

where the logit is

$$F_\theta(x,t,x',t') = \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) \tag{140}$$

$$- \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). \tag{141}$$

Choosing the perturbation kernel as $p_n(x',t'|x,t) = \delta_x(x')p_n(t'|t)$ removes all spatial perturbations. In particular, the perturbation leaves the spatial variable unchanged while perturbing only the temporal index. We now obtain

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(t'|t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x,t'))\right] \tag{142}$$

$$- \mathbb{E}_{p_d(x,t')p_n(t|t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,t,x,t'))\right)\right]. \tag{143}$$

Moreover,

$$F_\theta(x,t,x,t') = \log p_\theta(x|t) + \log p(t) + \log p_n(t'|t) - \log p_\theta(x|t') - \log p(t') - \log p_n(t|t') \tag{144}$$

$$= \log p_\theta(x|t) - \log p_\theta(x|t') + \Delta(t,t'). \tag{145}$$

Substituting these expressions back into the stNCE loss yields

$$\mathcal{L}_{\mathrm{stNCE}} = -\mathbb{E}_{p_d(x,t)p_n(t'|t)}\left[\log \mathrm{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x|t') + \Delta(t,t'))\right] \tag{146}$$

$$- \mathbb{E}_{p_d(x,t')p_n(t|t')}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x|t) - \log p_\theta(x|t') + \Delta(t,t'))\right)\right], \tag{147}$$

which exactly recovers the tNCE objective.

B.5 Conditional Noise-Contrastive Estimation (CNCE)

Original loss The Conditional Noise-Contrastive Estimation (CNCE) objective [13] is given by

$$\mathcal{L}_{\mathrm{CNCE}}(\theta) \tag{148}$$

$$= -\mathbb{E}_{p_d(x|1)p_n(x'|x)}\left[\log \mathrm{sigmoid}(\log p_\theta(x) + \log p_n(x'|x) - \log p_\theta(x') - \log p_n(x|x'))\right] \tag{149}$$

$$- \mathbb{E}_{p_d(x'|1)p_n(x|x')}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x) + \log p_n(x'|x) - \log p_\theta(x') - \log p_n(x|x'))\right)\right]. \tag{150}$$

Our loss Recall that the generic stNCE loss is

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right] \tag{151}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right)\right], \tag{152}$$

where the logit is

$$F_\theta(x,t,x',t') = \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) \tag{153}$$

$$- \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). \tag{154}$$

Choosing the prior on time as $p(t) = \delta_1(t)$ and the perturbation kernel as $p_n(x', t'|x, t) = p_n(x'|x)\delta_t(t')$ removes all temporal perturbations. In particular, the temporal index remains fixed while the perturbation acts only on the spatial variable. We now obtain

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x)p_n(x'|x)}\left[\log \mathrm{sigmoid}(F_\theta(x, 1, x', 1))\right] \tag{155}$$

$$- \mathbb{E}_{p_d(x')p_n(x|x')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x, 1, x', 1))\right)\right]. \tag{156}$$

Moreover,

$$F_\theta(x, 1, x', 1) = \log p_\theta(x|1) + \log p(1) + \log p_n(x'|x) - \log p_\theta(x'|1) \tag{157}$$

$$- \log p(1) - \log p_n(x|x') \tag{158}$$

$$= \log p_\theta(x|1) + \log p_n(x'|x) - \log p_\theta(x'|1) - \log p_n(x|x'). \tag{159}$$

Identifying $p_\theta(x|1)$ with $p_\theta(x)$, we obtain

$$F_\theta(x, 1, x', 1) = \log p_\theta(x) + \log p_n(x'|x) - \log p_\theta(x') - \log p_n(x|x'). \tag{160}$$

Substituting these expressions back into the stNCE loss yields

$$\mathcal{L}_{\mathrm{stNCE}} \tag{161}$$

$$= -\mathbb{E}_{p_d(x|1)p_n(x'|x)}\left[\log \mathrm{sigmoid}(\log p_\theta(x) + \log p_n(x'|x) - \log p_\theta(x') - \log p_n(x|x'))\right] \tag{162}$$

$$- \mathbb{E}_{p_d(x'|1)p_n(x|x')}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x) + \log p_n(x'|x) - \log p_\theta(x') - \log p_n(x|x'))\right)\right], \tag{163}$$

which exactly recovers the CNCE objective.

B.6 Contrastive Divergence (CD)

Yair and Michaeli [83] observed that this is a special case of CNCE where the perturbation kernel $p_n(x'|x)$ is an MCMC kernel with stationary distribution $\mathrm{stopgrad}(p_\theta)$ constantly updated during training.

B.7 Temporal Conditional Noise-Contrastive Estimation (tCNCE)

Original loss The temporal Conditional Noise-Contrastive Estimation (tCNCE) objective, which is a natural extension of the CNCE objective, is given by

$$\mathcal{L}_{\mathrm{tCNCE}}(\theta) \tag{164}$$

$$= -\mathbb{E}_{p_d(x,t)p_n(x'|x)}\left[\log \mathrm{sigmoid}(\log p_\theta(x|t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p_n(x|x'))\right] \tag{165}$$

$$- \mathbb{E}_{p_d(x',t)p_n(x|x')}\left[\log\left(1 - \mathrm{sigmoid}(\log p_\theta(x|t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p_n(x|x'))\right)\right]. \tag{166}$$

Our loss Recall that the generic stNCE loss is

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x, t, x', t'))\right] \tag{167}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x, t, x', t'))\right)\right], \tag{168}$$

where the logit is

$$F_\theta(x, t, x', t') = \log p_\theta(x|t) + \log p(t) + \log p_n(x', t'|x, t) \tag{169}$$

$$- \log p_\theta(x'|t') - \log p(t') - \log p_n(x, t|x', t'). \tag{170}$$

Choosing the perturbation kernel as $p_n(x', t'|x, t) = p_n(x'|x)\delta_t(t')$ leaves the temporal index unchanged while perturbing only the spatial variable. We now obtain

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x'|x)}\left[\log \mathrm{sigmoid}(F_\theta(x, t, x', t))\right] \tag{171}$$

$$- \mathbb{E}_{p_d(x',t)p_n(x|x')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x, t, x', t))\right)\right]. \tag{172}$$

Moreover,

$$F_\theta(x,t,x',t) = \log p_\theta(x|t) + \log p(t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p(t) - \log p_n(x|x') \tag{173}$$

$$= \log p_\theta(x|t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p_n(x|x'). \tag{174}$$

Substituting these expressions back into the stNCE loss yields

$$\mathcal{L}_{\text{stNCE}} \tag{175}$$

$$= -\mathbb{E}_{p_d(x,t)p_n(x'|x)}\left[\log \text{sigmoid}(\log p_\theta(x|t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p_n(x|x'))\right] \tag{176}$$

$$- \mathbb{E}_{p_d(x',t)p_n(x|x')}\left[\log\left(1 - \text{sigmoid}(\log p_\theta(x|t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p_n(x|x'))\right)\right], \tag{177}$$

which exactly recovers the tCNCE objective.

B.8 stNCE-m and sum of tNCE and tCNCE

Original loss Consider the sum of the tNCE and tCNCE objectives:

$$\mathcal{L} = \mathcal{L}_{\text{tNCE}} + \mathcal{L}_{\text{tCNCE}}. \tag{178}$$

Our loss Recall that the generic stNCE loss is

$$\mathcal{L}_{\text{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \text{sigmoid}(F_\theta(x,t,x',t'))\right] \tag{179}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \text{sigmoid}(F_\theta(x,t,x',t'))\right)\right], \tag{180}$$

where the logit is

$$F_\theta(x,t,x',t') = \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) \tag{181}$$

$$- \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). \tag{182}$$

Choosing the perturbation kernel as

$$p_n(x',t'|x,t) = \frac{1}{2}\delta_t(t')p_n(x'|x) + \frac{1}{2}\delta_x(x')p_n(t'|t) \tag{183}$$

creates two perturbation branches. The first branch leaves the temporal index unchanged while perturbing the spatial variable, whereas the second branch leaves the spatial variable unchanged while perturbing the temporal index. We now split the stNCE loss according to these two branches:

$$\mathcal{L}_{\text{stNCE}}(F_\theta) = -\frac{1}{2}\mathbb{E}_{p_d(x,t)p_n(x'|x)}\left[\log \text{sigmoid}(F_\theta(x,t,x',t))\right] \tag{184}$$

$$- \frac{1}{2}\mathbb{E}_{p_d(x',t)p_n(x|x')}\left[\log\left(1 - \text{sigmoid}(F_\theta(x,t,x',t))\right)\right] \tag{185}$$

$$- \frac{1}{2}\mathbb{E}_{p_d(x,t)p_n(t'|t)}\left[\log \text{sigmoid}(F_\theta(x,t,x,t'))\right] \tag{186}$$

$$- \frac{1}{2}\mathbb{E}_{p_d(x,t')p_n(t|t')}\left[\log\left(1 - \text{sigmoid}(F_\theta(x,t,x,t'))\right)\right]. \tag{187}$$

For the spatial perturbation branch, we obtain

$$F_\theta(x,t,x',t) = \log p_\theta(x|t) + \log p(t) + \log p_n(x'|x) \tag{188}$$

$$- \log p_\theta(x'|t) - \log p(t) - \log p_n(x|x') \tag{189}$$

$$= \log p_\theta(x|t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p_n(x|x'), \tag{190}$$

which is exactly the tCNCE logit. Similarly, for the temporal perturbation branch,

$$F_\theta(x,t,x,t') = \log p_\theta(x|t) + \log p(t) + \log p_n(t'|t) \tag{191}$$

$$- \log p_\theta(x|t') - \log p(t') - \log p_n(t|t') \tag{192}$$

$$= \log p_\theta(x|t) - \log p_\theta(x|t') + \Delta(t,t'), \tag{193}$$

where

$$\Delta(t,t') = \log p(t) + \log p_n(t'|t) - \log p(t') - \log p_n(t|t'). \tag{194}$$

This is exactly the tNCE logit. Substituting these expressions back into the stNCE loss yields

$$\mathcal{L}_{\text{stNCE}} = \frac{1}{2}\mathcal{L}_{\text{tCNCE}} + \frac{1}{2}\mathcal{L}_{\text{tNCE}}, \tag{195}$$

which recovers the sum of the tNCE and tCNCE objectives up to a constant rescaling factor independent of θ.

B.9 Time Score Matching

Original loss The Time Score Matching objective is given by

$$\mathcal{L}_{\mathrm{TSM}}(\theta) = \mathbb{E}_{p_d(x,t)}\left[\left(\partial_t \log p_\theta(x|t) - \partial_t \log p_d(x|t)\right)^2\right]. \tag{196}$$

where $\partial_t \log p_d(x|t)$ denotes the ground-truth temporal score. In practice, the ground-truth temporal score is not available in closed-form so many have proposed tractable ways to rewrite the objective, using integration by parts [20] or by introducing a conditioning variable [85].

