Iterated Denoising Energy Matching for Sampling from Boltzmann Densities

Tara Akhound-Sadegh [*1 2 3] **Jarrid Rector-Brooks** [*1 3 4] **Avishek Joey Bose** [*1 3 5] **Sarthak Mittal** [1 4]
Pablo Lemos [1 3 4 6 7] **Cheng-Hao Liu** [1 2 3] **Marcin Sendera** [1 4 8] **Siamak Ravanbakhsh** [1 2 9] **Gauthier Gidel** [1 4 9]
Yoshua Bengio [1 4 9] **Nikolay Malkin** [1 4] **Alexander Tong** [1 3 4]

Abstract

Efficiently generating statistically independent samples from an unnormalized probability distribution, such as equilibrium samples of many-body systems, is a foundational problem in science. In this paper, we propose ITERATED DENOISING ENERGY MATCHING (iDEM), an iterative algorithm that uses a novel stochastic score matching objective leveraging solely the energy function and its gradient—and no data samples—to train a diffusion-based sampler. Specifically, iDEM alternates between (I) sampling regions of high model density from a diffusion-based sampler and (II) using these samples in our stochastic matching objective to further improve the sampler. iDEM is scalable to high dimensions as the inner matching objective, is *simulation-free*, and requires no MCMC samples. Moreover, by leveraging the fast mode mixing behavior of diffusion, iDEM smooths out the energy landscape enabling efficient exploration and learning of an amortized sampler. We evaluate iDEM on a suite of tasks ranging from standard synthetic energy functions to invariant n-body particle systems. We show that the proposed approach achieves state-of-the-art performance on all metrics and trains $2-5\times$ faster, which allows it to be the first method to train using energy on the challenging 55-particle Lennard-Jones system.

1. Introduction

A fundamental task in probabilistic inference is drawing samples from an unnormalized probability density. Computational approaches have been employed to tackle this significant problem, yielding a multitude of applications across various scientific domains, such as spin-lattice states (Li & Wang, 2018; Nicoli et al., 2020), nuclear physics (Albergo et al., 2019), and proteins (Jumper et al., 2021; Bose et al., 2024). In this work, we focus on sampling from the (target) equilibrium distribution μ_{target} of many-body systems, *e.g.*, molecules, with density proportional to a Boltzmann-type distribution $\mu_{\text{target}}(x) \propto \exp(-\mathcal{E}(x))$ with the specific goal of efficiently drawing samples that cover all modes of the complex and dimensionless energy $\mathcal{E}$. Moreover, the density μ_{target} associated with the system is symmetric as the energy $\mathcal{E}$ function is invariant to rotations, reflections, and permutations of particles in 3D space.

Unlike the typical machine learning settings where the starting point is a dataset, learning to sample in many of these scientific settings is especially challenging as we often have little to no initial samples from μ_{target}, or the

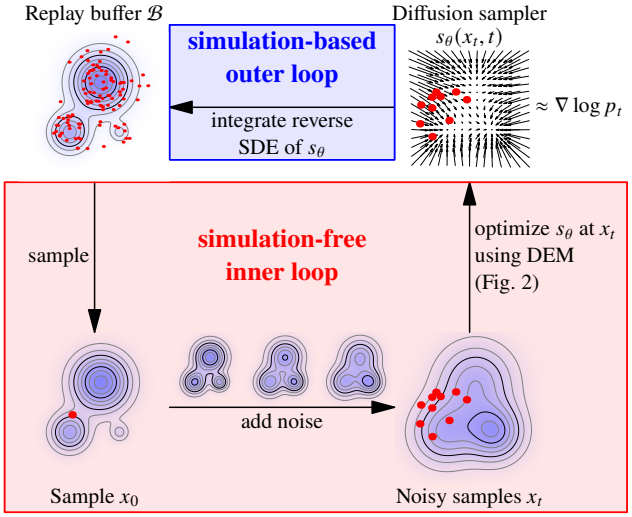

Figure 1. iDEM fits a diffusion sampler to a target distribution given by an unnormalized density. In the outer loop, iDEM populates a buffer with samples from the current model s_θ. In the inner loop, iDEM uses the DEM objective (§3.1) to regress s_θ to an estimate of the score at noised samples from the buffer. The inner loop is simulation-free, *i.e.*, requires no SDE integration.

[*]Equal contribution [1]Mila – Québec AI Institute [2]McGill University [3]Dreamfold [4]Université de Montréal [5]University of Oxford [6]Ciela Institute [7]Center for Computational Astrophysics, Flatiron Institute [8]Jagiellonian University [9]CIFAR. Correspondence to: <{jarrid.rector-brooks,tara.akhoundsadegh}@mila.quebec>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

given samples only cover a small set of modes (Noé et al., 2019). Acquiring additional high-quality samples can be achieved by leveraging Monte Carlo (MC) techniques such as Annealed Importance Sampling (AIS; Neal, 2001) or Sequential Monte Carlo (SMC; Del Moral et al., 2006) or simulating the actual (molecular) dynamics (MD; Leimkuhler & Matthews, 2013). Unfortunately, MC techniques and MD are computationally expensive with poor scaling to high dimensions inhibiting their easy application in complex high-dimensional physical systems.

The lack of sufficient data in sampling from Boltzmann-type distributions also precludes training deep generative models, q_θ, to match μ_{target} in the classical sense by maximizing the likelihood—*i.e.*, minimizing $\text{KL}(\mu_{\text{target}}||q_\theta)$. An alternative to Markov Chain Monte Carlo (MCMC), SMC, and MD is to consider variational approaches where q_θ is optimized using a metric where the samples are drawn under the *model* rather than the data. A convenient choice is the reverse-KL divergence, $\text{KL}(q_\theta||\mu_{\text{target}})$, but such a discrepancy measure suffers from mode-seeking behavior and is incapable of exploring the entire energy landscape. In scientific applications, one avenue is to leverage exact likelihood-based generative models—*e.g.*, normalizing flows (Rezende & Mohamed, 2015; Dinh et al., 2017)—trained using a combination of the forward and reverse KL to approximate μ_{target} and use importance sampling weights to correct for modelling errors (Noé et al., 2019; Midgley et al., 2023b). Despite the popularity of this approach, termed Boltzmann generators, the efficacy of the sampled points under the target is underpinned by the quality of initial samples from MCMC or MD, the expressive power of the class of flows, as well as the fidelity of the importance sampling estimator.

Given the complexity of physical processes, purely learning-based neural samplers such as the Path Integral Sampler (PIS; Zhang & Chen, 2022), Time-reversed Diffusion Sampler (DIS; Berner et al., 2022), and Denoising Diffusion Sampler (DDS; Vargas et al., 2023) are attractive substitutes for Boltzmann generators as they can amortize MCMC, and learn in the absence of data. Despite this, at present all neural samplers must *simulate* expensive forward and reverse trajectories and their gradients during learning which prevents their use when scaling to large-scale scientific applications. Thus, the search for an improved neural sampler motivates the following research question:

> *Can we find a scalable sampler that can learn from $\mathcal{E}(x)$ and $\nabla\mathcal{E}$ while achieving high mode-coverage of μ_{target}?*

Present work. In this paper, we propose ITERATED DENOISING ENERGY MATCHING (iDEM) a neural sampler based on denoising diffusion models for sampling from a Boltzmann distribution with a known energy function. iDEM is not only computationally tractable (Tab. 1), but also provides a good coverage of all modes of the distribution. In addition, iDEM can readily be imbued with any symmetries that manifest as invariances in $\mathcal{E}(x)$ making it well suited for scientific applications. Furthermore, in stark contrast to methods using MCMC, variational objectives, AIS, FAB, and SMC (Noé et al., 2019; Midgley et al., 2023b; Matthews et al., 2022) iDEM uses diffusion sampled data from the model mixed with an exploratory off-policy scheme to avoid the need for samples from μ_{target} while providing the option of using existing data.

Our proposed approach iDEM is structured as a bi-level algorithm in which the inner loop iteratively updates a diffusion sampler using a novel *simulation-free* stochastic regression objective directly on the energy function $\mathcal{E}(x)$. The outer loop of iDEM uses simulation of the reverse SDE of the updated (iterated) diffusion sampler and serves two important goals: 1.) it amortizes sampling—imitating a well-mixed MCMC chain as training progresses and 2.) it enables efficient exploration of the energy landscape as the inner loop updates push the model closer to matching $\mathcal{E}$, allowing us to sample closer to the true energy. Intuitively, as depicted in Fig. 1, iDEM takes inspiration from denoising objectives popularized in conventional diffusion models (Ho et al., 2020) and constructs a forward Gaussian process that adds noise in the (energy) function space until we reach the unnormalized log probability of a standard Normal distribution. By smoothing the energy directly using diffusion, iDEM builds upon important theoretical benefits such as fast-mixing times in high dimensions (De Bortoli et al., 2021). Reaching all modes during inner loop updates provides an informative learning target for the iterated diffusion sampler, whose reverse process learns to then transport particles from low to high-density regions under μ_{target}.

We test the empirical caliber of iDEM by conducting a range of experiments on synthetic Gaussian mixtures as well as $\text{SE}(3) \times \mathbb{S}_n$-invariant double-well and Lennard-Jones potentials associated to n-body particle systems with DW-4, LJ-13, and LJ-55 (Köhler et al., 2020; Klein et al., 2023b). We empirically find that iDEM achieves performance which is competitive and often exceeds previous state-of-the-art approaches in FAB (Midgley et al., 2023b) and all neural sampler baselines (Zhang & Chen, 2022; Vargas et al., 2023). Importantly, the performance of iDEM is achieved at a fraction of the training and memory cost of previous approaches which enables **iDEM to be the *first* method to successfully scale to LJ-55** using energy-based training.

2. Background and preliminaries

We are concerned with sampling problems in which we seek to draw samples from a target distribution μ_{target} over $\mathbb{R}^d$,

$$\mu_{\text{target}}(x) = \frac{\exp\left(-\mathcal{E}(x)\right)}{\mathcal{Z}}, \; \mathcal{Z} = \int_{\mathbb{R}^d} \exp\left(-\mathcal{E}(x)\right) dx.$$

Table 1. A comparison of approaches that are a) MCMC-free, b) are trained off-policy, c) require L forward simulation steps while training, and d) require backward gradients through time for d-dimensional samples. See §E for details and discussion.

Method	MCMC-free	Off-policy	Time	Memory
FAB (Midgley et al., 2023b)	✗	✓	$\mathcal{O}(L)$	$\mathcal{O}(L+d)$
PIS (Zhang & Chen, 2022)	✓	✗	$\mathcal{O}(L)$	$\mathcal{O}(Ld)$
DDS (Vargas et al., 2023)	✓	✗	$\mathcal{O}(L)$	$\mathcal{O}(Ld)$
pDEM (ours)	✓	✓	$\mathcal{O}(1)$	$\mathcal{O}(d)$
iDEM (ours)	✓	✓	$\mathcal{O}(L)$	$\mathcal{O}(d)$

The denominator $\mathcal{Z}$ is known as the *partition function* and is intractable for general energies $\mathcal{E}$. Consequently, we are unable to evaluate the exact density at a point $x \in \mathbb{R}^d$. Instead, we assume that we have access to the energy $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ and thus to the unnormalized probability density, $\mu_{\text{target}} \propto \exp(-\mathcal{E}(x))$. In scientific applications, such densities—modeled as *Boltzmann distributions*—can be used to express the probability of a system being in a particular state as a function of an energy function $\mathcal{E}(x)$. We next outline various standard approaches to sampling from μ_{target}.

2.1. Classical sampling methods

It is often the case we wish to compute expectations of some observable $f(x)$ by drawing samples from our distribution of interest $x \sim \mu_{\text{target}}$. If μ_{target} is an easy-to-sample distribution we could simply compute the Monte Carlo estimate which is the sample average. But, if μ_{target} is complex or not easy to sample from we must resort to alternative methods.

Importance sampling. By selecting an easy-to-sample from distribution $q(x)$ it is possible to construct a consistent estimator. We do so by drawing K independent samples $x^i \sim q(x), i \in [K]$ and computing the importance weights which is the ratio $w(x^i) = \exp(-\mathcal{E}(x^i))/q(x^i)$. This allows us to estimate the expectation of $f(x)$ under μ_{target} as:

$$\text{IS} := \mathbb{E}_{x\sim\mu_{\text{target}}}[f(x)] \approx \frac{\sum_k w(x^i) f(x^i)}{\sum_k w(x^i)}, \; x^i \sim q(x).$$

The optimal $q(x^i)$ is the one that minimizes the variance of the estimator and is roughly proportional to $f(x^i)\mu_{\text{target}}(x^i)$ (Owen, 2013). As a result, finding a good q in high dimensions or when μ_{target} is multimodal with separated modes is often challenging.

A detailed review of MCMC techniques is provided in §D.

2.2. Denoising diffusion

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) are probabilistic models whose generative process is the reverse of a tractably sampled stochastic process. Our iDEM borrows key modeling assumptions from diffusion, and we briefly review them here.

We denote p_t with $t \in [0, 1]$ the marginal distribution of the diffusion process which starts at $p_0 = \mu_{\text{target}}$ as a distribution over $\mathbb{R}^d$. In typical denoising diffusion, p_0 is a mixture of Dirac measures over the training dataset. We consider the stochastic differential equation (SDE),

$$dx_t = -\alpha(t)x_t\, dt + g(t)\, dw_t, \tag{1}$$

where w_t is a Brownian motion and α and g are functions of time. This SDE is known as the *forward (noising) process* which progressively adds noise starting from data $x_0 \sim p_0$ and runs over an interval $t \in [0, 1]$. Common choices for the decay rate α include $\alpha(t) = 0$ (variance-exploding (VE)) and $\alpha(t) = \frac{g(t)^2}{2}$ (variance-preserving (VP)).

The marginal distribution of the process (1) at time t is denoted p_t and has a smooth density for $t > 0$ under mild assumptions on μ_{target}. The corresponding *reverse process* SDE with Brownian motion $\overline{w}_t$ associated with (1) is then

$$dx_t = [-\alpha(t)x_t - g(t)^2 \nabla \log p_t(x_t)]\, dt + g(t)\, d\overline{w}_t. \tag{2}$$

To use the reverse SDE as a generative model, it is necessary to estimate the (Stein) score function of the convolved data distribution, $\nabla \log p_t(x_t)$. Denoising diffusion models fit a neural network $s_\theta(x_t, t)$, to this score via a stochastic regression. To be precise, in the example of the VE SDE, the density p_t is recognized as a convolution:

$$p_t = p_0 * \mathcal{N}(0, \sigma_t^2), \quad \sigma_t^2 := \int_0^t g(s)^2\, ds, \tag{3}$$

from which one can derive that

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E}_{x_0\sim p(x_0|x_t)} \overbrace{\left[\frac{x_0 - x_t}{\sigma_t^2}\right]}^{=\nabla_{x_t} \log \mathcal{N}(x_t; x_0, \sigma_t^2)}, \tag{4}$$

where $p(x_0|x_t) \propto p(x_0)p(x_t|x_0) = p(x_0)\mathcal{N}(x_t; x_0, \sigma_t^2)$. This expression suggests a stochastic regression objective—called denoising score matching—for the estimated score:

$$\mathcal{L} = \mathbb{E}_{\substack{x_0\sim p_0(x_0) \\ x_t\sim\mathcal{N}(x_t;x_0,\sigma_t^2)}} \left\| \frac{x_0 - x_t}{\sigma_t^2} - s_\theta(x_t, t) \right\|^2. \tag{5}$$

The objective (5) requires sampling from p_0 to be tractable. In the next section, we will study the case where p_0 is not tractable but is a Boltzmann density with known energy $\mathcal{E}$.

3. Iterated Denoising Energy Matching

We now present iDEM, which is designed to sample from a distribution μ_{target}. From henceforth, we will interchangeably use μ_{target} and p_0 to refer to the target density at time $t = 0$ and set p_1 to denote a tractable prior at time $t = 1$. We assume that $\mathcal{E}$ is known and $\nabla\mathcal{E}$ is cheaply computable, but that $\mathcal{Z}$ is not, and thus exact sampling is not tractable.

We motivate the design of iDEM by first outlining the two principal challenges that inhibit the training of a diffusion sampler in the absence of data. **(C1)** The score function $\nabla \log p_t(x_t)$ is not available and **(C2)** we do not know where in the sample space to match the score.

To overcome these challenges iDEM is composed of two key algorithmic components organized in a bi-level iterative scheme. The inner loop tackles $(\mathbf{C1})$ by proposing DENOISING ENERGY MATCHING (DEM) a novel stochastic regression objective to the score using only the energy $\mathcal{E}$ and its gradient while the outer loop addresses $(\mathbf{C2})$ by proposing informative starting points x_0 which can then be diffused and used in the subsequent inner loop of the algorithm.

- **(C1) Inner Loop**. The sampler s_θ is trained to approximate the score of the target density convolved with varying levels of noise. Specifically, s_θ is updated using DEM (§3.1). In principle, we can optimize s_θ with respect to the DEM objective at any point in time t and point x_t, but an optimal training scheme would prudently select points x_t at which to train the score estimator. The 'off-policy' nature of the DEM objective allows flexibility in the choice of x_t.
- **(C2) Outer Loop**. For the DEM objective to provide a useful learning signal it is critical to select informative points x_t. While any sampling strategy is possible, including off-policy methods and MCMC, we make an algorithmic choice to utilize s_θ via its reverse SDE as an amortized sampler (§3.2), whose proposals enable fast exploration in high dimension. Iteratively updating s_θ in every inner loop phase synergistically improves the sampling quality of s_θ in the outer loop.

A complete description of iDEM is provided in Algorithm 1.

3.1. Denoising diffusion with a Boltzmann target (C1)

We consider the same noising process as in §2.2, given by an SDE of the form (1), but now with p_0 being a Boltzmann density. Recall from (2) that reversing the noising process requires the score function $\nabla \log p_t(x_t)$, where $p_t = p_0 * \mathcal{N}(0, \sigma_t^2)$. However, unlike in the case of an empirical data distribution p_0, we cannot tractably sample p_t or regress to its score. The main ideas of the stochastic regression objective in iDEM are (1) to estimate the score of p_t by Monte Carlo and (2) to regress a neural network estimator s_θ to this estimated score. We describe each idea in turn.

MC score estimation. We write the score of p_t as an expectation in a manner similar to (4), here for the VE SDE:

$$\nabla \log p_t(x_t) = \frac{\nabla \left(p_0 * \mathcal{N}(0, \sigma_t^2)\right)(x_t)}{p_t(x_t)}.$$

The key observation is that the gradient of the Gaussian convolution with p_0 can be done in a specific way that gives an avenue for efficient estimation as described below.

$$\begin{aligned}\nabla \log p_t(x_t) &= \frac{((\nabla p_0) * \mathcal{N}(0, \sigma_t^2))(x_t)}{p_t(x_t)} && (6)\\ &= \frac{\mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\nabla p_0(x_{0|t})]}{\mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[p_0(x_{0|t})]} \\ &= \frac{\mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\nabla \exp(-\mathcal{E}(x_{0|t}))]}{\mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\exp(-\mathcal{E}(x_{0|t}))]}, && (7)\end{aligned}$$

where (6) is by a standard property of convolutions and (7) uses that the normalizing factor $\frac{1}{Z}$ appears in both the numerator and denominator. Since, (7) is true for *any* x_t, it means that it can provide a training signal to learn the score function using samples that come from any distribution, not necessarily those associated with μ_{target}. This provides two principal advantages: simulation-free computation of the gradient, and off-policy training, which can be exploratory.

We note a connection between (6) and the score in the empirical case (4). In (4), the gradient is placed on the second term of the convolution, $\mathcal{N}(0, \sigma_t^2)$, which allows estimation when p_0 has no density but is tractable to sample. In (6), the gradient is instead placed on the first term, p_0, taking advantage of the fact that sampling from the normal distribution is feasible, while for p_0 sampling is not possible but we can compute a gradient. §C contains further discussion and connection with flow matching algorithms, stochastic control, and the recently proposed Reverse Diffusion Monte Carlo (Huang et al., 2024).

The expression (7) suggests a Monte Carlo estimator that uses the same set of samples from $\mathcal{N}(x_t, \sigma_t^2)$ to approximate the numerator and denominator. That is, we write

$$\begin{aligned}\nabla \log p_t(x_t) &\approx \frac{\frac{1}{K}\sum_i \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \\ &= \nabla_{x_t} \log \sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)})), && (8)\\ x_{0|t}^{(1)}, \ldots, x_{0|t}^{(K)} &\sim \mathcal{N}(x_t, \sigma_t^2),\end{aligned}$$

where in the second line $x_{0|t}^{(i)}$ is understood as a function of x_t via the reparametrization $x_{0|t}^{(i)} = x_t + \epsilon^{(i)}$, $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma_t^2)$ and the notation $0 \mid t$ indicates a sample at time $t = 0$ is drawn from a distribution centred at x_t. For numerical stability in low-density regions (8) is implemented using the LogSumExp trick.

The K-sample approximation in (8), which we denote $\mathcal{S}_K(x_t, t)$, can also be understood as an importance-

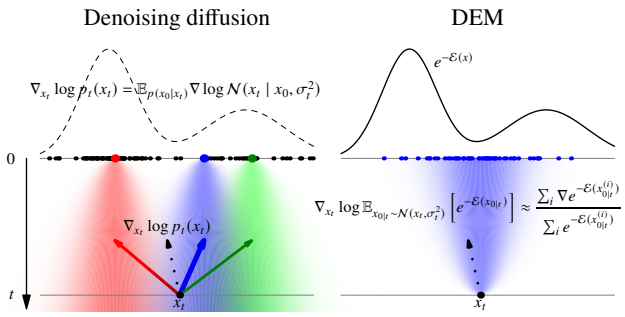

Figure 2. Two ways of estimating the score $\nabla \log p_t(x_t)$. **Left:** A diffusion model estimates the score convolved with noise by stochastically regressing to the scores of distributions conditioned on x_0—*i.e.*, points ●, ●, ●—weighted by the likelihood of $p(x_0|x_t)$ (indicated by the arrow thickness). This regression requires samples from μ_{target}. **Right:** DEM assumes an unnormalized density over x_0 and expresses the score of the convolved density as an expectation and regresses to a consistent estimator of this score.

weighted estimate over $p_0(x_{0|t})\mathcal{N}(x_{0|t}; x_t, \sigma_t^2)$ as follows,

$$\mathcal{S}_K(x_t, t) = -\sum_i w_i \nabla \mathcal{E}(x_{0|t}^{(i)}), \tag{9}$$

$$w_i := \frac{\exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\sum_j \exp(-\mathcal{E}(x_{0|t}^{(j)}))} \propto_i p_0(x_{0|t}^{(i)}).$$

This recalls the expectation over $p(x_0 \mid x_t)$ in (4). In addition, Fig. 2 visually illustrates the MC estimator in (8), which is distinguished from a classical diffusion objective.