Our loss Recall that the generic stNCE loss is

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right] \tag{197}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,t,x',t'))\right)\right], \tag{198}$$

where the logit is

$$F_\theta(x,t,x',t') = \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) \tag{199}$$

$$- \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). \tag{200}$$

To recover Time Score Matching, we choose a purely temporal infinitesimal perturbation. That is, the perturbation kernel leaves the spatial variable unchanged and perturbs time according to $p_n(x',t'|x,t) = \delta_x(x')\mathcal{N}(t';t,\varepsilon)$, with $\varepsilon \to 0$. We also assume that the temporal perturbation is symmetric, so $p_n(t'|t) = p_n(t|t')$ up to boundary effects. We now obtain

$$\mathcal{L}_{\mathrm{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(t'|t)}\left[\log \mathrm{sigmoid}(F_\theta(x,t,x,t'))\right] \tag{201}$$

$$- \mathbb{E}_{p_d(x,t')p_n(t|t')}\left[\log\left(1 - \mathrm{sigmoid}(F_\theta(x,t,x,t'))\right)\right]. \tag{202}$$

Moreover,

$$F_\theta(x,t,x,t') = \log p_\theta(x|t) + \log p(t) + \log p_n(t'|t) - \log p_\theta(x|t') - \log p(t') - \log p_n(t|t') \tag{203}$$

$$= \log p_\theta(x|t) - \log p_\theta(x|t') + \log p(t) - \log p(t'), \tag{204}$$

where the perturbation-kernel terms cancel by symmetry. If the time prior is locally uniform, or if it is absorbed into the time-dependent density, this reduces to

$$F_\theta(x,t,x,t') = \log p_\theta(x|t) - \log p_\theta(x|t'). \tag{205}$$

Writing $t' = t + \sqrt{\varepsilon}\xi$ with $\xi \sim \mathcal{N}(0,1)$, a first-order Taylor expansion gives

$$F_\theta(x,t,x,t') = -\sqrt{\varepsilon}\,\partial_t \log p_\theta(x|t)\xi + o(\sqrt{\varepsilon}). \tag{206}$$

Using the second-order expansion of the logistic loss around $F_\theta = 0$, the infinitesimal temporal perturbation limit gives

$$\mathcal{L}_{\mathrm{stNCE}} = \frac{\varepsilon}{4}\mathbb{E}_{p_d(x,t)}\left[\left(\partial_t \log p_\theta(x|t) - \partial_t \log p_d(x|t)\right)^2\right] + C + o(\varepsilon), \tag{207}$$

where C is independent of θ. Therefore, up to the constant factor $\varepsilon/4$ and additive terms independent of θ, the infinitesimal temporal-only limit of stNCE recovers the Time Score Matching objective.

B.10 Space Score Matching

Original loss The Space Score Matching objective is given by

$$\mathcal{L}_{\mathrm{SSM}}(\theta) = \mathbb{E}_{p_d(x|1)}\left[\left\|\nabla_x \log p_\theta(x|1) - \nabla_x \log p_d(x|1)\right\|^2\right]. \tag{208}$$

where $\nabla_x \log p_d(x|1)$ denotes the ground-truth spatial score. This quantity is unknown in closed-form so many have rewritten the objective in a tractable way using integration by parts [43] or by introducing a conditioning variable [81].

Our loss Recall that the Energy-based Diffusion loss derived above is obtained from the infinitesimal spatial-only limit of stNCE:

$$\mathcal{L}_{\text{stNCE}} = \frac{\varepsilon}{4}\mathbb{E}_{p_d(x,t)}\left[\|\nabla_x \log p_\theta(x|t) - \nabla_x \log p_d(x|t)\|^2\right] + C + o(\varepsilon). \tag{209}$$

Restricting the time distribution to the single time point $t = 1$, that is, choosing $p(t) = \delta_1(t)$, yields

$$\mathcal{L}_{\text{stNCE}} = \frac{\varepsilon}{4}\mathbb{E}_{p_d(x|1)}\left[\|\nabla_x \log p_\theta(x|1) - \nabla_x \log p_d(x|1)\|^2\right] + C + o(\varepsilon), \tag{210}$$

which exactly recovers the standard Space Score Matching objective up to a constant rescaling factor and additive terms independent of θ.

B.11 Energy-based Diffusion

Original loss The Energy-based Diffusion loss [71; 75] is given by

$$\mathcal{L}_{\text{EBD}}(\theta) = \mathbb{E}_{p_d(x,t)}\left[\|\nabla_x \log p_\theta(x|t) - \nabla_x \log p_d(x|t)\|^2\right]. \tag{211}$$

where $\nabla_x \log p_d(x|t)$ denotes the ground-truth spatial score.

Our loss Recall that the generic stNCE loss is

$$\mathcal{L}_{\text{stNCE}}(F_\theta) = -\mathbb{E}_{p_d(x,t)p_n(x',t'|x,t)}\left[\log \text{sigmoid}(F_\theta(x,t,x',t'))\right] \tag{212}$$

$$- \mathbb{E}_{p_d(x',t')p_n(x,t|x',t')}\left[\log\left(1 - \text{sigmoid}(F_\theta(x,t,x',t'))\right)\right], \tag{213}$$

where the logit is

$$F_\theta(x,t,x',t') = \log p_\theta(x|t) + \log p(t) + \log p_n(x',t'|x,t) \tag{214}$$

$$- \log p_\theta(x'|t') - \log p(t') - \log p_n(x,t|x',t'). \tag{215}$$

To recover the Energy-based Diffusion loss, we choose a purely spatial infinitesimal perturbation. That is, the perturbation kernel leaves the temporal index unchanged and perturbs the spatial variable according to $p_n(x',t'|x,t) = \mathcal{N}(x';x,\varepsilon I)\delta_t(t')$, with $\varepsilon \to 0$. Since the Gaussian perturbation is symmetric, $p_n(x'|x) = p_n(x|x')$. We now obtain

$$\mathcal{L}_{\text{stNCE}} = -\mathbb{E}_{p_d(x,t)p_n(x'|x)}\left[\log \text{sigmoid}(F_\theta(x,t,x',t))\right] \tag{216}$$

$$- \mathbb{E}_{p_d(x',t)p_n(x|x')}\left[\log\left(1 - \text{sigmoid}(F_\theta(x,t,x',t))\right)\right]. \tag{217}$$

Moreover,

$$F_\theta(x,t,x',t) = \log p_\theta(x|t) + \log p(t) + \log p_n(x'|x) - \log p_\theta(x'|t) - \log p(t) - \log p_n(x|x') \tag{218}$$

$$= \log p_\theta(x|t) - \log p_\theta(x'|t), \tag{219}$$

where the time prior terms cancel because $t' = t$, and the perturbation kernel terms cancel by symmetry. Writing $x' = x + \sqrt{\varepsilon}\xi$ with $\xi \sim \mathcal{N}(0, I)$, a first-order Taylor expansion gives

$$F_\theta(x,t,x',t) = -\sqrt{\varepsilon}\,\nabla_x \log p_\theta(x|t)^\top \xi + o(\sqrt{\varepsilon}). \tag{220}$$

Using the second-order expansion of the logistic loss around $F_\theta = 0$, the infinitesimal spatial perturbation limit gives

$$\mathcal{L}_{\text{stNCE}} = \frac{\varepsilon}{4}\mathbb{E}_{p_d(x,t)}\left[\|\nabla_x \log p_\theta(x|t) - \nabla_x \log p_d(x|t)\|^2\right] + C + o(\varepsilon), \tag{221}$$

where C is independent of θ. Therefore, up to the constant factor $\varepsilon/4$ and additive terms independent of θ, the infinitesimal spatial-only limit of stNCE recovers the Energy-Based Diffusion loss.

C Additional discussion

C.1 Compute burden of the likelihood: training-time *versus* inference-time

This work highlights how the literature of learning energy-based models falls within three families of learning objectives: temporal, spatial, and spatiotemporal. Another axis for organizing the literature is two parameterizations of the energy-based model differences, both enabling likelihood estimation but differing primarily in when the computational burden is incurred.

A first class defers this cost to inference time by integrating learned quantities, including temporal methods based on $\partial_t \log p_t(x)$ [20; 85; 18], spatial methods based on $\nabla \log p_t(x)$, and spatiotemporal approaches integrating $\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(x_t)$, such as continuous normalizing flows [15] and pathwise formulations [20]. These methods are relatively easy to train, as they avoid higher-order differentiation, but incur significant inference-time cost due to numerical integration, although recent works look into *learning* to integrate in few steps [10; 2; 61; 16].

A second class instead parameterizes these differences directly with an energy-based model, enabling one-step likelihood evaluation. This includes temporal methods such as NCE [36], spatial methods such as (denoising) score matching [43; 81], and recent spatio-temporal approaches, including ours [37; 34; 62; 1; 60]. These methods yield cheap inference—often a single network evaluation—but are harder to train, e.g., due to higher-order autodiff.

Overall, these approaches trade training-time for inference-time compute. We focus on the EBM approach, favoring expensive training for efficient inference. A systematic study of this trade-off is left for future work.

C.2 Discussion on related spatiotemporal methods

Several recent papers [1; 34; 37; 60; 62] proposed similar ideas of training both spatial and temporal differences of the energy-based models. Nevertheless, to the best of our knowledge, our work is the first to propose such a unifying framework that generalizes several of the previous work. As discussed earlier in the section, both Aggarwal et al. [1] and Ouyang et al. [60] are highly relevant to our tNCE algorithm. Essentially, the Noise-Contrastive Estimation loss used by Aggarwal et al. [1] is tNCE, while the diffusive classification loss proposed in Ouyang et al. [60] reduces to tNCE when there are two classes. As such, they suffer from the same pathology of tNCE. Both Guth et al. [34] and Plainer et al. [62] learn the time scores apart from the Stein scores and can be interpreted as special instances of our framework, as will be discussed in Appendix A.4 and Appendix C.4. He et al. [37] is related to our stNCE framework, but unlike stNCE, it is not consistent when using the model's own scores and used on its own; see further discussion in Appendix C.3.

C.3 Discussion on RNE

Next we consider RNE as proposed in He et al. [37], which was shown to be capable of regularizing energy-based diffusion models. The RNE loss is roughly given by

$$\mathcal{L} = \mathbb{E}_{x_t,t,x_{t+\Delta t},t+\Delta t} \left\| \mathrm{sg}\left(\log p(x_t|x_{t+\Delta t}) - \log p(x_{t+\Delta t}|x_t)\right) \right. \tag{222}$$

$$\left. + \log p_{t+\Delta t}(x_{t+\Delta t}) - \log p_t(x_t) \right\|^2, \tag{223}$$

where the diffusion convention of time is used, and sg denotes stop gradient. He et al. [37] showed that the regularizer, when combined with denoising score matching loss, allows training normalized energy-based models.

In our experiments, we refer to RNE with oracle access to the ground truth score as RNE-o, and RNE with the model's own scores as RNE-s. When used on its own and using the score estimates as given by the model itself, the RNE-s loss suffers from a notable pathology. Specifically, the optimum satisfying the Bayes rule constraint induced by RNE-s is not unique, in contrast to stNCE-s: while the ground-truth time-varying density p_t is one optimum, there exist infinitely many other optimums that may differ arbitrarily from the data distribution. For example, the degenerate solution where $p_t \equiv c$ for some constant c is an optimum under the Variance-Exploding noising process with Euler-Maruyama discretization, and any time-varying probability density induced by a clean data distribution that differ from the empirical distribution is also an optimum.

C.4 Comparison against Dual Score Matching

As discussed in Appendix A.4, we can recover dual score matching from stNCE. Compared with stNCE without DSM, dual score matching requires optimizing the derivatives of the network and is more computationally costly; in this case stNCE provides a unique balance between training cost and performance. When combining stNCE with DSM, as validated on ImageNet-64 in Section 6.3, it can provide better performance than dual score matching.

C.5 Broader impacts

Our work is relatively theoretical and studies how to train EBMs. Nevertheless, due to the widespread usage of EBMs, it could be used by practioners in malicious manners. We attempt to mitigate such negative impacts by advocating for responsible usage of such trained models.

D Implementation considerations

D.1 Sampling mechanisms

stNCE logits are determined by joint log densities in the form of $\log p(x,t) + \log p_n(x',t'|x,t) = \log p(t) + \log p(x|t) + \log p_n(x'|x,t,t') + \log p_n(t'|t)$. As such, one would need to take into account $\log p(t) + \log p_n(t'|t) = \log p_n(t,t')$ when forming the logit, unless $\log p_n(t,t') = \log p_n(t',t)$, in which case they cancel out with each other. We next introduce the folding mechanism, which will be beneficial afterwards.

D.1.1 Useful identity: folding mechanism

We would like to sample t' given t, such that when t is uniformly distributed between $[0,1]$, t' follows the same distribution while remaining reasonably close to t. This can be achieved using the folding mechanism in Algorithm 1.

Algorithm 1 Folding mechanism.

Require: $t \sim U(0,1)$, $\sigma > 0$
$\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$
$\tilde{t} \leftarrow t + \epsilon$
$t' = 1 - |(\tilde{t} \mod 2) - 1|$
return t'

Proposition 7 *Let $t \sim U(0,1)$ and $\epsilon \sim \mathcal{N}(0,\sigma^2)$ be independent. Define*

$$t' = 1 - |\left((t+\epsilon) \mod 2\right) - 1|, \tag{224}$$

*Then for any $\sigma > 0$, $t' \sim U(0,1)$ (**uniformity**), and $|t'-t| \leq |\epsilon|$ (**locality**).*

Proof. **Uniformity**

Denote ϵ to be the Gaussian noise with standard deviation σ. Observe that a given value of t' can be obtained from different $\tilde{t}$. Specifically, they could be $\tilde{t} = 2k + t'$ or $\tilde{t} = 2k - t'$, where $k \in Z$. We thus have

$$p_{t'}(t') = \sum_{k=-\infty}^{\infty} \left[p_{\tilde{t}}(2k+t') + p_{\tilde{t}}(2k-t')\right], \tag{225}$$

i.e. the sum of two terms. Note that

$$p_{\tilde{t}}(\tilde{t}) = \int_0^1 p_\epsilon(\tilde{t} - t)\mathrm{d}t, \tag{226}$$

and

$$p_{t'}(t') = \sum_{k=-\infty}^{\infty} \left[\int_0^1 p_\epsilon(2k + t' - t)\mathrm{d}t + \int_0^1 p_\epsilon(2k - t' - t)\mathrm{d}t\right]. \tag{227}$$

For the first term, using a change of variable $u = t - 2k$, we have

$$\sum_{k=-\infty}^{\infty} p_{\tilde{t}}(2k+t') = \sum_{k=-\infty}^{\infty} \int_{-2k}^{-2k+1} p_\epsilon(t'-u)\mathrm{d}u, \tag{228}$$

i.e. it is the sum over all intervals of the form $[2k, 2k+1]$. For the seocnd term, using a change of variable $u = 2k - t$, we have

$$\sum_{k=-\infty}^{\infty} p_{\tilde{t}}(2k-t') = \sum_{k=-\infty}^{\infty} \int_{2k-1}^{2k} p_\epsilon(t'-u)\mathrm{d}u, \tag{229}$$

i.e. it is the sum over all intervals of the form $[2k-1, 2k]$.

Summarizing, these intervals cover the entire real line. As such,

$$p_{t'}(t') = \int_{-\infty}^{\infty} p_\epsilon(t'-u)\mathrm{d}u = 1, \tag{230}$$

i.e. t' follows uniform distribution.

Locality

Denote $F(x) = 1 - |(x \mod 2) - 1|$. Observe that $F(x) = x$ for $x \in [0, 1]$, and $|F'(x)| = 1$ whenever the derivative exists. As such, F is globally $1-$Lipschitz, and

$$|t' - t| = |F(t + \epsilon) - t| = |F(t + \epsilon) - F(t)| \leq |(t + \epsilon) - t| = |\epsilon|. \tag{231}$$

□

Furthermore, given two values t and t', $p_n(t'|t) = p_n(t|t')$, as these probabilities only depend on the distances of the path between t and t', while a Gaussian is symmetric around its mean.

Note that there are different approaches to sample t, t', x and x'. In the following we describe two main sampling schemes, which we term `default` and `reuse`. We use z to denote the clean data, and consider as perturbation kernel $p_n(x', t'|x, t) = p_n(t'|t)p_n(x'|x, t, t')$.

D.1.2 default

Perhaps the most natural implementation is to follow the data generation process of stNCE; we note that this is also the sampling mechanism as employed by Aggarwal et al. [1]. Given one z, one (x, t, x', t') is constructed. It can be summarized as follows:

$$t \sim p(t), \quad x \sim p(x|z, t), \tag{232}$$

$$t' \sim p_n(t'|t), \quad x' \sim p_n(x'|x, t, t'). \tag{233}$$

However, special care is needed on the logits involving t and t'. Perhaps the simplest case is $p(t)$ is uniform and $p_n(t'|t) = p_n(t|t')$, as the part involving the time prior and time proposal naturally cancels out. Interestingly, when $p(t)$ is uniform, the folding mechanism from Section D.1.1 guarantees that $p_n(t'|t) = p_n(t|t')$, while ensuring that the marginal distribution $p(t') = \int p_n(t'|t)p(t)\mathrm{d}t$ remains uniform.

When combining the NCE loss with a loss on the space score, we can use the sampled x and t as they are indeed unbiased samples from the stochastic interpolant.

D.1.3 reuse

Alternatively, one can consider the following mechanism:

$$t_0 \sim p(t_0), \quad t_1 \sim p_n(t_1|t_0), \tag{234}$$

$$t = [t_0, t_1], \quad t' = [t_1, t_0], \tag{235}$$

$$x_0 \sim p(x_0|z, t_0), \quad x_1 \sim p(x_1|z, t_1), \tag{236}$$

$$x'_0 \sim p_n(x'_0|x_0, t_0, t_1), \quad x'_1 \sim p_n(x'_1|x_1, t_1, t_0), \tag{237}$$

$$x = [x_0, x_1], \quad x' = [x'_0, x'_1]. \tag{238}$$

Given one z, two (x, t, x', t') instances are constructed, using (t, t') and (t', t), respectively. In this case, regardless of the choice of $p(t_0)$ and $p_n(t_1, t_0)$, the sampling mechanism ensures that $p_n(t, t') = p_n(t', t)$. Nevertheless, one may still prefer to employ the folding mechanism from Section D.1.1 so as to make t and t' remain uniform. Furthermore, this enables mechanisms to reduce variances and even save computations, as detailed below.

If we combine the NCE loss with a space score loss, as x and t, which contain x_0, x_1, t_0, t_1, are all unbiased samples for the stochastic interpolant, we use them when forming the latter loss.

General case Rhodes et al. [64] noted that reusing the same noise can reduce variance. As such, we could use the same noise when simulating $p(x|z, t)$ or $p_n(x'|x, t, t')$. Given one z, we would like to obtain samples from t_0 and t_1, and we can achieve this by interpolating between z and the same noise using t_0 and t_1. Certain $p_n(x'|x, t, t')$ adds Gaussian noise of some standard deviations, and we can add the same Gaussian noise to x_0 and x_1 when simulating x'_0 and x'_1.

stNCE with forward-reverse In the case of stNCE with forward-reverse kernel, p_n could naturally correspond to diffusion forward process. As such, we could exploit this identity to reduce computations. Specifically, without loss of generality, one can design it so, such that t_0 is always smaller than t_1 by adding a sorting step upon the sampled t_0 and t_1. For numerical stability, one additionally ensure that the difference between them is not too small. Observe that one could reuse x_0 for x_1', suppose one simulate x_0 from x_1 by following the diffusion forward process from t_1 to t_0. Furthermore, stNCE with forward-reverse kernel requires the scores always for the smaller t, and we only need the scores for t_0. When we combine stNCE loss with a loss on the space score, these scores are naturally needed for the latter loss, as x_0 are indeed samples from the corresponding distributions.

In order for the forward kernel to perfectly coincide with the forward process while avoiding numerical issues, we additionally employ a mechanism to guarantee that t_1 and t_0 are apart by at least ϵ. We follow the convention that this mechanism is employed when we use the `reuse` scheme, even for algorithms apart from stNCE with forward-reverse.

D.2 Implementing forward-reverse kernel

Denote the forward denoising process as $x_t = \alpha_t z + \beta_t n$, where z are data samples and n are noise samples from standard normal distribution.

From the perspective of diffusion and flows, the choice of the kernel in the case where $t' < t$ is obvious (note that we use the flow matching convention of time), as it can be given by the forward process of diffusion models. In this case, we can choose

$$p(x'|x,t',t) = \mathcal{N}\left(x' \middle| \frac{\alpha_{t'}}{\alpha_t}x, \beta_{t'}^2 - \frac{\alpha_{t'}^2\beta_t^2}{\alpha_t^2}\right), \tag{239}$$

which essentially generates samples from $p(x'|t')$ given samples from $p(x|t)$.

Under the scenario that $t' > t$, with the above forward process one might be tempted to use as transition kernel the one that leads to the reverse kernel of diffusions, in which case we would sample from the posterior distribution. However, this posterior is highly non-trivial and is available only under specific data distributions.

D.2.1 Recovery likelihood

Inspired by diffusion recovery likelihood [28], we can construct approximations to the reverse kernel. In the following, to simplify notations, we denote $p(x'|x,t',t)$ as $p(x_\tau|x_t)$, where $x_t = \alpha x_\tau + \beta n$, with $\alpha = \frac{\alpha_t}{\alpha_\tau}$ and $\beta = \sqrt{\beta_t^2 - \frac{\alpha_t^2\beta_{t'}^2}{\alpha_{t'}^2}}$. We have

$$\begin{aligned}
\log p(x_\tau|x_t) &= \log p(x_\tau) + \log p(x_t|x_\tau) + c && (240)\\
&\approx \log p(x_t) + \langle \nabla_x \log p(x_t), x_\tau - x_t\rangle - \frac{1}{2\beta^2}\|x_t - \alpha x_\tau\|^2 + c && (241)\\
&= \langle \nabla_x \log p(x_t), x_\tau\rangle - \frac{\alpha^2}{2\beta^2}\left\|\frac{x_t}{\alpha} - x_\tau\right\|^2 + c && (242)\\
&= \langle \nabla_x \log p(x_t), x_\tau\rangle - \frac{\alpha^2}{2\beta^2}\|x_\tau\|^2 + \frac{\alpha^2}{2\beta^2}2\left\langle x_\tau, \frac{x_t}{\alpha}\right\rangle + c && (243)\\
&= -\frac{\alpha^2}{2\beta^2}\left(\langle x_\tau, x_\tau\rangle - 2\left\langle x_\tau, \frac{x_t}{\alpha}\right\rangle - \frac{2\beta^2}{\alpha^2}\langle x_\tau, \nabla_x \log p(x_t)\rangle\right) + c && (244)\\
&= -\frac{\alpha^2}{2\beta^2}\left\|x_\tau - \left(\frac{x_t}{\alpha} + \frac{\beta^2}{\alpha^2}\nabla_x \log p(x_t)\right)\right\|^2 + c, && (245)
\end{aligned}$$

which motivates using the following Gaussian approximation:

$$p(x_\tau|x_t) = \mathcal{N}\left(x_\tau \middle| \frac{x_t}{\alpha} + \frac{\beta^2}{\alpha^2}\nabla_x \log p(x_t), \frac{\beta^2}{\alpha^2}I\right). \tag{246}$$

D.2.2 Exponential integrator

The sampling scheme based on diffusion recovery likelihood builds on the assumption that the score is locally constant across x. However, this assumption may not be ideal. Indeed, works on accelerated diffusion samplers [56; 29] typically assume that the score is locally constant across t.