The estimator $\mathcal{S}_K(x_t, t)$ is a consistent estimator and we characterize its bias with the following proposition.

Proposition 1. *If* $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ *and* $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$ *are sub-Gaussian, then, there exists a constant* $c(x_t)$ *such that with probability* $1-\delta$ *(over* $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2)$*) we have*

$$\|\mathcal{S}_K(x_t, t) - \nabla \log p_t(x_t)\| \leq \frac{c(x_t) \log\left(\frac{1}{\delta}\right)}{\sqrt{K}}.$$

We present all proofs in §A. Prop. 1 elucidates that the bias of $\mathcal{S}_K$ decays at a rate of $O(1/\sqrt{K})$. Note that this means that for regions with large values of $\mathbb{E}_{x_{0|t} \sim \mathcal{N}(x_t, \sigma_t^2)}[\exp(-\mathcal{E}(x_{0|t}))]$ we can obtain an accurate estimate for modest values of K. In contrast, for low-density regions we need K large such that $\frac{c(x_t)}{\sqrt{K}} \leq \mathbb{E}_{x_{0|t} \sim \mathcal{N}(x_t, \sigma_t^2)}[\exp(-\mathcal{E}(x_{0|t}))]$, which motivates the search for an informative starting sample x_0 and is the focus of §3.2. Finally, note that the sub-Gaussian assumption is relatively mild and all our energies and their gradient norms studied in this paper satisfy this property.

Regressing to the estimate. As in standard diffusion models, we aim to fit a neural network $s_\theta(x_t, t)$ with parameters θ to the score $\nabla \log p_t(x_t)$. This is achieved by minimizing the regression loss:

$$\mathcal{L}_{\text{DEM}}(x_t, t) := \|\mathcal{S}_K(x_t, t) - s_\theta(x_t, t)\|^2, \tag{10}$$

at a given point x_t and time t. As the estimator $\mathcal{S}_K$ is stochastic, the optimal solution for (10) in the space of all values for $s_\theta(x_t, t)$ is $s_\theta^*(x_t, t) = \mathbb{E}[\mathcal{S}_K(x_t, t)]$, which by Prop. 1, approaches the true score $\nabla \log p_t(x_t)$ as $K \to \infty$.

The objective (10) can be computed for a fixed x_t, and its global minimum in function space does not depend on the choice of t and x_t at which it is optimized (as long as the training distribution has full support). This property is in contrast to (5), in which x_t must be sampled from a distribution conditioned on a data point x_0. The flexibility in the choice of t and x_t in $\mathcal{L}_{\text{DEM}}$ allows "off-policy" methods that recycle points generated by past iterations of the model.

We also note that to construct DEM we made use of two Gaussian convolutions, namely $\mathcal{N}(0, \sigma_t^2)$ and $\mathcal{N}(x_t, \sigma_t^2)$. The first convolution is used to create a probability p_t from which we draw samples x_t. Sampling from p_t enables us to smooth the energy landscape via diffusion. The second convolution is used to construct the MC estimate of the score $\nabla \log p_t$, which is the regression target for s_θ.

Algorithm 1 ITERATED DENOISING ENERGY MATCHING

Input: Network s_θ, Batch size b, Noise schedule σ_t^2, Prior p_1, Num. integration steps L, Replay buffer $\mathcal{B}$, Max Buffer Size $|\mathcal{B}|$, Num. MC samples K.
while Outer-Loop **do**
  $\{x_1\}_{i=1}^b \sim p_1(x_1)$
  $\{x_0\}_{i=1}^b \leftarrow \texttt{sde\_int}(\{x_1\}_{i=1}^b, s_\theta, L)$ {Sample}
  $\mathcal{B} = (\mathcal{B} \cup \{x_0\}_{i=1}^b)$ {Update Buffer $\mathcal{B}$}
  while Inner-Loop **do**
    $x_0 \leftarrow \mathcal{B}.\texttt{sample()}$ {Uniform sampling from $\mathcal{B}$}
    $t \sim \mathcal{U}(0, 1)$, $x_t \sim \mathcal{N}(x_0, \sigma_t^2)$
    $\mathcal{L}_{\text{DEM}}(x_t, t) = \|\mathcal{S}_K(x_t, t) - s_\theta(x_t, t)\|^2$
    $\theta \leftarrow \texttt{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{DEM}})$
  end while
end while
output s_θ

3.2. Amortized sampling with a diffusion sampler (C2)

The DEM loss introduced in §3.1 serves as a useful learning target whenever a sample x_0 corresponds to a high value of $\exp(-\mathcal{E}(x_0))$. Specifically, constructing a stochastic regression objective starting from such an x_0 enables us to train s_θ such that reverse SDE can start from any point with a low value of $\mathcal{E}$ and reach a mode of $\mathcal{E}$. Thus, what remains is finding informative points to construct our DEM objective.

To find informative points we start by first noting that DEM can be used as an off-policy objective, which means that the

objective can be evaluated using any set of samples. Consequently, in problem settings where we have access to an initial dataset, *e.g.*, from MCMC or MD simulations, we can readily leverage them to warm start training of s_θ. In contrast, in settings with no initial samples, this feature is not possible. However, randomly exploring the sample space is unlikely to yield an informative x_0, especially in high dimensions where the energy landscape might be sparse.

We alleviate this cold-start problem by directly using our diffusion sampler s_θ. In particular, we use the reverse SDE associated with s_θ which enables us to start from a mass covering prior, *e.g.*, standard normal, and reach points that are progressively more informative samples. Note that we are free to choose a different diffusion coefficient $g(t)d\overline{w}_t$ in (2) to increase or decrease the amount of exploration when generating a point x_0. In addition, we can run each reverse SDE in parallel to produce a batch of samples that we store in a replay buffer $\mathcal{B}$. Buffer samples can then be used in the inner-loop; resembling a persistent contrastive-divergence objective (Tieleman, 2008) to train energy-based models.

We highlight that in this outer loop sampling phase s_θ is fixed—*i.e.*, the parameters θ are not updated—and as a result, despite simulation of the reverse SDE, iDEM is computationally cheap as we do not need to backpropagate gradients through the SDE solver. We note that making the algorithmic choice of sampling x_0 with the reverse s_θ iDEM can be viewed as a hybrid approach within the spectrum of on-policy to off-policy methods. This is due to the fact the forward SDE to get x_t differs from the reverse SDE with a modified diffusion coefficient used to populate x_0 in $\mathcal{B}$.

By training our diffusion sampler in every inner loop step we obtain an improved diffusion sampler. This in turn improves the fidelity of the batch of samples in the replay buffer $\mathcal{B}$ produced by s_θ. Thus, every pair of inner and outer loop operations in iDEM (see Algorithm 1) produces a new sampler that is iteratively retrained and new sampled points that populate the buffer. From this perspective, we can view the process of learning as obtaining a higher-quality amortized sampler—mimicking a fully mixed MCMC algorithm—after the completion of the inner loop. Importantly, this sampler can be used in the *absence* of any ground truth data and is the chief vehicle that allows iDEM to explore and find all salient modes of the energy function.

3.3. Incorporating symmetries in iDEM

Boltzmann-type distributions found in physical processes are beholden to the symmetries of the system. In this case, the symmetries arise from the spatial invariance of the energy function itself. More precisely, if we take n-body systems in $\mathbb{R}^d$, where $d = 3n$, the symmetries correspond to the rotation, translation, and permutation of the particles. These symmetries render the target density μ_{target} invariant to the product group $G = \mathrm{SE}(3) \times \mathbb{S}_n$.

If G is a subgroup of the orthogonal group $\mathrm{O}(n)$—*i.e.*, rotations and reflections—and carries an orthogonal action, then the gradient of a G-invariant function is G-equivariant (Papamakarios et al., 2021, Lemma 2). As a result, we have that if the energy function $\mathcal{E}$ is G-invariant, due to μ_{target} being G-invariant, the gradient $\nabla\mathcal{E}$ is G-equivariant. We can extend this result to the product group $\mathrm{SE}(3) \times \mathbb{S}_n$ by embedding this in $\mathrm{O}(3n)$ by first projecting to a translation invariant subspace and defining an extended action[1] that acts orthogonally in $\mathbb{R}^{3n}$. Invoking the $\mathrm{SE}(3) \times \mathbb{S}_n$ symmetry constraint in the DEM objective leads to the following proposition:

Proposition 2. *Let G be the product group $\mathrm{SE}(3) \times \mathbb{S}_n \hookrightarrow O(3n)$ and p_0 be a G-invariant density in $\mathbb{R}^d$. Then the Monte Carlo score estimator of $\mathcal{S}_K(x_t, t)$, is G-equivariant if the sampling distribution $x_{0|t} \sim \overline{\mathcal{N}}(x_{0|t}; x_t, \sigma_t^2)$ is G-invariant, i.e., $\overline{\mathcal{N}}(x_{0|t}; g \circ x_t, \sigma_t^2) = \overline{\mathcal{N}}(g^{-1}x_{0|t}; x_t, \sigma_t^2)$.*

In practice, we can easily implement an equivariant $\mathcal{S}_K(x_t, t)$ by replacing the standard normal distribution with a normal distribution that has zero center of mass. Note that a standard normal is already rotation and permutation invariant and an orthogonal action induces no change in volume as the determinant is 1. Whilst it is not possible to define a translation invariant measure on $\mathbb{R}^d$ we can still achieve this symmetry by constructing a normal distribution that has zero center of mass. Intuitively, this is a projection of the density in $\mathbb{R}^d$ to $\mathbb{R}^{d-1}$ and this subspace is translation invariant (Köhler et al., 2020; Garcia Satorras et al., 2021; Midgley et al., 2023a). Finally, to use symmetries within our iDEM algorithm we also need the diffusion sampler s_θ to be equivariant to the product group. While this design choice is necessary it does not come with any loss of generality as there always exists an equivariant map between two group invariant distributions on $\mathbb{R}^d$ (Bose et al., 2021).

4. Experimental results

We evaluate iDEM on multiple unnormalized densities including synthetic and $\mathrm{SE}(3) \times \mathbb{S}_n$-equivariant n-body particle systems of varying complexity as examples of scientific applications[2]. For our metrics, in Table 2, we report both sample-based metrics, such as 2-Wasserstein that assesses mode coverage, and Effective Sample Size (ESS), as well as the standard negative log-likelihood (NLL). We report additional metrics such as log partition function ($\log Z$) and Total Variation (TV) distance in Table 5 in §G.1

Datasets. We evaluate iDEM on four datasets, a 40-Gaussian mixture model (GMM), and three equivariant

[1] $\mathrm{O}(3) \times \mathbb{S}_n$ action on $\mathbb{R}^{3n}$ is such that $\mathrm{O}(3)$ acts diagonally while $\mathbb{S}_n$ acts by an orthogonal permutation matrix on the particles.

[2] Code for iDEM is available at https://github.com/jarridrb/dem.