Here we focus on the exponential integrator for the reverse SDE as presented in Gonzalez et al. [29], specifically the SEEDS-1 variant, which is a first order SDE solver. The update rule is given by

$$x_\tau = \frac{\alpha_\tau}{\alpha_t} x_t - 2\beta_\tau \left(\exp(h) - 1\right) F(x_t, t) - \beta_\tau \sqrt{\exp(2h) - 1}\epsilon, \tag{247}$$

where F is the output of noise prediction model, and $h = \lambda_\tau - \lambda_t = \log\left(\frac{\alpha_\tau}{\beta_\tau}\right) - \log\left(\frac{\alpha_t}{\beta_t}\right)$. As such, $\exp(h) = \frac{\alpha_\tau \beta_t}{\beta_\tau \alpha_t}$ and $\exp(2h) = \left(\frac{\alpha_\tau \beta_t}{\beta_\tau \alpha_t}\right)^2$.

Observe that the update rule of SEEDS-1 naturally induces a Gaussian distribution. The mean is given by

$$\frac{\alpha_\tau}{\alpha_t} x_t + 2\beta_\tau \left(\frac{\alpha_\tau \beta_t}{\beta_\tau \alpha_t} - 1\right) \beta_t \nabla_x \log p_t(x) \tag{248}$$

$$= \frac{\alpha_\tau}{\alpha_t} x_t + 2\left(\frac{\alpha_\tau \beta_t^2}{\alpha_t} - \beta_\tau \beta_t\right) \nabla_x \log p_t(x) \tag{249}$$

$$= \frac{\alpha_\tau}{\alpha_t} x_t + 2\beta_t \left(\frac{\alpha_\tau}{\alpha_t}\beta_t - \beta_\tau\right) \nabla_x \log p_t(x), \tag{250}$$

and the covariance is given by

$$\beta_\tau^2 \left(\left(\frac{\alpha_\tau \beta_t}{\beta_\tau \alpha_t}\right)^2 - 1\right) I = \left(\frac{\alpha_\tau^2 \beta_t^2}{\alpha_t^2} - \beta_\tau^2\right) I. \tag{251}$$

Compared with the one obtained through recovery likelihood, the only difference is the scaling term in front of the score.

Consider the scenario where $\alpha_t = t$ and $\beta_t = 1 - t$. The above expression simplifies, resulting in a Gaussian distribution with mean given by

$$\frac{\tau}{t} x_t + \frac{2(1-t)(\tau - t)}{t} \nabla_x \log p_t(x_t), \tag{252}$$

and covariance given by

$$\frac{(\tau - t)(t + \tau - 2t\tau)}{t^2} I. \tag{253}$$

D.3 Parameterization of energy-based diffusion models

A proper parameterization of energy-based diffusion models, $E_\theta(x, t)$, could be crucial in practice, where the training stability of the high variance of the Denoising Score Matching objective at low noise levels required to be cured [66]. Let $x_t = \alpha_t x_1 + \sigma_t z$. In standard Diffusion models, Karras et al. [44] proposes to parameterize a denoiser with skip connection:

$$D_\theta(x_t, t) = c_{\text{skip}}(t) x_t + c_{\text{out}}(t) f_\theta(c_{\text{in}}(t) x_t, c_t(t)), \tag{254}$$

where $c_{\text{skip}}(t), c_{\text{out}}(t), c_{\text{in}}(t)$ are predefined parameters that depend on the variance of target data and noise schedules. Assumed an optimal denoiser, the denoising score, $\nabla \log p_t$, is then recovered through Tweedie's formula:

$$c_{\text{skip}}(t) x_t + c_{\text{out}}(t) f_\theta(c_{\text{in}}(t) x_t, c_t(t)) = \alpha_t^{-1}\big(x_t + \sigma_t^2 \nabla \log p_t(x_t)\big). \tag{255}$$

Therefore, one could parameterize an energy network as [75]:

$$U_\theta(x_t, t) = \frac{1 - \alpha_t c_{\text{skip}}(t)}{2\sigma_t^2} \|x_t\|^2 - \frac{\alpha_t c_{\text{out}}(t)}{c_{\text{in}}(t)\sigma_t^2} F_\theta(c_{\text{in}}(t) x_t, c_t(t)), \tag{256}$$

where F_θ is an arbitrary scalar network. Then applying Tweedie's formula, it recovers the EDM preconditioning:

$$D_\theta(x_t, t) = \alpha_t^{-1}\big(x_t - \sigma_t^2 \nabla U_\theta(x_t, t)\big) = c_{\text{skip}}(t) x_t + c_{\text{out}}(t) \nabla_{c_{\text{in}}(t) x_t} F_\theta(c_{\text{in}}(t) x_t, c_t(t)). \tag{257}$$

D.3.1 Pathology of model without preconditioning

In the following, we focus on the case of using Cond OT probability path [53].

One might be tempted to directly parameterize the EBMs using some neural architectures and obtain its gradient as the score. However, recall that the velocity can be expressed using the score as

$$v = \frac{x}{t} + \frac{1-t}{t}\nabla \log p_t(x_t). \tag{258}$$

As such, when t is small, any small error in the estimated $\nabla \log p_t(x_t)$ is significantly amplified.

D.3.2 Choice of preconditioning

Next we develop our exact choice of preconditioning. In the following we use σ to denote the standard deviation of data.

$$c_{in}(t) = \frac{1}{\sqrt{t^2\sigma^2 + (1-t)^2}}, \tag{259}$$

$$c_{skip}(t) = \frac{t\sigma^2}{t^2\sigma^2 + (1-t)^2}, \tag{260}$$

$$c_{out}(t) = \frac{(1-t)\sigma}{\sqrt{t^2\sigma^2 + (1-t)^2}}, \tag{261}$$

$$c_t(t) = \log\left(\frac{1-t}{t}\right). \tag{262}$$

With the above choices, the velocity can be expressed using the model as

$$v = \frac{c_{skip}(t) - 1}{1-t}x + \sigma\nabla_x F(c_{in}x, c_t(t)), \tag{263}$$

which avoids the singularity issue under the vanilla model.

$c_{in}(t)$ serves to keep the inputs to the network with unit standard deviation. We have

$$\text{Var}(x_t) = t^2\sigma^2 + (1-t)^2, \tag{264}$$

$$\text{Std}(x_t) = \sqrt{t^2\sigma^2 + (1-t)^2}, \tag{265}$$

$$c_{in}(t) = \frac{1}{\sqrt{t^2\sigma^2 + (1-t)^2}}. \tag{266}$$

$c_{skip}(t)$ serves as the optimal linear coefficient to predict x_1 based on x_t, and its optimal value is given by

$$c_{skip}(t) = \frac{\text{Cov}(x_1, x_t)}{\text{Var}(x_t)} \tag{267}$$

$$= \frac{\text{Cov}(x_1, tx_1 + (1-t)x_0}{t^2\sigma^2 + (1-t)^2} \tag{268}$$

$$= \frac{t\sigma^2}{t^2\sigma^2 + (1-t)^2}. \tag{269}$$

$c_{out}(t)$ serves to keep the network output have variance 1, and can be derived by considering the variance of $x_1 - c_{skip}(t)x_t$. Observe that

$$x_1 - c_{skip}(t)x_t \tag{270}$$

$$= x_1 - c_{skip}(t)(tx_1 + (1-t)x_0) \tag{271}$$

$$= (1 - tc_{skip}(t))x_1 - (1-t)c_{skip}(t)x_0, \tag{272}$$

and

$$\text{Var}(x_1 - c_{skip}(t)x_t) = (1 - tc_{skip}(t))^2 \sigma^2 + ((1-t)c_{skip}(t))^2 . \tag{273}$$

Denote the denominator of $x_1 - c_{skip}(t)x_t$ to be $D = t^2\sigma^2 + (1-t)^2$, as such $c_{skip}(t) = \frac{t\sigma^2}{D}$, and

$$1 - tc_{skip}(t) = 1 - \frac{t^2\sigma^2}{D} = \frac{(1-t)^2}{D}. \tag{274}$$

We have

$$\text{Var}(x_1 - c_{skip}(t)x_t) = \left(\frac{(1-t)^2}{D}\right)^2 \sigma^2 + \left(\frac{(1-t)t\sigma^2}{D}\right)^2 \tag{275}$$

$$= \frac{(1-t)^4\sigma^2 + (1-t)^2t^2\sigma^4}{D^2} \tag{276}$$

$$= \frac{(1-t)^2\sigma^2\left((1-t)^2 + t^2\sigma^2\right)}{D^2} \tag{277}$$

$$= \frac{(1-t)^2\sigma^2}{D}. \tag{278}$$

As such,

$$c_{out}(t) = \text{Std}(x_1 - c_{skip}(t)x_t) = \frac{(1-t)\sigma}{\sqrt{t^2\sigma^2 + (1-t)^2}}. \tag{279}$$

$c_{noise}(t)$ provides time embedding that reflects the effective noise level. With Cond OT path, the ratio of noise std to signal std is given by $\frac{1-t}{t\sigma}$, and we have

$$c_{noise}(t) = \ln\left(\frac{1-t}{t\sigma}\right). \tag{280}$$

Flow matching model is known to work well without needing $c_{in}(t)$ and $c_t(t)$. As such, we can drop them from F, while keeping them elsewhere.

It might be tempting to directly employ the above choices to parameterize the energy and velocity. Observe that $\frac{c_{out}(t)}{c_{in}(t)} = (1-t)\sigma$. We have

$$U(x_t, t) = \frac{1 - tc_{skip}(t)}{2(1-t)^2}\|x_t\|^2 - \frac{\sigma t}{1-t}F, \tag{281}$$

$$v = \frac{c_{skip}(t) - 1}{1-t}x + \sigma\nabla_x F. \tag{282}$$

Note that $c_{skip}(1) = 1$. As such, when $t = 1$, $U(x_t, t)$ has a singularity.

In order to resolve the singularity, we can either sample time not until 1 or replace $1 - t$ with $1 - at$ for some $a < 1$. In this work, we set $a = 0.75$.

D.4 Score and velocity

Flow Matching [53] provides an alternative viewpoint for diffusion models. When the noise distribution is Gaussian, we can convert between the score and the velocity. Similar identities have been shown in previous works, e.g. Holderrieth et al. [40].