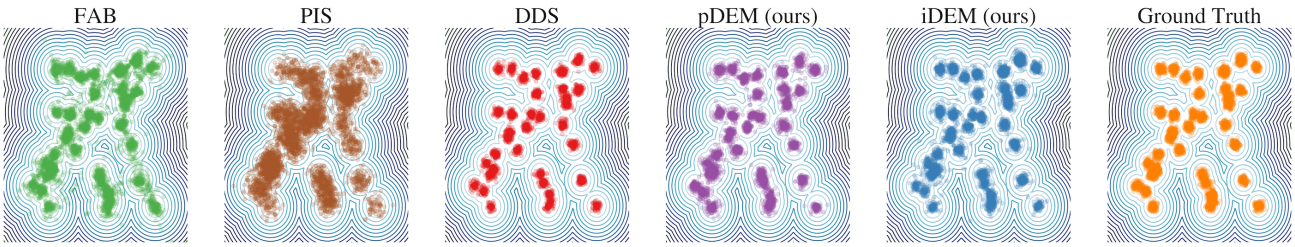

Figure 3. Contour lines for the target distribution, which is a GMM with 40 modes. Colored points represent samples from each method.

potentials: A 4-particle double-well potential (DW-4), a 13-particle Lennard-Jones potential (LJ-13), and a 55-particle Lennard-Jones potential (LJ-55) (see §F.4 for details). These benchmark datasets are chosen to demonstrate how scaling dimension d affects algorithms, and due to their use in scientific applications (Köhler et al., 2020; Klein et al., 2023b).

Baselines. We compare iDEM to three recent works: the path integral sampler (PIS) (Zhang & Chen, 2022), denoising diffusion sampler (DDS) (Vargas et al., 2023), and flow annealed bootstrapping (FAB) (Midgley et al., 2023b). PIS and DDS are the most comparable models to iDEM as they are both diffusion-based but require simulating trajectories to evaluate their objective. On the other hand, FAB is the current state-of-the-art approach that combines AIS samples and (equivariant) normalizing flow training within a buffer. We also include prior-DEM (pDEM) which fills the buffer with the $\mathrm{SE}(3) \times \mathbb{S}_n$-invariant prior.

Architecture. For iDEM, PIS, and DDS, we can use any network $s_\theta : (\mathbb{R}^d, \mathbb{R}^+) \to \mathbb{R}^d$. We use an MLP with sinusoidal positional embeddings for the GMM and an EGNN flow model architecture (Satorras et al., 2021) for the equivariant densities (DW-4, LJ-13, and LJ-55) following Klein et al. (2023b). FAB, however, requires a specialized invertible architecture, so we use the architecture from Midgley et al. (2023b) for GMM and $\mathrm{SE}(3)$-augmented coupling flow architecture from Midgley et al. (2023a) for the equivariant tasks. Finally, in our parametrization of iDEM, we sometimes find it useful to pin the score at $t = 0$ to $\nabla \log p_0(x_0)$ as we have access to it and do not need to estimate it with MC—*i.e.*, $\mathcal{S}_K(x_0, 0)$. We provide further details on the experimental setup in §F.1.

4.1. Main results

We report the sample likelihood-based metrics in Tab. 2. For a fair comparison between iDEM and all baselines—some of which cannot readily provide an NLL—we fit an optimal transport conditional flow matching (OT-CFM) model (Tong et al., 2023) on generated samples from each method. We use the reverse ODE of this OT-CFM model to compute a test NLL which is presented in Tab. 2. We find that iDEM outperforms all considered baselines on $\mathcal{W}_2^2$

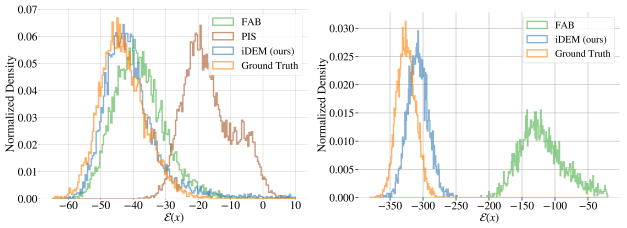

Figure 4. Comparison of the ground truth energy histograms of LJ-13 (left) and LJ-55 (right) and energies of samples generated from various methods. DDS is omitted from both plots while PIS is omitted from LJ-55 as they diverge in these settings.

and TV indicating high-quality generated samples. This result is also qualitatively substantiated in Fig. 3 for GMMs where we notice iDEM and DDS obtain the best samples.

For NLL we find that iDEM matches or outperforms all baselines on GMM, DW-4, and LJ-13. Importantly, on the most challenging and high-dimensional energy LJ-55, unlike iDEM which obtains the best NLL, PIS and DDS experience unstable training and cannot learn successfully on the task. Thus, We reconcile this by noting that LJ-55 has an energy with a high Lipschitz constant (see §F.4.3) and without smoothing represents a significant modeling challenge.

In Fig. 4 we visualize the energy histograms of the LJ-13 and LJ-55 systems (see §G for DW-4) in comparison to model samples. We also report inter-atomic distances for samples from the training data and all models in §G.2. We observe that iDEM is the best approach in terms of accurately modeling the true energy of each system with a significant separation between iDEM and FAB, the second best approach, on the LJ-55 energy. Notably, methods such as DDS and PIS are unable to train properly on this task.

Computational complexity. We quantify the computational footprint of each method by reporting training time in hours to convergence in Tab. 3. We find that iDEM is significantly faster than the previous SOTA FAB on all tasks due to FAB being bottlenecked by AIS. In particular, iDEM is $\sim \mathbf{4}\times$ faster on high dimensional tasks LJ-13 and LJ-55 while $\sim \mathbf{1.8}\times$ faster on the lower dimensional, less computationally expensive GMM and DW-4 tasks. Furthermore, iDEM is also faster than neural

Table 2. Sampler performance with mean ± standard deviation over 3 seeds for negative log-likelihood (NLL), Effective Sample Size (ESS), and 2-Wasserstein metrics ($\mathcal{W}_2$). $*$ indicates divergent training. **Bold** via Welch's two sample t-test $p < 0.1$. See §F.2 for more details.

Energy →	GMM ($d = 2$)			DW-4 ($d = 8$)			LJ-13 ($d = 39$)			LJ-55 ($d = 165$)		
Algorithm ↓	NLL	ESS	$\mathcal{W}_2$	NLL	ESS	$\mathcal{W}_2$	NLL	ESS	$\mathcal{W}_2$	NLL	ESS	$\mathcal{W}_2$
FAB (Midgley et al., 2023b)	7.14±0.01	0.653 ±0.017	12.0±5.73	**7.16**±0.01	**0.947** ±0.007	2.15±0.02	**17.52**±0.17	0.101 ±0.059	4.35±0.01	200.32±62.3	0.063 ±0.001	18.03±1.21
PIS (Zhang & Chen, 2022)	7.72±0.03	0.295 ±0.018	**7.64**±0.92	7.19±0.01	0.901 ±0.003	**2.13**±0.02	47.05±12.46	0.004 ±0.002	4.67±0.11	*	*	*
DDS (Vargas et al., 2023)	7.43±0.46	0.687 ±0.208	9.31±0.82	11.27±1.24	0.408 ±0.001	2.15±0.04	*	*	*	*	*	*
pDEM (ours)	7.10±0.02	0.634 ±0.084	12.20±0.14	7.44±0.05	0.547 ±0.010	**2.11**±0.03	18.80±0.48	0.044 ±0.013	**4.21**±0.06	*	*	*
iDEM (ours)	**6.96**±0.07	**0.734** ±0.092	**7.42**±3.44	**7.17**±0.00	0.825 ±0.002	**2.13**±0.04	**17.68**±0.14	**0.231** ±0.005	**4.26**±0.03	**125.86**±18.03	**0.106** ±0.022	**16.128**±0.071

Table 3. Training time results in hours excluding evaluation time. $*$ denotes divergent training runs.

Algorithm ↓ Dataset →	GMM	DW-4	LJ-13	LJ-55
FAB (Midgley et al., 2023b)	1.71	6.87	21.78	40.35
PIS (Zhang & Chen, 2022)	4.11	11.29	17.36	*
DDS (Vargas et al., 2023)	1.81	5.65	*	*
pDEM (ours)	0.36	1.40	1.79	*
iDEM (ours)	0.87	4.30	6.55	7.75

sampler baselines like PIS and DDS, which we attribute to the simulation-free gradients in our DEM objective. Finally, we observe that training times for pDEM are significantly smaller than all other methods due to it being truly simulation-free. However, this comes at the cost of training stability—2 of 3 pDEM runs diverged on LJ-55.

4.2. Ablation experiments

We next investigate different aspects of the iDEM in a set of ablation studies that seek to answer a series of questions (**Q1**-**Q3**) using the GMM, DW-4, and LJ-13 energies. We also include additional ablation experiments in §G.3.

Q1: Bias and MSE of DEM vs. K. In Fig. 5 (left) we report the bias and mean squared error (MSE) of $\mathcal{S}_K$ versus K on the GMM in log-log plot. We find that as $K \to \infty$ the bias and MSE decrease as we increase K and in particular the bias goes to 0 which verifies that DEM is a consistent estimator. Additionally, a linear regression to the bias reveals an asymptotic decay rate of $O(1/K)$, which empirically validates Prop. 1 with a slightly sharper rate than $O(1/\sqrt{K})$.

Q2: MSE of DEM vs. t **for different** K. In Fig. 5 (right) we study the log-MSE as a function of diffusion time—*i.e.*, x_t for $t \in [0, 1]$—versus K on GMM. As observed, the log MSE drops as we increase the number of MC samples but increases as $t \to 1$ as we get closer to the prior. As we increase time the diffusion process moves us farther from the modes of $\mathcal{E}$, which means that our estimator has a higher bias for the same K. This is also empirically observed and supports the finding in the ablation experiments in the main paper §4.2 and is a consequence of Prop. 1.

Q3: The utility of a buffer in iDEM. We study the performance of DEM with and without samples from s_θ in a buffer. In particular, we ablate iDEM to prior-DEM (pDEM), which is a pure off-policy method, requiring no simulation-based outer loop. We see a clear trend on Fig. 6 (left) for DW-4 energy histograms, where using samples from s_θ leads to better performance. On LJ-13 plotted on Fig. 6 (right) iDEM and pDEM perform roughly the same and there are minor improvements in using s_θ. Finally, on LJ-55 pDEM failed to learn in 2 out of 3 runs, which highlights the increased stability of iDEM over pDEM.

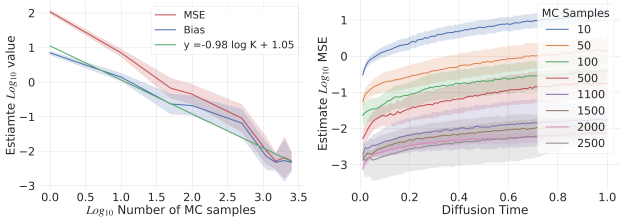

Figure 5. **Left:** Log-log plot of bias and MSE vs. K and a regression to the bias. **Right:** Plot of log bias vs. energy for different K. The MSE and bias are calculated for GMM with a linear noise schedule. The standard deviations for the log-transformed values are over 10 seeds with the variance estimated over 256 samples. For the plot on the right, the values are averaged over $x_0 \sim p_0$.

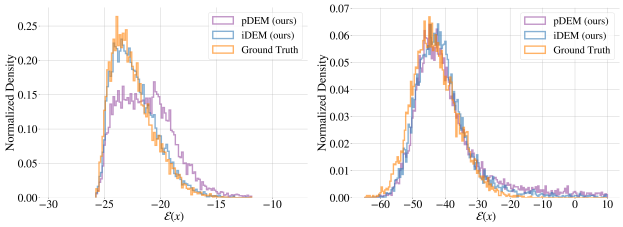

Figure 6. Comparison of the ground truth energy histograms of DW-4 (**left**) and LJ-13 (**right**) in relation to energies of samples generated from pDEM and iDEM.

5. Related work

MCMC and variational approximations. MC methods like AIS (Neal, 2001) and SMC (Del Moral et al., 2006) are often regarded as gold standards for sampling but are expensive and often hampered by slow convergence (Robert et al., 1999). Variational techniques such as mean-field approximation (Wainwright et al., 2008) and amortized methods like normalizing flows (Papamakarios et al., 2021)

are appealing alternatives for distribution approximation. Hybrid approaches that combine flows and MCMC to improve the transition kernels (Wu et al., 2020; Geffner & Domke, 2021; Thin et al., 2021; Doucet et al., 2022; Geffner & Domke, 2023; Grenioux et al., 2023) are an attractive compromise and have shown empirical benefits, *e.g.*, FAB (Midgley et al., 2023b), CRAFT (Matthews et al., 2022). Similar in approach to iDEM, Song et al. (2023) applies Monte Carlo approximation to the guidance term for solving inverse problems with diffusion models.

Equivariant flows and Boltzmann generators. Several key works use Boltzmann generators to sample from unnormalized probability densities (Noé et al., 2019). These include equivariant approaches using normalizing flows (Köhler et al., 2020; Midgley et al., 2023a; Klein et al., 2023b; Köhler et al., 2023). MD simulations have also seen the benefits of flow-based proposal distributions (Klein et al., 2023a). Generative models for $\mathrm{SE}(3)$-equivariant distributions span application domains such as robotics (Brehmer et al., 2023a;b), molecular modeling (Hoogeboom et al., 2022; Xu et al., 2022; Igashov et al., 2022), and protein generation (Yim et al., 2023b;a; Bose et al., 2024).

Neural samplers. Motivated by the Schrödinger bridge problem as a unifying perspective linking generative modeling to stochastic control (Pavon, 1989; Dai Pra, 1991; Tzen & Raginsky, 2019b), neural samplers seek to amortize MCMC. Most similar to our approach are the works of Berner et al. (2022); Vargas et al. (2023); Zhang & Chen (2022); Richter et al. (2024); Vargas et al. (2024) which exploit diffusion processes for fast mode mixing. However, these approaches require simulation to compute the objective, unlike iDEM. Finally, iDEM uses an iterative scheme where the sampler is trained on modifications of its own initial samples, resembling training diffusion models on their on data (Bertrand et al., 2023; Alemohammad et al., 2023).

GFlowNets. Continuous generative flow networks (Lahlou et al., 2023) are deep reinforcement learning algorithms that have the explicit aim of *off-policy* training of sequential samplers, diffusion-structured samplers being a particular case (Zhang et al. (2023); Lahlou et al. (2023), §4.2). These methods can stably learn from sampled states or trajectories without differentiation through the simulated process that produced them (Malkin et al., 2023). Avoiding SDE integration in the training loop is one of the motivations for our work, and iDEM can be seen as a simulation-free training algorithm for generative flow networks of a certain structure.

6. Conclusion

In this paper, we tackle the problem of amortized sampling from Boltzmann distributions. Our proposed iDEM algorithm uses a novel stochastic matching loss in the inner loop to train a diffusion sampler. Exploiting the amortization benefits of the diffusion sampler, we leverage it to propose informative samples to further accelerate its training. Empirically, we find iDEM to be significantly faster than previous approaches while offering high mode coverage and state-of-the-art performance on multiple benchmarks and is the first approach that is scalable to the challenging LJ-55 for energy-based training. While iDEM is computationally cheap, the DEM objective is biased and may be affected by the variance of the samples. Reducing the variance of DEM, including with adaptive techniques (Bugallo et al., 2017), and leveraging advances in SDE simulation to speed up the outer loop are natural directions for future investigation.

Impact statement

This work studies amortized sampling from Boltzmann densities, a problem of general interest in machine learning that arises both in pure statistical modeling (*e.g.*, sampling high-dimensional Bayesian posteriors over parameters) and in applications. We highlight the molecular design task—in turn applicable to drug and material discovery—as a motivation and immediate application for iDEM. While we do not foresee immediate negative impacts of our advances in this area, we encourage due caution to prevent their potential misuse.

Contribution statement

J.R., N.M., and A.T. initially conceived the idea of a stochastic off-policy continuous regression objective. A.J.B., T.A., G.G., and A.T. independently approached the problem through the lens of active inference with a generative model in a bi-level iterative scheme. Experiments were primarily led by T.A., J.R., A.T. (iDEM) and J.R,S.M., P.L., C.H.L., M.S. (ablations and baselines). G.G., A.J.B., and N.M. led the development of the theory. S.R., G.G., and Y.B. guided the project. A.J.B. and N.M. drove the writing of the paper, with contributions from all other authors. All authors contributed to designing the experiments.

Acknowledgments

The authors would like to thank James Vuckovic, Raymond Chua, Karam Ghanem, and Christos Tsirigotis for useful comments on early versions of this manuscript. In addition, the authors thank Julius Berner for sharing their code for PIS and DDS. A.J.B. is supported through an NSERC Postdoctoral fellowship.

The authors acknowledge funding from UNIQUE, CIFAR, NSERC, Intel, and Samsung. The research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca), Mila (https://mila.quebec), Dreamfold, Anyscale, Google GCP, and NVIDIA.

References

Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations (ICLR)*, 2023.

Albergo, M. S., Kanwar, G., and Shanahan, P. E. Flow-based generative models for markov chain monte carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019.

Alemohammad, S., Casco-Rodriguez, J., Luzi, L., Humayun, A. I., Babaei, H., LeJeune, D., Siahkoohi, A., and Baraniuk, R. G. Self-consuming generative models go mad. *International Conference on Learning Representations (ICLR)*, 2023.

Berner, J., Richter, L., and Ullrich, K. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Bertrand, Q., Bose, A. J., Duplessis, A., Jiralerspong, M., and Gidel, G. On the stability of iterative retraining of generative models on their own data. *International Conference on Learning Representations (ICLR)*, 2023.

Bose, A. J., Brubaker, M., and Kobyzev, I. Equivariant finite normalizing flows. *arXiv preprint arXiv:2110.08649*, 2021.

Bose, A. J., Akhound-Sadegh, T., Fatras, K., Huguet, G., Rector-Brooks, J., Liu, C.-H., Nica, A. C., Korablyov, M., Bronstein, M., and Tong, A. SE(3)-stochastic flow matching for protein backbone generation. *International Conference on Learning Representations (ICLR)*, 2024.

Brehmer, J., Bose, J., De Haan, P., and Cohen, T. EDGI: Equivariant diffusion for planning with embodied agents. *Neural Information Processing Systems (NeurIPS)*, 2023a.

Brehmer, J., De Haan, P., Behrends, S., and Cohen, T. Geometric algebra transformer. *Neural Information Processing Systems (NeurIPS)*, 2023b.

Bugallo, M. F., Elvira, V., Martino, L., Luengo, D., Miguez, J., and Djuric, P. M. Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79, 2017.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Neural Information Processing Systems (NIPS)*, 2018.

Dai Pra, P. A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, 23:313–329, 1991.

De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. Diffusion schrödinger bridge with applications to score-based generative modeling. *Neural Information Processing Systems (NeurIPS)*, 2021.

Del Moral, P., Doucet, A., and Jasra, A. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

Doucet, A., Grathwohl, W., Matthews, A. G., and Strathmann, H. Score-based diffusion meets annealed importance sampling. *Neural Information Processing Systems (NeurIPS)*, 2022.

Feroz, F., Hobson, M. P., Cameron, E., and Pettitt, A. N. Importance nested sampling and the MultiNest algorithm. *arXiv preprint arXiv:1306.2144*, 2013.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Garcia Satorras, V., Hoogeboom, E., Fuchs, F., Posner, I., and Welling, M. E(n) equivariant normalizing flows. *Neural Information Processing Systems (NeurIPS)*, 2021.

Geffner, T. and Domke, J. MCMC variational inference via uncorrected Hamiltonian annealing. *Neural Information Processing Systems (NeurIPS)*, 2021.

Geffner, T. and Domke, J. Langevin diffusion variational inference. *Artificial Intelligence and Statistics (AISTATS)*, 2023.

Grenander, U. and Miller, M. I. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

Grenioux, L., Durmus, A., Moulines, É., and Gabrié, M. On sampling with approximate transport maps. *arXiv preprint arXiv:2302.04763*, 2023.

Handley, W., Hobson, M., and Lasenby, A. Polychord: nested sampling for cosmology. *Monthly Notices of the Royal Astronomical Society: Letters*, 450(1):L61–L65, 2015.

Hastings, W. K. Monte carlo sampling methods using markov chains and their applications. 1970.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 2020.

Hoffman, M. D., Gelman, A., et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. *International Conference on Machine Learning (ICML)*, 2022.

Huang, X., Dong, H., Hao, Y., Ma, Y.-A., and Zhang, T. Reverse diffusion Monte Carlo. *International Conference on Learning Representations (ICLR)*, 2024.

Igashov, I., Stärk, H., Vignac, C., Satorras, V. G., Frossard, P., Welling, M., Bronstein, M., and Correia, B. Equivariant 3d-conditional diffusion models for molecular linker design. *International Conference on Learning Representations (ICLR)*, 2022.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983.

Klein, L., Foong, A. Y., Fjelde, T. E., Mlodozeniec, B., Brockschmidt, M., Nowozin, S., Noé, F., and Tomioka, R. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. *Neural Information Processing Systems (NeurIPS)*, 2023a.

Klein, L., Krämer, A., and Noé, F. Equivariant flow matching. *Neural Information Processing Systems (NeurIPS)*, 2023b.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: exact likelihood generative learning for symmetric densities. *International Conference on Machine Learning (ICML)*, 2020.

Köhler, J., Invernizzi, M., De Haan, P., and Noé, F. Rigid body flows for sampling molecular crystal structures. *International Conference on Machine Learning (ICML)*, 2023.

Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-García, A., Ezzine, L. N., Bengio, Y., and Malkin, N. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

Leimkuhler, B. and Matthews, C. Rational construction of stochastic numerical methods for molecular sampling. *Applied Mathematics Research eXpress*, 2013(1):34–56, 2013.

Lemos, P., Malkin, N., Handley, W., Bengio, Y., Hezaveh, Y., and Perreault-Levasseur, L. Improving gradient-guided nested sampling for posterior inference. *arXiv preprint arXiv:2312.03911*, 2023.

Li, S.-H. and Wang, L. Neural network renormalization group. *Physical review letters*, 121(26):260601, 2018.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *International Conference on Learning Representations (ICLR)*, 2023.

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

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

Matthews, A., Arbel, M., Rezende, D. J., and Doucet, A. Continual repeated annealed flow transport monte carlo. *International Conference on Machine Learning (ICML)*, 2022.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Midgley, L. I., Stimper, V., Antorán, J., Mathieu, E., Schölkopf, B., and Hernández-Lobato, J. M. SE(3) equivariant augmented coupling flows. *Neural Information Processing Systems (NeurIPS)*, 2023a.