Consider $x_t = \alpha_t x_1 + \beta_t x_0$, with x_0 following standard normal distribution. For simplicity, we ignore subscripts t of α_t and β_t below. We have

$$v_t(x) = \frac{\dot\alpha}{\alpha}x + \left(\frac{\dot\alpha\beta^2}{\alpha} - \dot\beta\beta\right)\nabla\log p_t(x), \tag{283}$$

$$\nabla\log p_t(x) = \frac{v_t(x) - \frac{\dot\alpha}{\alpha}x}{\frac{\dot\alpha\beta^2}{\alpha} - \dot\beta\beta}. \tag{284}$$

Proof. In this proof, expectations are taken by conditioning on $x_t = x$, and we simply denote them as $\mathbb{E}$.

$$\nabla \log p_t(x|x_1) = -\frac{1}{\beta^2}(x - \alpha x_1), \tag{285}$$

$$v_t(x|x_1) = \dot{\alpha} x_1 + \dot{\beta}\frac{x - \alpha x_1}{\beta} = \dot{\alpha} x_1 - \dot{\beta}\beta \nabla \log p_t(x|x_1), \tag{286}$$

$$\nabla \log p_t(x) = \mathbb{E}\left[-\frac{x - \alpha x_1}{\beta^2}\right] = -\frac{x - \alpha \mathbb{E}[x_1]}{\beta^2}, \tag{287}$$

$$\mathbb{E}[x_1] = \frac{\beta^2 \nabla \log p_t(x) + x}{\alpha}, \tag{288}$$

$$v_t(x) = \mathbb{E}\left[\dot{\alpha} x_1 - \dot{\beta}\beta \nabla \log p_t(x|x_1)\right] = \dot{\alpha}\mathbb{E}[x_1] - \dot{\beta}\beta \mathbb{E}[\nabla \log p_t(x|x_1)] \tag{289}$$

$$= \frac{\dot{\alpha}}{\alpha}\left(\beta^2 \nabla \log p_t(x) + x\right) - \dot{\beta}\beta \nabla \log p_t(x) = \frac{\dot{\alpha}}{\alpha} x + \left(\frac{\dot{\alpha}\beta^2}{\alpha} - \dot{\beta}\beta\right) \nabla \log p_t(x), \tag{290}$$

$$\nabla \log p_t(x) = \frac{v_t(x) - \frac{\dot{\alpha}}{\alpha} x}{\frac{\dot{\alpha}\beta^2}{\alpha} - \dot{\beta}\beta}. \tag{291}$$

□

The above connection between the score and the velocity also enables one to use flow matching loss to train models under different parameterizations [51]. Therefore, while Dual SM, following Guth et al. [34], uses the original denoising score matching loss, for stNCE-s+DSM we adapt flow matching loss to train the space score. As it still trains the space score, we refer to it still as a space score matching loss.

D.5 Obtaining approximately normalized energies

Guth et al. [34] looked into how to obtain estimates of approximately normalized energies from the trained models. Specifically, as they trained the model using dual score matching, upon optimality the model could still possess an unknown normalization constant. They estimated the constant based on the entropy of the initial noise distribution.

In similar spirit, we estimate such constants for our models as well.

Spatial methods, even when training models that are time-varying to optimality, could naturally lead to different normalizing constants across times. As such, in order to normalize them, we require access to the ground truth energy at the time point corresponding to data.

For temporal methods, we extract the density ratio learned by the model and add it to the known log density of the Gaussian distribution at $t = 0$.

For spatiotemporal methods, for general model parameterizations, we estimate such constant through importance sampling around $t = 0$ using the tractable initial noise distribution as the proposal as

$$x_i \sim p_d(x|0), \quad \widehat{Z_0} = \frac{1}{M}\sum_{i=1}^{M} \frac{\exp\left(-E_\theta(x_i, 0)\right)}{p_d(x_i|0)}, \quad \ell_\theta(x, t) = -E_\theta(x, t) - \log \widehat{Z_0}, \tag{292}$$

where E_θ is the raw output energy and ℓ_θ is the estimated log density. For specific model parameterizations, e.g. the one as in Appendix D.3, the energy at $t = 0$ is analytically available, hence the constant is also tractable, in which case we directly use the analytical one.

E Experimental details

Recall that default scheme results in one training instance for each input data, and reuse scheme results in two training instances for each input data. As such, when comparing between reuse and default, we use two times larger batch size for default. Unless otherwise stated, we train the model for 100000 steps and perform evaluation every 2000 steps.

The algorithms involve different hyperparameters. Especially, when the algorithm involves comparisons across different times, we need to determine the sampling scheme for t' given t. Inspired by Aggarwal et al. [1], we consider either using a local Gaussian proposal before wrapping things back to the range of t or sampling t' uniformly at random. We denote the standard deviation of the Gaussian proposal as σ_{time}, and use $\sigma_{time} = -1.0$ to denote the case where t' is sampled uniformly at random. Some algorithms involve adding local Gaussian perturbations to the positions, in which case we tune the standard deviation of the perturbation σ_{white} between $[0.01, 0.1, 1.0]$.

For tNCE, we tune σ_{time}. For tCNCE, we tune σ_{white}; under reuse scheme, as an ablation we additionally tune σ_{time}. We tune stNCE-w in terms of both σ_{time} and σ_{white}. stNCE-s and stNCE-o are tuned in terms of σ_{time}.

We generally select the best model based on val set results under a single random seed, and report test set results. For each job, we select the step that yields the best val set results and use the test set results at that step as the final metrics. Sometimes we would like to obtain uncertainty estimates, in which case we additionally train the models using the best configs with two additional random seeds.

Unless otherwise stated, we use Adam optimizer [46] with no weight decay.

E.1 Glossary of methods

We provide a compact glossary of methods in Table 4.

Table 4: Experimental variants. "self" denotes the space score estimates induced by the energy network, and "oracle" denotes the ground truth space scores. Appendix B presents additional discussion on the methods, and the training objective on the space scores is discussed in Appendix E.2.

Method	Training objective
tNCE	stNCE loss with perturbation kernel keeping x and perturbing t
tCNCE	stNCE loss with perturbation kernel keeping t and perturbing x
stNCE-m	stNCE loss combining an equal mixture of tNCE perturbation and tCNCE perturbation
stNCE-w	stNCE loss jointly perturbing x and t
stNCE-s	stNCE loss with forward-reverse kernel using detached self space scores
stNCE-o	stNCE loss with forward-reverse kernel using oracle space scores
RNE-s	RNE objective using self space scores
RNE-o	RNE objective using oracle space scores
tNCE+DSM	sum / average of tNCE and a loss on the space scores
stNCE-s+DSM	sum / average of stNCE-s loss and a loss on the space scores
CTSM	Conditional Time Score Matching loss as in Yu et al. [85] and Guth et al. [34]
DSM	Denoising Score Matching loss as in Song et al. [71] and Thornton et al. [75]
Dual SM	sum / average of CTSM loss and DSM loss

E.2 Losses on the scores

We may use loss functions on the scores induced by the model. For instance, we may combine tNCE or stNCE-s with a loss function on the space score.

For experiments apart from molecule, we combine an NCE loss $\mathcal{L}_{\text{NCE}}$ with a loss on the space score $\mathcal{L}_{\text{DSM}}$ to obtain the final loss $\mathcal{L}_{\text{total}}$ as

$$\mathcal{L}_{\text{total}} = 0.5 w_{\text{NCE}} \mathcal{L}_{\text{NCE}} + 0.5 \mathcal{L}_{\text{DSM}}, \tag{293}$$

where w_{NCE}, defaulting to 1, is an additional weight that can be tuned. In practice, we may implement $\mathcal{L}_{\text{DSM}}$ in two ways:

(1) As weighted denoising score matching loss similar to Song et al. [71] and Thornton et al. [75] as

$$s_\theta^{\text{space}}(x,t) = \nabla_x \log p_\theta(x \mid t), \tag{294}$$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{p(t)p(x_1)p_t(x|x_1)} \left[\beta_t^2 \left\| s_\theta^{\text{space}}(x,t) - \nabla_x \log p_t(x|x_1) \right\|^2 \right]. \tag{295}$$

(2) As a conditional flow matching loss with uniform weighting as in Lipman et al. [53], where the velocity estimates are obtained using the space score estimates obtained by differentiating the energy network, as described in Appendix D.4.

$$v_\theta(x,t) = \frac{\dot{\alpha}_t}{\alpha_t} x + \left(\frac{\dot{\alpha}_t \beta_t^2}{\alpha_t} - \dot{\beta}_t \beta_t \right) \nabla_x \log p_\theta(x|t), \tag{296}$$

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{p(t)p(x_1)p_t(x|x_1)} \left[\left\| v_\theta(x,t) - v(x|x_1) \right\|^2 \right] \tag{297}$$

Apart from the space score, we may also use CTSM loss [85; 34] to train the time score, given by

$$s_\theta^{\text{time}}(x,t) = \partial_t \log p_\theta(x|t), \tag{298}$$

$$w(t) = \frac{1}{\mathbb{E}_{p(x_1)p_t(x|x_1)} \left[\partial_t \log p_t(x|x_1)^2 \right]}, \tag{299}$$

$$\mathcal{L}_{\text{CTSM}} = \mathbb{E}_{p(t)p(x_1)p_t(x|x_1)} \left[w(t) \left\| s_\theta^{\text{time}}(x,t) - \partial_t \log p_t(x|x_1) \right\|^2 \right], \tag{300}$$

where we use CTSM weighting, which can be derived in closed-form for stochastic interpolants in the form of $x_t = \alpha_t x_1 + \beta_t x_0$, as in Yu et al. [85].

For both denoising score matching loss and CTSM loss, we may implement the weighting using the identity that $w(t)\|s-t\|^2 = \left\| \sqrt{w(t)} s - \sqrt{w(t)} t \right\|^2$ and prevent some singularities at the boundaries of time, following Tong et al. [76].

For Dual SM, based on Guth et al. [34], the noisy x and t are sampled based on the clean data. The final loss $\mathcal{L}_{\text{total}}$ is given as the weighted sum of CTSM loss $\mathcal{L}_{\text{CTSM}}$ with weighting function following Yu et al. [85] and Guth et al. [34] and DSM loss $\mathcal{L}_{\text{DSM}}$ with weighting function following Ho et al. [39] and Guth et al. [34]

$$\mathcal{L}_{\text{total}} = 0.5 \mathcal{L}_{\text{CTSM}} + 0.5 \mathcal{L}_{\text{DSM}}. \tag{301}$$

The sampling mechanisms of data points used for the DSM loss vary across `default` and `reuse`, and are described in Appendix D.1.

For molecule experiments, we adapt the implementation of Plainer et al. [62], where the final loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{NCE}} + \mathcal{L}_{\text{DSM}}, \tag{302}$$

and the data points for DSM loss are sampled following the `reuse` scheme described above for experiments apart from molecule.

E.3 Network architecture

Table 5 summarizes the model architectures for toy mixture, MNIST mixture, MNIST and ImageNet experiments; the networks for the other experiments are detailed in the corresponding sections. The network structure for toy mixture is based on Du et al. [24]. The networks for MNIST mixture, MNIST and ImageNet-64 are adapted from Lipman et al. [54]. Our preconditioning is described in Appendix D.3.

Given a U-Net with vector valued output, we consider two strategies to generate a scalar valued energy: (1) **sum**, directly summing the vector, (2) **inner product**, taking the inner product between

Table 5: Neural network architectures for toy mixture, MNIST mixture, MNIST and ImageNet-64. For U-Net model, the encoder and decoder residual block counts per level exclude the bottleneck, which has two additional ResBlocks surrounding a self attention block. Dropout applies to every U-Net Resblock.

Dataset	Backbone; channels	Residual blocks	Dropout	Energy parameterization
Toy mixture	MLP with residual layers; width 128, expansion factor 2	4 total	None	Final linear layer mapping to scalar
MNIST mixture	U-Net; $(32, 64, 64)$	2 encoder / 3 decoder per level	0	Preconditioned sum
MNIST	U-Net; $(64, 128, 128)$	2 encoder / 3 decoder per level	0.1	Preconditioned inner product with input, minus $z_\phi(t)$
ImageNet-64	U-Net; $(64, 128, 128)$	3 encoder / 4 decoder per level	0	Preconditioned inner product with input, minus $z_\phi(t)$

the network output and the input. $z_\phi(t)$ is a separate neural network with scalar valued output, and the final output energy is given by the output of the preconditioned U-Net minus the output of $z_\phi(t)$. Our parameterization is inspired by previous works [24; 75; 37; 60].