Midgley, L. I., Stimper, V., Simm, G. N., Schölkopf, B., and Hernández-Lobato, J. M. Flow annealed importance sampling bootstrap. *International Conference on Learning Representations (ICLR)*, 2023b.

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

Neal, R. M. Slice sampling. *The annals of statistics*, 31(3): 705–767, 2003.

Neal, R. M. et al. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

Nicoli, K. A., Nakajima, S., Strodthoff, N., Samek, W., Müller, K.-R., and Kessel, P. Asymptotically unbiased estimation of physical observables with neural samplers. *Physical Review E*, 101(2):023304, 2020.

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

Owen, A. B. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Pavon, M. Stochastic control and nonequilibrium thermodynamical systems. *Applied Mathematics and Optimization*, 19:187–202, 1989.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. *International Conference on Machine Learning (ICML)*, 2015.

Richter, L., Berner, J., and Liu, G.-H. Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*, 2024.

Robert, C. P., Casella, G., and Casella, G. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

Roberts, G. O. and Rosenthal, J. S. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

Roberts, G. O. and Tweedie, R. L. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. *International Conference on Machine Learning (ICML)*, 2021.

Skilling, J. Nested sampling for general Bayesian computation. 2006.

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

Song, J., Zhang, Q., Yin, H., Mardani, M., Liu, M.-Y., Kautz, J., Chen, Y., and Vahdat, A. Loss-guided diffusion models for plug-and-play controllable generation. *International Conference on Machine Learning (ICML)*, 2023.

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

Speagle, J. S. DYNESTY: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *Monthly Notices of the Royal Astronomical Society*, 493 (3):3132–3158, 2020.

Thin, A., Kotelevskii, N., Doucet, A., Durmus, A., Moulines, E., and Panov, M. Monte Carlo variational auto-encoders. *International Conference on Machine Learning (ICML)*, 2021.

Tieleman, T. Training restricted boltzmann machines using approximations to the likelihood gradient. *International Conference on Machine Learning (ICML)*, 2008.

Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Tzen, B. and Raginsky, M. Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019a.

Tzen, B. and Raginsky, M. Theoretical guarantees for sampling and inference in generative models with latent diffusions. *Conference on Learning Theory (CoLT)*, 2019b.

Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*, 2023.

Vargas, F., Padhy, S., Blessing, D., and Nüsken, N. Transport meets variational inference: Controlled Monte Carlo diffusions. *International Conference on Learning Representations (ICLR)*, 2024.

Vershynin, R. *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018.

Wainwright, M. J., Jordan, M. I., et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

Wu, H., Köhler, J., and Noé, F. Stochastic normalizing flows. *Neural Information Processing Systems (NeurIPS)*, 2020.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

Yim, J., Campbell, A., Foong, A. Y. K., Gastegger, M., Jiménez-Luna, J., Lewis, S., Satorras, V. G., Veeling, B. S., Barzilay, R., Jaakkola, T., and Noé, F. Fast protein

backbone generation with SE(3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023a.

Yim, J., Trippe, B. L., De Bortoli, V., Mathieu, E., Doucet, A., Barzilay, R., and Jaakkola, T. SE(3) diffusion model with application to protein backbone generation. *International Conference on Machine Learning (ICML)*, 2023b.

Zhang, D., Chen, R. T. Q., Malkin, N., and Bengio, Y. Unifying generative models with GFlowNets and beyond. *arXiv preprint arXiv:2209.02606v2*, 2023.

Zhang, Q. and Chen, Y. Path integral sampler: a stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*, 2022.

A. Proofs of propositions

Proposition 1. *If* $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ *and* $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$ *are sub-Gaussian, then, there exists a constant* $c(x_t)$ *such that with probability* $1-\delta$ *(over* $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2)$*) we have* $\|\mathcal{S}_K(x_t,t) - \nabla \log p_t(x_t)\| \leq \frac{c(x_t)\log\left(\frac{1}{\delta}\right)}{\sqrt{K}}$.

Proof of Prop. 1. We seek to estimate $\nabla_{x_t} \log p_t(x_t) = -\nabla \mathcal{E}_t(x_t)$ where $\exp(-\mathcal{E}_t(x)) := \mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\exp(-\mathcal{E}(x_{0|t}))] \propto p_t(x)$. With a slight abuse of notation let us denote $x_{0|t} = (x_{0|t}^{(1)}, \cdots, x_{0|t}^{(K)})$, we consider the *biased estimator* $\mathcal{S}_K(x_{0|t},t) = \nabla \log \frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))$ with $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2)$. Denote, $\mathcal{S}(x_t,t) := -\nabla \mathcal{E}_t(x_t)$, we now estimate the bias of this estimator as a function of K.

If we assume that the random variables $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ and $\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))$ with $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2)$ are sub-Gaussian, then by Hoeffding's inequality on sub-Gaussian random variables (Vershynin, 2018), we have that there exists a constant $C > 0$ such that for any $\delta > 0$ with probability $1-\delta$ we have

$$\left| \frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)})) - \exp(-\mathcal{E}_t(x_t)) \right| \leq C\sqrt{\frac{\log(\frac{2}{\delta})}{K}} \tag{11}$$

and

$$\left\| \frac{1}{K}\sum_i \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)})) - \nabla \exp(-\mathcal{E}_t(x_t)) \right\| \leq C\sqrt{\frac{\log(\frac{2}{\delta})}{K}} \tag{12}$$

Thus, for $K \geq 4C^2 \log(1/\delta)\exp(2\mathcal{E}_t(x_t))$, with probability $1-\delta$ (over the sampling of $x_{0|t}$) we get,

$$\|\mathcal{S}_k(x_{0|t},t) - \mathcal{S}(x,t)\| = \left\| \frac{\frac{1}{K}\sum_i \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))} - \frac{\nabla \exp(-\mathcal{E}_t(x_t))}{\exp(-\mathcal{E}_t(x_t))} \right\| \tag{13}$$

$$= \left\| \frac{\exp(-\mathcal{E}_t(x_t))\frac{1}{K}\sum_i \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)})) - \frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)})\nabla \exp(-\mathcal{E}_t(x_t))}{\exp(-\mathcal{E}_t(x_t))\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \right\| \tag{14}$$

$$= \left\| \frac{e^{-\mathcal{E}_t(x_t)}(\frac{1}{K}\sum_i \nabla e^{-\mathcal{E}(x_{0|t}^{(i)})} - \nabla e^{-\mathcal{E}_t(x_t)}) + (e^{-\mathcal{E}_t(x_t)} - \frac{1}{K}\sum_i e^{-\mathcal{E}(x_{0|t}^{(i)})})\nabla e^{-\mathcal{E}_t(x_t)}}{\exp(-\mathcal{E}_t(x_t))\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \right\| \tag{15}$$

$$\leq C\sqrt{\frac{\log(\frac{2}{\delta})}{K}} \frac{\exp(-\mathcal{E}_t(x_t)) + \|\nabla \exp(-\mathcal{E}_t(x_t))\|}{\exp(-\mathcal{E}_t(x_t))\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \tag{16}$$

$$\leq C\sqrt{\frac{\log(\frac{2}{\delta})}{K}} \frac{1 + \|\nabla \mathcal{E}_t(x)\|}{\exp(-\mathcal{E}_t(x_t)) - C\sqrt{\frac{\log(\frac{2}{\delta})}{K}}} \tag{17}$$

$$\leq \frac{2C\sqrt{\log(\frac{2}{\delta})}(1 + \|\nabla \mathcal{E}_t(x_t)\|)(\exp(\mathcal{E}_t(x_t)))}{\sqrt{K}} \tag{18}$$

where for the last inequality we used the fact that K is large enough to have $\exp(-\mathcal{E}_t(x_t)) - C\sqrt{\frac{\log(\frac{2}{\delta})}{K}} \geq \exp(-\mathcal{E}_t(x_t))/2$ Thus, there exists a constant $c(x_t) > 0$ such that we have with probability $1-\delta$

$$\|\mathcal{S}_k(x_{0|t},t) - \mathcal{S}(x_t,t)\| \leq \frac{c(x_t)\sqrt{\log(1/\delta)}}{\sqrt{K}}. \tag{19}$$

Thus we have shown the conclusion of the proposition holds for K sufficiently large, which easily implies the general case (with a possibly larger value of $c(x_t)$). □

Proposition 2. *Let G be the product group $\mathrm{SE}(3) \times \mathbb{S}_n \hookrightarrow O(3n)$ and p_0 be a G-invariant density in $\mathbb{R}^d$. Then the Monte Carlo score estimator of $\mathcal{S}_K(x_t, t)$, is G-equivariant if the sampling distribution $x_{0|t} \sim \overline{\mathcal{N}}(x_{0|t}; x_t, \sigma_t^2)$ is G-invariant, i.e., $\overline{\mathcal{N}}(x_{0|t}; g \circ x_t, \sigma_t^2) = \overline{\mathcal{N}}(g^{-1} x_{0|t}; x_t, \sigma_t^2)$.*

Proof of Prop. 2. First note that since p_0 is G-invariant so, $\nabla p_0(x_{0|t}^{(i)})$ is G-equivariant. This means $\mathcal{E}$ and $\nabla \mathcal{E}$ are G-invariant and G-equivariant respectively. A group element g acts on $x \in \mathbb{R}^d$ in the standard way $g \circ x = gx$. G acts on the space of distributions over $\mathbb{R}^d$: if X is an $\mathbb{R}^d$-valued random variable with density p, then $g \circ X$ is distributed with density $(g \circ p)(x) = p(g^{-1}x)$. Applying the group action to $\bar{\mathcal{N}}$, we get

$$\begin{aligned}
\mathcal{S}_K(g \circ x_t, t) &= \sum_i \frac{\exp(-\mathcal{E}(g \circ x_{0|t}^{(i)}))}{\sum_j \exp(-\mathcal{E}(g \circ x_{0|t}^{(j)}))} \nabla \mathcal{E}(g \circ x_{0|t}^{(i)}) \\
&= g \circ \left(\sum_i \frac{\exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\sum_j \exp(-\mathcal{E}(x_{0|t}^{(j)}))} \nabla \mathcal{E}(x_{0|t}^{(i)}) \right) \\
&= g \circ \mathcal{S}_K(x_t, t), \\
x_{0|t}^{(1)}, \ldots, x_{0|t}^{(K)} &\sim \overline{\mathcal{N}}(x_t, \sigma_t^2).
\end{aligned}$$

Note that in the first line we used that $x_{0|t}^{(i)} \sim \overline{\mathcal{N}}(x_t, \sigma_t^2)$ is equivalent to $g \circ x_{0|t}^{(i)} \sim \overline{\mathcal{N}}(g \circ x_t, \sigma_t^2)$. □

B. iDEM for non-VE noising processes

We sketch how the objective described in §3.1 can be generalized to general noising processes of the form (1).

Consider a SDE of the form $dx_t = -\alpha(t) x_t \, dt + g(t) \, dw_t$, and define

$$y_t := \beta(t) x_t, \quad \beta(t) := \exp\left(-\int_0^t \alpha(s) \, ds \right). \tag{20}$$

We have $y_0 = x_0$, and, by Itô's lemma, y_t also obeys a SDE:

$$\begin{aligned}
dy_t &= [\beta'(t)x + \beta(t)\alpha(t)x]dt + g(t)\beta(t) \, dw_t \\
&= g(t)\beta(t) \, dw_t,
\end{aligned} \tag{21}$$

where we have used that $\beta'(t) = -\beta(t)\alpha(t)$ by the definition (20). The SDE (21) is variance-exploding, and the analysis in §2.2 (for diffusion models) or in §3.1 (for iDEM) applies to y_t.