When using DSM alone to train EBMs on MNIST mixture, we disable the bias of the final U-Net convolution. The bias is otherwise enabled. This difference leads to negligible difference in the number of parameters.

E.4 Failure modes

Experimental settings We consider a simple setup where the reference distribution is a standard Gaussian and the data distribution is a Gaussian mixture:

$$p_0 = \mathcal{N}(0, 1), \quad p_1 = 0.5\,\mathcal{N}(\mu_1, 0.1^2) + 0.5\,\mathcal{N}(\mu_2, 0.1^2). \tag{303}$$

By varying (μ_1, μ_2), we control both the multimodality of p_1 and the mismatch between the high-density regions of p_0 and p_1, quantified in $[0, 1]$. We measure the error as $1 - R^2 \in [0, 1]$. Here, the Pearson R^2 measures the correlation between the ground truth and learnt energies of the data points. We evaluate three representative methods: tNCE (temporal differences), tCNCE (spatial differences), and stNCE-m (spatio-temporal).

Multimodality and mismatch scores We used heuristic scores to measure two properties of the data distribution p_1: its multi-modality and its mismatch with the reference distribution p_0. The multimodality score is computed as

$$\text{score}_{\text{multimodality}} = \frac{\mu_2 - \mu_1}{\mu_2 - \mu_1 + 2\sigma}, \in [0, 1]. \tag{304}$$

The mismatch score is computed as

$$\text{score}_{\text{mismatch}} = 1 - \frac{\max\big(0,\ \min(b, 1) - \max(a, -1)\big)}{b - a}, \tag{305}$$

where

$$a = \min(\mu_1 - \sigma,\ \mu_2 - \sigma), \quad b = \max(\mu_1 + \sigma,\ \mu_2 + \sigma). \tag{306}$$

To vary both these scores on an approximately uniform grid on $[0, 1] \times [0, 1]$, we set $\sigma = 0.1$ and use μ_1 and μ_2 from the grid shown in Table 6.

Table 6: The means of the two components of the one dimensional Gaussian mixture in the form of (μ_1, μ_2). Rows correspond to different mismatch scores, and columns correspond to different multimodality scores.

Mismatch score	Multimodality score				
	≈ 0	≈ 0.25	≈ 0.50	≈ 0.75	$[0.90, 0.96]$
≈ 0	$(-0.90, -0.90)$	$(0.79, 0.85)$	$(0.01, 0.21)$	$(0.01, 0.61)$	$(-0.91, 0.91)$
≈ 0.25	$(-0.95, -0.95)$	$(-0.97, -0.90)$	$(0.80, 1.00)$	$(-1.10, -0.50)$	$(-1.07, 1.42)$
≈ 0.5	$(1.00, 1.00)$	$(-1.03, -0.97)$	$(-1.10, -0.89)$	$(-1.30, -0.70)$	$(-2.29, 1.55)$
≈ 0.75	$(-1.05, -1.05)$	$(-1.10, -1.03)$	$(1.00, 1.20)$	$(-1.50, -0.90)$	$(-3.00, -0.37)$
≈ 1	$(-3.00, -3.00)$	$(-3.00, -2.93)$	$(-3.00, -2.80)$	$(-3.00, -2.39)$	$(-3.00, -1.07)$

Model and optimization The energy function is parameterized by a neural network that is a simple Multi-Layer Perceptron with 2 hidden layers and SiLU activations. Its weights are updated using the AdamW optimizer [55] with learning rate $1e-3$, weight decay $1e-4$, using a batch size of 256 over $1e4$ epochs. An exponential moving average of the weights is used, with a high parameter of 0.9999, to remove the residual variance in the loss. We empirically observed that the residual variance was high when the methods were expected to have a high estimation error, as is the case for example for tCNCE when the data distribution is multi-modal.

E.5 Toy mixture

We consider the problem of modeling a 20 mode Gaussian mixture in 10 dimensions, where the means of the Gaussian mixture are sampled from a standard normal distribution, and the covariances of the individual mixture components are given by $0.1^2 I$.

We use a batch size of 250 for default scheme and 125 for reuse scheme. We tune the learning rates between $[1e-4, 3e-4, 1e-3]$, and select the best model based on NormMSE on the val set.

E.6 MNIST mixture

E.6.1 Settings

We use a batch size of 512 for default scheme and 256 for reuse scheme. We fix the learning rate of all methods to $1e-4$, and select the best config based on NormMSE. For RNE, we tune Δt between $[1e-5, 1e-4, 1e-3]$, while setting λ, the weighting of the RNE regularization loss, to 1000, following He et al. [37].

E.6.2 Evaluations

Since we have access to both the ground truth data density and the sampling mechanism that generates samples from it, we estimate the log constant of the model density at the time corresponding to clean data using importance sampling, denoted as $\widehat{\log Z_1}$, and evaluate the model density normalized using $\widehat{\log Z_1}$.

$$\text{LogMSE} := \log\left(\mathbb{E}_{x\sim p_d(\cdot|1)}\left[\left(\log p_d(x \mid 1) - \log p_\theta(x \mid 1)\right)^2\right]\right), \tag{307}$$

$$\text{LogRatio} := \log\Big(\mathbb{E}_{x,x'\sim p_d(\cdot|1)^{\otimes 2}}\Big[\Big(\log p_d(x \mid 1) - \log p_d(x' \mid 1) \tag{308}$$

$$- \log p_\theta(x \mid 1) + \log p_\theta(x' \mid 1)\Big)^2\Big]\Big), \tag{309}$$

$$\text{LogNormMSE} := \log\left(\mathbb{E}_{x\sim p_d(\cdot|1)}\left[\left(\log p_d(x \mid 1) - \log p_\theta(x \mid 1) + \widehat{\log Z_1}\right)^2\right]\right), \tag{310}$$

$$\text{NormNLL} := \mathbb{E}_{x\sim p_d(\cdot|1)}\left[-\log p_\theta(x \mid 1) + \widehat{\log Z_1}\right]. \tag{311}$$

The first is the log squared error between the data and model log densities. The second is the log squared error between *spatial* differences of the data and model log densities — this metric is well-suited for evaluating unnormalized densities given that it does not depend on normalizing constants.

The third and the fourth evaluate how good the model is as an unnormalized EBM, though naturally depends on the accuracy of the estimated $\widehat{\log Z_1}$.

All means and standard deviations are computed based on results under the log transformed space.

E.7 Images

We do *not* use the NormNLL as the evaluation metric for two reasons. First, related works [34] do not use it and we wish to replicate a fair comparison. Second, the idea behind NormNLL is to correct the approximate normalization of the model using a separate estimate of the normalizing constant. In Section 6.2, we could obtain a precise estimate because we knew the ground-truth distribution of the dataset. Here, we do not know the ground-truth distribution and the standard way to estimate the normalizing constant is using AIS, which in our preliminary experiments and in related literature [85] was very high-variance on image datasets.

E.7.1 MNIST

We always use `reuse` scheme, and employ a batch size of 256. We fix the learning rates of all methods to $1e-4$. For variants combining space score loss, we observe that using a smaller w_{NCE} typically leads to better performance, and tune it between 1.0, 0.1, 0.01 and 0.001. The best configurations are selected based bits-per-dimension on the val set. Especially for stNCE-s+DSM, we avoid the configurations that show clear sign of overfitting to the train set as indicated by bits-per-dimension.

E.7.2 ImageNet

We use 50000 samples from the original training set as the val set. Same as Guth et al. [34], we use the original val set as our test set. We use reuse scheme with batch size 256. We additionally employ mixed precision with *bf16* to speed up training. The model is trained for 200000 steps. We use as learning rate $2e-4$ with $\sigma_{time}=0.01$ in the first 122000 steps, while changing the learning rate to $1e-4$ and σ_{time} to 0.001 afterwards. Here σ_{time} refers to the standard deviation used to perturb t; see σ as in Appendix D.1.1.

E.8 Molecules

E.8.1 Settings

In practice, we have access to energy and score function (*a.k.a.* force field) of the target distribution via physics, but they are expensive to evaluate as they may require solving a PDE. To avoid evaluating them for every new data point, there are two prevailing approaches in the literature.

The first approach is Machine Learning Force Fields (MLFFs), which regress a neural network against the energy and score functions on some finite grid of data points [41; 6]. The second approach which we follow here is Energy-based Models.

To form the dataset for training such models, a sampling process such as Langevin Dynamics (*a.k.a.* Molecular Dynamics), which requires evaluating the target energy and score functions, is run long enough so that the final samples are approximately *i.i.d.* samples from the target distribution.

Following Plainer et al. [62], we optimize the models using AdamW [55], where we directly use the models as employed by Plainer et al. [62]. Note that we do not employ the mixture version as in Plainer et al. [62].

We fix the weights of tNCE and stNCE-s loss to DSM loss as 1, and tune σ_{time} between $[-1.0, 0.1, 0.01]$; in some cases we experimented with the weights of NCE losses and $\sigma_{time}=0.001$, but did not observe notable differences.

When benchmarking the running time, for each method, we estimate the time per epoch by measuring the time for 10 epochs, following 2 epochs to warmup.

E.8.2 Evaluations

Given the histogram-based density estimates of the corresponding 2 dimensional ground-truth and model distributions as P_b and Q_b, we compute the resulting PMF error according to Durumeric et al. [25] and Plainer et al. [62]. Specificlly, let $M_b = \frac{P_b+Q_b}{2}$ and introduce $\tilde{P}_b$ and $\tilde{Q}_b$ as the masses with missing bins replaced by $\epsilon = 10^{-6}$. Define $w_b = \frac{\tilde{P}_b+\tilde{Q}_b}{\sum_c(\tilde{P}_c+\tilde{Q}_c)}$, then the PMF error is given by

$$\mathrm{PMF}(P,Q) = \sum_b w_b \left(\log \tilde{P}_b - \log \tilde{Q}_b\right)^2. \tag{312}$$

E.9 Compute resources

For failure modes experiments in Section 6.1, we run the experiments using Intel Xeon Gold 6230. Based on the saved logs and scripts, there are a total of 75 jobs, with each job uses 4 CPU cores, taking around half an hour. We could not verify whether these exact logs and scripts produced the results reported in the paper, but report them as they appear reasonable.

For synthetic Gaussian mixture experiments in Section F.1, we run the experiments using Intel Xeon Gold 6230. Apart from the jobs to generate the datasets which are generally fast, each job generally uses 4 CPU cores and takes roughly $1-2$ hours.

MNIST Gaussian mixture experiments in Section 6.2 and MNIST experiments in Section 6.3 are run using AMD MI250x GPU and AMD EPYC 7A53 CPU. For MNIST Gaussian mixture experiments, apart from some lightweight preprocessing jobs run using CPUs, each model is trained using 1 GPU, which is composed of 2 GCDs. Depending on the method, running times vary between 9 hours and 20 hours. For MNIST experiments, each model is trained using 2 GPUs, containing 4 GCDs. Across different methods, the running times vary between around 3 hours and around 8 hours. The long run for stNCE-s took around 19 hours.

ImageNet64 experiments are run using a single NVIDIA H200 GPU, which takes roughly 5 days.