This VE SDE generates marginal densities $\tilde{p}_t(y_t)$ from the initial distribution $\tilde{p}_0 = p_0$. An estimator of the score $\nabla \log \tilde{p}_t(y_t)$, which can be fit using the mentioned algorithms, is equivalent to an estimator of $\nabla \log p_t(x_t)$, since $\nabla \log \tilde{p}_t(y_t) = \nabla \log p_t(x_t)$. (Whether the neural network estimator takes x_t or the rescaled y_t as input is an implementation choice; we use x_t as input for numerical stability.)

Finally, we note that the above is readily generalized to the case of noising SDEs with matrix coefficients, $dx_t = A(t)x \, dt + G(t) \, dw_t$, a case which may be of interest in future generalizations of iDEM to Lie group-equivariant settings.

C. iDEM and flow matching

We remarked in §3.1 that the denoising diffusion objective (4) and the iDEM regression target (6) express the score of the convolution in different ways, with the diffusion objective using

$$\nabla p_t = \nabla(p_0 * \mathcal{N}(0, \sigma_t^2)) = p_0 * \nabla \mathcal{N}(0, \sigma_t^2)$$

and the iDEM objective using

$$\nabla p_t = \nabla(p_0 * \mathcal{N}(0, \sigma_t^2)) = \nabla p_0 * \mathcal{N}(0, \sigma_t^2).$$

It is interesting to consider the former expression in the case of a Boltzmann target distribution. We have, as in (4),

$$\nabla \log p_t(x_t) = \mathbb{E}_{x \sim p_0(x)\mathcal{N}(x; x_t, \sigma_t^2)}[\nabla \mathcal{N}(x; x_t, \sigma_t^2)] = \mathbb{E}_{x \sim p_0(x)\mathcal{N}(x; x_t, \sigma_t^2)}\left[\frac{x - x_t}{\sigma_t^2}\right]. \tag{22}$$

The quantity inside the expectation is the appropriately scaled velocity of a line segment from x_t to x. The expectation is intractable to compute, and Huang et al. (2024) recently proposed to estimate this expectation using Monte Carlo samples at generation time and showed bounds on the variance of the estimate. This can also be understood as a simulation-free estimation of the Föllmer drift; see Zhang & Chen (2022), §3.1.

This expression also recalls the method of *flow matching* (Lipman et al., 2023) for fitting the probability flow ODE of a stochastic process—a continuous normalizing flow (CNF)—given its boundary marginals.[3] Generalizations of flow matching by (Liu, 2022; Albergo & Vanden-Eijnden, 2023; Tong et al., 2023) allow learning ODEs linking arbitrary marginal distributions and amount to regressing the ODE's drift at x_t to the expected velocity of a line segment linking a source point to a target point and passing through x_t.

The expression (22) is the target of flow matching for a certain interpolant density, but the expectation over $p_0(x)\mathcal{N}(x; x_t, \sigma_t^2)$ is intractable. Tong et al. (2023) proposed an importance weighting solution that trains the CNF using flow matching on a weighted dataset of points x_0, allowing approximate flow matching with Boltzmann target densities. However, the results of this paper lead us to speculate about variants that estimate the regression target at x_t directly, as iDEM does, and take advantage of a buffer of past generated points x_0 for efficient simulation-free training.

D. Sampling with MCMC

This section focuses on the most popular methods for sampling from complex distributions that do not rely on deep learning. We provide a brief summary of some existing methods, but we refer the reader to the corresponding literature for more details.

D.1. Metropolis-Hastings

Perhaps the most commonly used sampling algorithm is the Metropolis-Hastings (MH) algorithm. The MH algorithm is a class of MCMC algorithm (Metropolis et al., 1953; Hastings, 1970), meaning based on the idea of constructing a Markov chain whose stationary distribution is the target distribution μ_{target}.

The algorithm starts by sampling an initial state x_0 from some proposed initial distribution $q(x_0)$, and then iteratively samples a new state x_{t+1} from a proposal distribution $q(x_{t+1} \mid x_t)$, typically a Gaussian distribution centred at x_t. The new state is accepted with probability $\alpha(x_t, x_{t+1})$, where

$$\alpha(x_t, x_{t+1}) = \min\left\{1, \frac{p_{\text{target}}(x_{t+1})q(x_t \mid x_{t+1})}{p_{\text{target}}(x_t)q(x_{t+1} \mid x_t)}\right\}. \tag{23}$$

The MH algorithm is guaranteed to converge to the target distribution if the detailed balance condition is satisfied:

$$p_{\text{target}}(x_t)q(x_{t+1} \mid x_t) = p_{\text{target}}(x_{t+1})q(x_t \mid x_{t+1}). \tag{24}$$

However, the algorithm can be very slow to converge, especially in high dimensions, and it is often necessary to use annealing schemes (§D.4) to improve mode coverage.

D.2. Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (Neal et al., 2011) is a different type of MCMC algorithm that uses the gradient of the energy function to construct a Markov chain that converges to the target distribution μ_{target}. Similarly to MH, the algorithm starts

[3] Note that for a VE SDE, the probability flow ODE is simply $dx_t = \frac{-g(t)^2}{2}\nabla \log p_t(x_t)\, dt$, so fitting the ODE amounts to learning the score.

by sampling an initial state x_0 from some proposed initial distribution $q(x_0)$. It then proceeds to iteratively sample new states x_{t+1} by simulating the Hamiltonian dynamics of a particle with mass m and position x_t in a potential energy field $-\mathcal{E}(x) = \log \mu_{\text{target}}$, for a fixed time T. Hamiltonian dynamics are given by the following system of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= \frac{p}{m} \\ \frac{dp}{dt} &= -\nabla \mathcal{E}(x), \end{aligned} \tag{25}$$

The HMC algorithm samples an auxiliary new momentum p_{t+1} from a Gaussian distribution centred at 0 and then uses an integrator, such as the leapfrog integrator, to simulate the Hamiltonian dynamics for a fixed time T. The new state x_{t+1} is then accepted with probability $\alpha(x_t, x_{t+1})$, given by (23). The HMC algorithm is more efficient than MH, thanks to the use of the gradient of the energy function, but it still struggles with mode coverage. Furthermore, the performance of the algorithm is very sensitive to tuning of the step size T and the mass m, although automatic tuning methods have been proposed, such as the No U-Turn Sampler (Hoffman et al., 2014).

D.3. Metropolis-Adjusted Langevin Algorithm

An alternative algorithm that uses the gradient of the energy function is the Metropolis Adjusted Langevin Algorithm (MALA) (Grenander & Miller, 1994; Roberts & Tweedie, 1996; Roberts & Rosenthal, 1998). Similarly to HMC, the algorithm starts by sampling an initial state x_0 from some proposed initial distribution $q(x_0)$. It then proceeds to iteratively sample new states x_{t+1} by simulating the Langevin dynamics of a particle with position x_t in a potential energy field $-\mathcal{E}(x) = \log \mu_{\text{target}}$, for a fixed time T. Langevin dynamics are given by the following system of differential equations:

$$\frac{dx}{dt} = -\nabla \mathcal{E}(x) + \sqrt{2}\xi, \tag{26}$$

where ξ is a standard Gaussian noise.

D.4. Annealed importance sampling

Markov chains have bad mode coverage, as the probability of jumping between separate modes can be extremely low. To help discover modes better we can combine chains with an annealing scheme where intermediate distributions p_j and p_{j+1} only differ slightly. For instance, in simulated annealing (Kirkpatrick et al., 1983), we can choose n distributions such that p_n allows for high mode coverage by setting $p_j \propto \mu_{\text{target}}^{\beta_j}$, for $1 = \beta_0 > \beta_1 > \cdots > \beta_L \geq 0$.

By viewing the annealing process as an importance sampling distribution we can derive a new estimator that has reduced variance in comparison to IS while achieving high mode coverage due to annealing. The Annealed Importance Sampling approach, like simulated annealing, considers a sequence of distributions, $\log p_j(x) = \beta_j \mu_{\text{target}}(x) + (1 - \beta_j) \log p_L(x)$, where the intermediate samples x^{j+1} produced by running a Markov chain transition (e.g HMC (Neal et al., 2011)) using samples $x^j \sim p_j$ leaves p_j invariant. Computing the importance weights along the sequence of distributions we get,

$$w_{\text{AIS}}(x^i) = \frac{p_{L-1}(x^i)}{p_L(x^i)} \frac{p_{L-2}(x^i)}{p_{L-1}(x^i)} \cdots \frac{\mu_{\text{target}}(x^i)}{p_1(x^i)}. \tag{27}$$

Plugging the weights w_{AIS} directly in the previously defined IS estimator gives us the AIS estimator (Neal, 2001).

D.5. Boltzmann generators

A Boltzmann generator (BG) samples from μ_{target} by combining an exact-likelihood model q_θ, typically a Normalizing Flow, and an algorithm to reweight model generated samples using μ_{target}. During training, a BG's flow is learned by minimizing a convex combination of both the forward and reverse KL-divergence. Thus, to compute the full training loss we need a dataset of ground truth samples from μ_{target} as it is required under the forward KL. As a result, BG's are ill-suited in settings with limited or no data as training only with the reverse KL is prone to mode-seeking behavior. Given a trained flow q_θ we can reweight observables $f(x)$ to be under μ_{target} by using the IS estimator.

D.6. Sequential Monte Carlo

Sequential Monte Carlo (SMC) (Del Moral et al., 2006) is an alternative to MCMC methods. It was originally proposed to find approximate solutions to filtering problems (hence, the original name particle filter methods), but was then adopted to sampling problems. In the context of sampling problems, SMC uses a sequence of distributions that map a known distribution to the target. The distributions are constructed in a way that the target distribution is the last distribution in the sequence. One typical choice for the sequence of distributions is the following:

$$p_j(x) = \mu_{\text{target}}(x)^{\beta_j} p_0(x)^{1-\beta_j}, \tag{28}$$

where p_0 is a known distribution, and β_j is a sequence of numbers such that $1 = \beta_0 > \beta_1 > \cdots > \beta_n \geq 0$. The SMC algorithm starts by sampling N particles from the initial distribution $p_0(x)$, and then proceeds to iteratively sample new particles from the intermediate distributions $p_j(x)$, for $j \in [1, \ldots, n]$, by applying a Markov transition kernel $K_j(x_{t+1} \mid x_t)$ to the particles $x_t \sim p_j(x)$. The particles are then weighted by the importance weights $w_j(x_t) = \frac{p_j(x_t)}{q_j(x_t)}$, and resampled according to the weights. The algorithm terminates when the particles are sampled from the target distribution. SMC can produce high-quality samples even for very complex distributions, particularly when using large numbers of intermediate distributions. Furthermore, the log-partition function can be computed as a by-product of the algorithm, as the average of the log-importance weights.