For molecule experiments, for ALDP and ALDP full, the models are trained using a single NVIDIA V100 GPU. For Chignolin dataset, the models are trained using a single NVIDIA A100 GPU. Each job takes between several hours and tens of hours. For benchmarking the running time, we use a single NVIDIA A100; using the recorded time per epoch in seconds to estimate the time to run the benchmarks, it takes less than one minute to run the benchmark for ALDP and few minutes to run the benchmark for Chignolin.

We do not know the exact peak memory usage of each job. Preliminary and failed runs and analysis and visualization runs consumed additional compute.

E.10 License information

For experiments apart from Failure modes and molecules, our code is inspired by Song et al. [71] (Apache-2.0 license), and, notably, some parts are based on Lipman et al. [54] (CC BY-NC license) and Guth et al. [34] (BSD-3-Clause license). Our code for molecule experiment is based on Plainer et al. [62] (MIT license), which uses OpenMM [26] (MIT license and LGPL license). MNIST dataset (MIT) license, ImageNet64 dataset (non-commercial research and education) and Chignolin dataset (D.E. Shaw Research Trajectory Data License Agreement) are also employed. Our code is based on PyTorch [4] (custom license) or JAX [11] (Apache-2.0 license). Huggingface accelerate [33] (Apache-2.0 license) is used to perform multi-GPU training, and Weights & Biases [9] (MIT license) is used to monitor trainings.

F Additional experimental results

F.1 Synthetic Gaussian mixtures

The models are evaluated using the same evaluation metrics as discussed in Section E.6.2. The experimental results using the reuse scheme are shown in Table 7.

Table 7: Results on random gmm 20 0.1 1.0 with dim 10

Category	Method	MSE ↓	Ratio ↓	NormMSE ↓	NormNLL ↓	Time ↓
Temporal	tNCE	72.1 ± 48.78	28.67 ± 25.24	62.16 ± 70.56	-0.29 ± 4.15	1.78
Spatial	tCNCE	34.46 ± 18.92	10.7 ± 0.96	33.89 ± 3.16	-0.5 ± 0.27	**1.45**
Spatial-temporal	stNCE-m	8.83 ± 2.4	5.74 ± 0.79	5.69 ± 1.08	-4.16 ± 0.23	2.08*
	stNCE-w	27.72 ± 17.25	5.47 ± 0.98	4.79 ± 0.92	-4.4 ± 0.16	1.62
	stNCE-s	**4.29 ± 2.2**	**3.82 ± 0.45**	**3.16 ± 0.51**	**-4.73 ± 0.15**	1.87
Oracle	stNCE-o	**2.02 ± 0.34**	**3.51 ± 0.42**	**2.82 ± 0.36**	**-4.8 ± 0.1**	**1.47**

In Table 8 we additionally present results using the default scheme as a verification that the results are consistent across different sampling schemes.

Table 8: Results on random gmm 20 0.1 1.0 with dim 10

Category	Method	MSE ↓	Ratio ↓	NormMSE ↓	NormNLL ↓	Time ↓
Temporal	tNCE	51.11 ± 21.93	18.33 ± 1.9	32.74 ± 6.27	-1.03 ± 0.62	1.72
Spatial	tCNCE	35.24 ± 2.26	10.63 ± 0.84	34.55 ± 3.17	-0.44 ± 0.28	**1.42**
Spatial-temporal	stNCE-m	4.2 ± 0.33	7.29 ± 0.78	6.79 ± 0.84	-4.06 ± 0.16	1.95*
	stNCE-w	4.32 ± 1.29	5.67 ± 0.61	4.89 ± 0.38	-4.4 ± 0.13	1.86
	stNCE-s	**2.15 ± 0.38**	**3.63 ± 0.37**	**2.78 ± 0.45**	**-4.86 ± 0.12**	2.52
Oracle	stNCE-o	**2.05 ± 0.31**	**3.68 ± 0.74**	**2.85 ± 0.59**	**-4.84 ± 0.13**	**1.82**

We additionally verified that the results are highly consistent across different configurations of the Gaussian mixture, e.g. when the means and / or the covariances are scaled.

F.2 MNIST mixture

We report samples drawn using the model trained with stNCE-s and the model trained with DSM in Figure 6. When drawing the samples, we differentiate the energy network to obtain score estimates, form velocity estimates using the score estimates as described in Appendix D.4, and integrate the resulting ODE using 1000 steps of Euler integrations. While DSM results in better samples, as shown in Table 1 of main paper and further verified in Figure 7, DSM does not learn the correct energies.

In the main paper, we reported results where the NCE algorithms use the reuse scheme. We also report the results where the NCE algorithms use the default scheme in Table 9. The results are largely consistent regardless of the sampling scheme.

Perhaps surprisingly, stNCE-s generally outperforms stNCE-o. One possible and heuristic explanation is that stNCE-s adjusts the perturbation kernel as training progresses, thus helping optimization. Nevertheless, we do not have enough evidence to provide a strong explanation.

F.2.1 Analysis of trained models

We consider a temporal method (tNCE), a spatial method (tCNCE), and four different spatiotemporal methods (stNCE-m, stNCE-w, stNCE-o and stNCE-s). When applicable, we use the models trained with the standard deviations of the spatial and temporal Gaussian perturbation kernels set to 0.01;

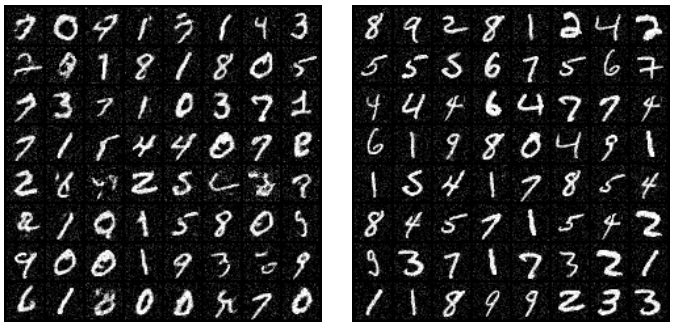


Figure 6: Left: samples drawn using model trained using stNCE-s; right: samples drawn using model trained using DSM.

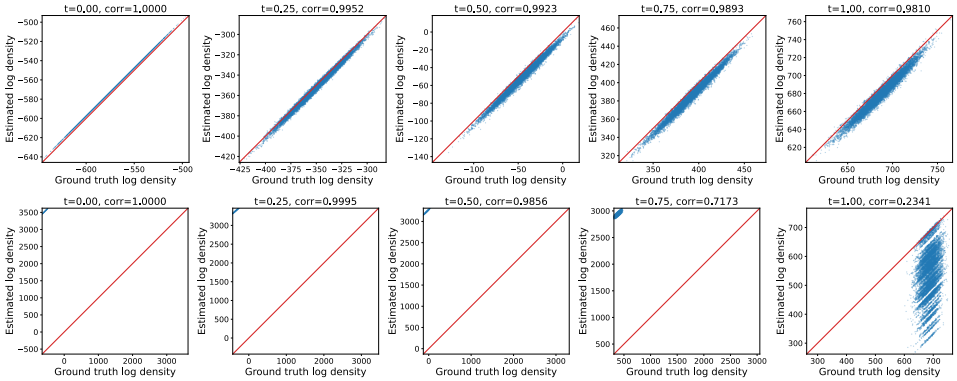


Figure 7: We generate density-density plots, where the model's learned energies are normalized using log constant estimated at time corresponding to data. $t = 0$ and $t = 1$ are approximated using the closest possible t that the models were trained on. Top: Density-density plots of model trained using stNCE-s across times. The learned densities are highly accurate. Bottom: Density-density plots of model trained using DSM across times. The trained model has time-varying log-constant, and the learned energies become highly inaccurate as t approaches 1, in which case the multimodality of the data distribution increases.

Table 9: Results on mnist mixture with dim 784 using default scheme. Reported times are average times per step in seconds. Results are largely consistent with reuse scheme.

Category	**Method**	**LogMSE** ↓	**LogRatio** ↓	**LogNormMSE** ↓	**NormNLL** ↓	**Time** ↓
Temporal	tNCE	13.31 ± 0.03	6.72 ± 0.11	8.89 ± 0.33	-604.45 ± 13.76	**0.4**
	CTSM	12.44 ± 0.45	7.14 ± 0.41	8.96 ± 0.24	-602.92 ± 9.59	0.68
Spatial	tCNCE	11.56 ± 0.59	7.58 ± 0.17	8.9 ± 0.08	-608.12 ± 2.46	**0.4**
	DSM	16.39 ± 0.07	9.31 ± 0.43	10.24 ± 0.39	-535.54 ± 28.81	0.7
Spatial-temporal	stNCE-m	10.55 ± 0.18	6.42 ± 0.36	7.72 ± 0.1	-644.25 ± 1.36	0.44*
	stNCE-w	10.09 ± 0.13	6.31 ± 0.41	7.59 ± 0.41	-646.07 ± 8.23	**0.4**
	stNCE-s	5.19 ± 0.11	3.64 ± 0.03	4.51 ± 0.1	-679.61 ± 0.47	0.47
	RNE-s	14.86 ± 0.31	7.75 ± 0.39	9.62 ± 0.05	-570.51 ± 1.09	0.44
	Dual SM	**3.73 ± 0.18**	**2.88 ± 0.06**	**3.25 ± 0.11**	**-683.96 ± 0.33**	0.7
Oracle	stNCE-o	4.99 ± 0.11	3.69 ± 0.06	4.95 ± 0.03	-677.07 ± 0.12	**0.32**
	RNE-o	**2.92 ± 0.08**	**2.06 ± 0.02**	**2.24 ± 0.07**	**-685.75 ± 0.09**	0.4

Table 10: Analysis of models trained on MNIST mixture. Unlike spatiotemporal methods, temporal and spatial methods encounter notable low density edges along the corresponding interpolation paths, and have consistently higher errors in such low density regions that high density ones.

method	HighDensityRMSE	LowDensityRMSE	LowDensityRate	LogNormMSE
tNCE	0.41	1.00	29.69%	9.22
tCNCE	5.79	27.54	57.47%	9.02
stNCE-m	0.44	0.51	1.19%	7.73
stNCE-w	0.43	0.53	1.19%	7.18
stNCE-o	0.20	0.19	1.19%	4.92
stNCE-s	0.18	0.14	1.19%	4.32

note that these do not necessarily correspond to their best configurations. We sample 100 data samples with one from each mode, so as to cover all the 100 modes of the Gaussian mixture. We randomly shift these data points and form 100 pairs of data samples, such that both the first elements and the second elements of the pairs cover all the modes and no pairs contain points from the same mode.

For each pair, we construct the interpolations as described in Figure 2 and Section 3 using 100 steps. The interpolation steps are indicated using k between 0 and K, adapting the notations from Section 3.

Along the interpolation, we classify a point (x, t) as having low density if $\min_j \frac{\|x - t\mu_j\|^2}{t^2(0.2)^2 + (1-t)^2} > \chi^2_{784}(0.99)$, and an edge $((x, t), (x', t'))$ as low density if either end point is low density; we remark that it is 0.2 instead of 0.1 due to rescaling of the data. We compute pointwise errors q_k and define the local ratio errors as $q_{k+1} - q_k$. The resulting ratio errors telescope across k when summed over all k, resulting in $q_K - q_0$.

Consider two data points, x and y. For the temporal method, we fix x, y and vary t. The pointwise error is given by

$$q_k = \left(\log p^\theta_{t_k}(x) - \log p^\theta_{t_k}(y)\right) - \left(\log p_{t_k}(x) - \log p_{t_k}(y)\right). \tag{313}$$

For spatial methods, we fix $t = t^*$, the time point corresponding to data, and linearly interpolate between x and y using z_k. The resulting pointwise errors are given by

$$q_k = \log p^\theta_{t^*}(z_k) - \log p_{t^*}(z_k). \tag{314}$$

For spatiotemporal methods, we sample a common standard normal noise ϵ. As t varies, x_k and y_k interpolate from ϵ to x and y. The pointwise errors are given by

$$q_k = \left(\log p^\theta_{t_k}(x_k) - \log p^\theta_{t_k}(y_k)\right) - \left(\log p_{t_k}(x_k) - \log p_{t_k}(y_k)\right). \tag{315}$$

Note that these expressions correspond to the expressions in Section 3.

We measure the proportions of low density edges along each interpolation and compare the models' errors in high and low density regions and report the results in Table 10. In the result table, **HighDensityRMSE** and **LowDensityRMSE** denote the RMSE of local ratio in high and low density regions, **LowDensityRate** denotes the proportion of edges classified as low density, and **LogNormMSE** refers to the same LogNormMSE as reported in the main paper.

As expected, tNCE and tCNCE, the temporal and spatial methods, frequently traverse low density regions and exhibit substantially larger errors in such regions than the high density ones. By contrast, spatiotemporal methods encounter few low density edges. This provides a fine-grained possible explanation regarding why the spatial and temporal methods fail on MNIST mixture dataset: their final performance is likely dominated by the estimation errors in low-density regions.

F.3 Images

F.3.1 MNIST

We draw samples by obtaining the spatial scores through differentiating the energy network, converting them into velocity estimates as in Appendix D.4, and integrate the resulting ODE with 1000 steps of Euler integrator. The samples as visualized in Figure 8 look realistic.

Table 11: Negative log likelihood (in bits/dimension) on MNIST test set. Result for Glow is taken from Máté et al. [57]. Results for FFJORD, i-ResNet, and MintNet are taken from Song et al. [68]. TSM result is taken from Yu et al. [85]. Reported times are average times per step in seconds.

Method	**Type**	**Single NFE**	**Time** $\downarrow$	**NLL** $\downarrow$
tNCE	Estimate	✓	0.127	7.01
FP-Diffusion [49]	Normalized	✗	-	2.98
TSM [20]	Estimate	✗	-	1.3
TRE [64]	Estimate	✗	-	1.09
i-ResNet [7]	Normalized	✗	-	1.06
Glow [47]	Normalized	✓	-	1.05
CTSM-v [85]	Estimate	✗	-	1.03
FFJORD [31]	Normalized	✗	-	0.99
MintNet [68]	Normalized	✓	-	0.98
Ours (stNCE-m)	Estimate	✓	0.17*	4.92
Ours (stNCE-w)	Estimate	✓	0.13	4.56
Ours (stNCE-s)	Estimate	✓	0.13	1.61
Ours (stNCE-s+DSM)	Estimate	✓	0.27	1.00



Figure 8: Samples drawn using stNCE-s+DSM.

Table 12: Test BPD of stNCE-s on MNIST. **Last** and **Best** indicates the test BPD at the step and at the step with the best validation BPD up to that step.

Step	**Last**	**Best**
100k	1.62	1.61
200k	1.49	1.48
300k	1.42	1.41
400k	1.37	1.36
500k	1.34	1.33

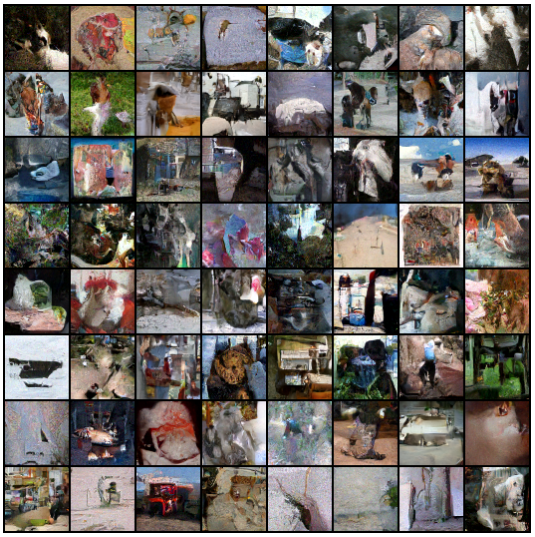


Figure 9: Samples drawn using 200 steps of Euler integrator with the trained model.

Table 13: Alanine dipeptide results. Time is time per epoch in seconds.

Method	Diffusion JS ↓	Langevin JS ↓	Diffusion PMF ↓	Langevin PMF ↓	Time ↓
tNCE	0.0093 ± 0.0001	0.0111 ± 0.0003	0.095 ± 0.001	0.119 ± 0.003	0.610
tNCE+DSM	0.0063 ± 0.0000	0.0065 ± 0.0001	0.063 ± 0.002	0.066 ± 0.003	0.780
stNCE-s	0.0098 ± 0.0002	0.0111 ± 0.0001	0.106 ± 0.004	0.139 ± 0.001	0.575
stNCE-s+DSM	0.0072 ± 0.0000	0.0082 ± 0.0002	0.076 ± 0.002	0.088 ± 0.005	0.648
FPE [62]	0.0082 ± 0.0002	0.0090 ± 0.0006	0.098 ± 0.003	0.104 ± 0.004	1.608

F.3.2 ImageNet

In order to validate the sampling capabilities of the model trained with stNCE-s+DSM on ImageNet-64, we form score estimates by differentiating through the energy network, form velocity estimates from the score estimates according to Appendix D.4 and draw samples using 200 steps of Euler integrator. Using a batch size of 512 on a single NVIDIA GH200, it takes less than a minute. We further visualize the samples in Figure 9, validating that the samples are largely reasonable.

F.4 Molecules

F.4.1 Additional results

In the main paper, we reported PMF and time per epoch. Here we additionally report JS. For Jensen-Shannon Divergence (JS), similar to PMF, we use histograms of the ground truth and generated samples in the 2 dimensional space, P_b and Q_b, and compute the JS as

$$\mathrm{JSD}(P, Q) = \frac{1}{2} \sum_b P_b \log \frac{P_b}{M_b} + \frac{1}{2} Q_b \log \frac{Q_b}{M_b}. \tag{316}$$

We report the numerical results in Table 13 and Table 14. JS largely follows the same trend as PMF.

We report the samples drawn using diffusion sampler and molecular dynamic simulations based on model trained using stNCE-s+DSM on Chignolin in Figure 10. The samples agree well with each other, highlighting the quality of the learned energies.

Table 14: Chignolin results. Time is time per epoch in seconds.

Method	Diffusion JS ↓	Langevin JS ↓	Diffusion PMF ↓	Langevin PMF ↓	Time ↓
tNCE	0.0593 ± 0.0001	0.2420 ± 0.0020	0.654 ± 0.001	3.143 ± 0.031	2.343
tNCE+DSM	0.0073 ± 0.0019	0.0181 ± 0.0055	0.060 ± 0.015	0.162 ± 0.053	2.866
stNCE-s	0.0097 ± 0.0000	0.0199 ± 0.0015	0.083 ± 0.000	0.170 ± 0.014	2.103
stNCE-s+DSM	0.0074 ± 0.0001	0.0082 ± 0.0006	0.059 ± 0.001	0.065 ± 0.005	2.349
FPE [62]	0.0048 ± 0.0001	0.0050 ± 0.0001	0.037 ± 0.000	0.039 ± 0.001	6.462

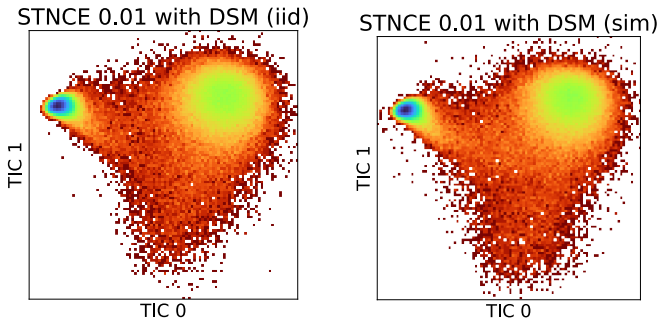


Figure 10: Samples drawn using diffusion sampler (left) and molecular dynamic simulations (right) with the model trained using stNCE-s+DSM. The samples are highly consistent, verifying the correctness of the learned energy-based diffusion model.

F.4.2 Higher dimensional problems

As presented in the main paper, in general, on alanine dipeptide tNCE outperforms stNCE-s either with or without DSM, while both outperfrom FPE, the previous SOTA solution, when combined with DSM. On chignolin stNCE-s outperforms tNCE, both with and without DSM.

We next list three molecule datasets along with their dimensionalities:

- alanine dipeptide (aldp), the coarse-grained version, has 15 dimensions obtained from 5 retained atoms,
- chignolin, has 30 dimensions,
- the full version of alainine dipeptide (aldp full), the fine-grained version, has 66 dimensions obtained from 22 atoms.

We hypothesize that the dimensionality of the dataset contributes to the relative performance of the different algorithms:

- temporal methods fail in high dimensions, because the support mismatch between the data and the noise generally grows larger as the dimensionality increases,
- spatiotemporal methods can work in high dimensions when they follow high-density regions as illustrated in blue in the left panel of Figure 1. This is especially the case for density-informed perturbation kernels, e.g. stNCE-s, as illustrated in Figure 3.

We similarly hypothesize that spatial methods also fail in high dimensions due to similar reason. In low dimensions and when data and noise have sufficient overlapping support, we do not claim that spatiotemporal methods would uniformly outperform temporal methods.

The empirical evidence indeed supports the above hypothesis. The molecule datasets are normalized before feeding into the network, as such the dimensionality is arguably the most important factor influencing the performance, where we would expect stNCE-s to outperform tNCE on aldp full given its larger dimensionality. Indeed, as verified in Table 15, stNCE-s notably outperforms tNCE and tCNCE, though aldp was obtained as taking a subset of coordinates from aldp full.

Table 15: Alanine dipeptide (full version) results. stNCE-s notably outperforms tNCE and tCNCE, validating the hypothesized improvement of spatiotemporal methods based on density-informed forward-reverse kernel over simpler temporal and spatial methods.

Method	IID JS ↓	Langevin JS ↓	IID PMF ↓	Langevin PMF ↓
tNCE	0.1023 ± 0.0002	0.2757 ± 0.0008	1.681 ± 0.010	5.952 ± 0.037
tCNCE	0.0891 ± 0.0001	0.3829 ± 0.0024	1.423 ± 0.005	11.838 ± 0.108
stNCE-s	0.0479 ± 0.0003	0.0635 ± 0.0005	0.678 ± 0.011	0.708 ± 0.011

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide a separate *Contributions* paragraph in Introduction. These claims were later verified either empirically or mathematically.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a separate *Limitations and future directions* paragraph in Discussion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For the theoretical results, we provide the full set of assumptions and proofs, possibly in the Appendix, notably Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide experimental details in the main paper or the Appendix, notably Appendix E, and provide code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide our code along with instructions to obtain the relevant datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details mainly in the Appendix, especially Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For experiments other than MNIST and ImageNet, we provide error bars as computed across multiple independent runs. For MNIST and ImageNet, following convention in generative modeling, we directly report results obtained in a single run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on the employed compute resources for the experiments in Appendix E.9.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: In our research we conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix C.5.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The trained EBMs could be used as e.g. image generators. We attempt to safeguard this by advocating for responsible usages of such trained models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide the license information in Appendix E.10.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the code and instructions on how to use the code to reproduce our results.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods does not involve LLMs as any important, original or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.