D.7. Nested Sampling

Nested Sampling (Skilling, 2006) is a method that was originally proposed for computing the evidence (or partition function) of a distribution but can also be used for sampling. Like in SMC, in Nested Sampling, we evolve from samples of a known distribution $q(x)$, which we call the prior distribution, to samples of the target distribution $\mu_{\text{target}}(x)$. However, in Nested Sampling we do not use a sequence of distributions, instead, we use a single distribution that is constructed by progressively removing the regions of low-probability mass. The fundamental concept behind nested sampling involves introducing a new variable known as the "cumulative prior mass" or "prior volume," defined as:

$$X(\lambda) = \int_{\mu_{\text{target}}(x) > \lambda} q(x) dx, \tag{29}$$

which signifies the proportion of the prior mass with a probability greater than that of the current point.

The core idea of nested sampling involves the following steps: Initially, a set of n_{live} "live points" is sampled from the prior distribution. The point with the lowest likelihood is identified and moved from the set, becoming a "dead point." Subsequently, it is replaced with a new point drawn from the prior, ensuring its likelihood surpasses that of the removed point. This replacing can be done through various procedures, such as building ellipsoids around the live point (Feroz et al., 2013), through slice sampling (Neal, 2003; Handley et al., 2015); or through Hamiltonian slice sampling (Speagle, 2020; Lemos et al., 2023). Although exact calculation of $X(\theta)$ for each new point is impractical, it can be approximated by recognizing that, at each iteration, the prior volume contracts by approximately:

$$\Delta X \approx \frac{n_{\text{live}}}{n_{\text{live}} + 1}. \tag{30}$$

The algorithm terminates when the prior volume is sufficiently small. The algorithm produces an estimate of the partition function, and samples from the distribution, by re-weighting the killed points by their importance weights:

$$w(x_k) = \frac{\mu_{\text{target}}(x_k) \cdot (X_{k-1} - X_k)}{Z}. \tag{31}$$

E. Related simulation-based and simulation-free diffusion-like samplers

In this section, we first describe the basic algorithms behind related work to understand their relative strengths and weaknesses.

Table 4. Table containing more detailed analysis of method properties during training, and during sampling. M MCMC steps per L annealing steps for FAB, and d dimensionality. These values assume a standard normalizing flow architecture for FAB as used in Midgley et al. (2023b;a).

Method	MCMC Free	Off-Policy	Gradient Time	Proposal Time	Memory	Sampling Time
FAB (Midgley et al., 2023b)	✗	✓	$\mathcal{O}(1)$	$\mathcal{O}(ML)$	$\mathcal{O}(L+d)$	$\mathcal{O}(d)$
PIS (Zhang & Chen, 2022)	✓	✗	$\mathcal{O}(L)$	$\mathcal{O}(L)$	$\mathcal{O}(Ld)$	$\mathcal{O}(Ld)$
DDS (Vargas et al., 2023)	✓	✗	$\mathcal{O}(L)$	$\mathcal{O}(L)$	$\mathcal{O}(Ld)$	$\mathcal{O}(Ld)$
DIS (Berner et al., 2022)	✓	✗	$\mathcal{O}(L)$	$\mathcal{O}(L)$	$\mathcal{O}(Ld)$	$\mathcal{O}(Ld)$
pDEM (ours)	✓	✓	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(Ld)$
iDEM (ours)	✓	✓	$\mathcal{O}(1)$	$\mathcal{O}(L)$	$\mathcal{O}(d)$	$\mathcal{O}(Ld)$

Flow Annealed Bootstrapping (Midgley et al., 2023b) (FAB). Flow Annealed Bootstrapping uses samples from annealed importance sampling (AIS) to train a normalizing flow model using an $\alpha = 2$ divergence. While a continuous normalizing flow (Chen et al., 2018) could be used for more flexibility in architecture, in practice a standard normalizing flow architecture is used, which constrains FAB to invertible architectures in contrast to continuous time models. This is because FAB requires computation of the model likelihood in its loss which requires simulation during training and makes using a continuous normalizing flow a computationally expensive choice.

We also note that FAB is most effective with a large buffer of AIS samples. Each AIS sampling step takes $\mathcal{O}(ML)$ steps where M is the number of MCMC steps per AIS intermediate distribution and L is the number of AIS levels. Finally, FAB needs to store the importance sampling ratio to compute w_{AIS} which increases its memory footprint. This leads to L importance sampling ratios to supplement the d dimensional data for $\mathcal{O}(L+d)$ memory footprint.

Finally, we note that FAB also finds that a biased objective often leads to better performance in practice than an unbiased objective. In this work, we come to a similar conclusion. That biased estimates can be quite effective in terms of sampling performance as compared to less efficient unbiased estimators.

Simulation-based SDE inference algorithms. The path integral sampler (PIS; Zhang & Chen, 2022), denoising diffusion sampler (DDS; Vargas et al., 2023), and (time-reversed) diffusion sampler (DIS; Berner et al., 2022) are related algorithms for training a neural SDE (Tzen & Raginsky, 2019a) to sample from a target distribution. These samplers aim to minimize a variational objective, the KL divergence between two distributions over trajectories: the distribution given by following the denoising SDE starting from the prior and the distribution given by following the fixed noising SDE starting from the target distribution. Such minimization can be achieved with varying choices of time discretizations or integration schemes, but all three methods approximate minimization of the divergence between path space measures in continuous time (Richter et al., 2024) and require numerical integration of the denoising SDE at each iteration of training, giving a computation time linear in the number of steps L. This also requires storing the trajectory for gradient computation leading to an $\mathcal{O}(Ld)$ memory footprint.

DEM. The computational complexity of DEM itself is low due to its simulation-free inner loop. With pDEM, sampling from the prior does not require any simulation during training (only during inference)

Sampling Complexity. All methods require simulating forward through the trajectory to generate samples. However, since FAB uses a standard normalizing flow, this is in general cheaper depending on architectural details (but less flexible).

F. Additional Details on the Experiments

F.1. Experimental Setup

In this section, we detail the exact setup for all of our experiments. For each experimental task and method (besides FAB for GMM, DW-4, and LJ-13 for which we use the hyperparameters reported best in Midgley et al. (2023b), Midgley et al. (2023a) respectively) we performed a grid search to find the best hyperparameters and evaluated each setting over three random seeds. For iDEM we use a geometric noise schedule $\sigma(t) = \sigma_{min}(\frac{\sigma_{max}}{\sigma_{min}})^t$ and tune over learning rate as well as σ_{min} and σ_{max}. For PIS we tune over the learning rate and the coefficient of the Brownian motion. For DDS we tune over using their proposed exponential or Euler integration, σ_{max} and α_{max} when using the exponential integration, and β_{min} and β_{max} for Euler integration. For FAB on LJ-55 we tune over learning rate and the number of intermediate distributions.

All networks were optimized using Adam and were performed on NVIDIA A100 GPUs with 40GB of VRAM. We commit to releasing our code upon publication.

We provide further details of our setup for each of the experiments below:

GMM. All models use an MLP with sinusoidal and positional embeddings. The MLP has 3 layers of size 128 as well as positional embeddings of size 128. Both iDEM and FAB use a replay buffer of max length 10000 which is prioritized by energy for FAB and unprioritized for iDEM. All methods were trained with Adam.

For training iDEM, the generated data was in the range [-1, 1] so to calculate the energy it was scaled appropriately by unnormalizing by a factor of 50. iDEM was trained with a geometric noise schedule with $\sigma_{\min} = 1e-5, \sigma_{\max} = 1$, $K = 500$ samples for computing the regression target $\mathcal{S}_K$ and we clipped the norm of $\mathcal{S}_K$ to 70. iDEM was trained with a learning rate of $5e-4$. All other methods did not normalize the data. PIS was trained with a learning rate of $5e-4$ and a Brownian motion coefficient of 30. DDS was trained with a learning rate of $5e-4$ and used the exponential integrator proposed in Vargas et al. (2023). We use $\alpha = 0.3$ and $\sigma = 30$ for the exponential integration. FAB was trained following exactly the settings used in Midgley et al. (2023b).

DW-4. All models besides FAB used an EGNN with 3 message-passing layers and a 2-hidden layer MLP of size 128. For FAB, we used the SE(3)-augmented coupling flow architecture from Midgley et al. (2023a) due to its requirement of an invertible architecture. iDEM was trained with a geometric noise schedule with $\sigma_{min} = 1e-5$ and $\sigma_{max} = 3$, a learning rate of $1e-3$, and $K = 1000$ samples for computing the regression target $\mathcal{S}_K$ and we clipped the regression target to a max norm of 20. PIS was trained with a learning rate of $5e-4$ and we used 1 for the coefficient of the Brownian motion. DDS was trained with a learning rate of $5e-3$ and we used Euler integration with $\beta_{min} = 0.5, \beta_{max} = 1.5$. FAB was trained following exactly the settings used in Midgley et al. (2023a).

LJ-13. All models besides FAB used an EGNN with 5 hidden layers and hidden layer size 128 while FAB used the architecture from Klein et al. (2023b). iDEM was trained with a geometric noise schedule with $\sigma_{min} = 0.01$ and $\sigma_{max} = 2$, a learning rate of $1e-3$, $K = 1000$ samples for the regression target $\mathcal{S}_K$ and clipped the regression target to a max norm of 20. PIS was trained with a learning rate of $1e-4$ and a Brownian motion coefficient of 1. DDS was trained with a learning rate of $5e-3$ and Euler integration with $\beta_{max} = 0.5$ and $\beta_{min} = 0.01$. FAB was trained following exactly the settings used in Midgley et al. (2023a) using their SE(3)-augmented coupling flow architecture with spherical projection.

LJ-55. All models besides FAB used an EGNN with 5 hidden layers and hidden layer size 128 while FAB used the architecture from Klein et al. (2023b). iDEM was trained with a geometric noise schedule with $\sigma_{min} = 0.5$ and $\sigma_{max} = 4$, a learning rate of $1e-3$, $K = 100$ samples for the regression target $\mathcal{S}_K$ and clipped the regression target to a max norm of 20. PIS was trained with a learning rate of $1e-4$ and a Brownian motion coefficient of 1. DDS was trained with a learning rate of $5e-3$ and Euler integration with $\beta_{max} = 0.5$ and $\beta_{min} = 0.01$. FAB was trained with the SE(3)-augmented coupling flow architecture from Midgley et al. (2023a) with spherical projection, 16 intermediate distributions, a learning rate of $2e-5$, and a batch size of 8 (the most that fit in GPU memory of the 40GB NVIDIA A100 GPUs used).

F.2. Metrics reported in Table 2 and Table 5

Tab. 2 depicts the main experimental sample quality results for various methods on all datasets and additional metrics are reported in Tab. 5. In this section, we explain the methodology for each experiment. We note that it was difficult to train PIS and DDS for the high-dimensional Lennard-Jones tasks. 2/3 seeds of DDS diverged for LJ-13.

Negative Log Likelihood (NLL). The negative log-likelihood measures how likely a test dataset is under a model. For different model types, there are different methods to numerically calculate or approximate the NLL of a test sample. In this work, we use the exact likelihood from a continuous normalizing flow (CNF) model. We train this CNF using optimal transport flow matching on samples from each model (OT-CFM) (Tong et al., 2023). Then this CNF can subsequently be used to calculate the likelihood of a test set. This allows us to compare the sample quality of various model architectures fairly using the same model architecture, training regime, and numerical likelihood approximation independent of the sampler form.

For the CNF with flow function $f : ([0, 1], \mathbb{R}^d) \to \mathbb{R}^d$, we use the exact estimator of the likelihood, *i.e.*, for a test sample x_0, its likelihood $p_{\text{model}}(x)$ can be estimated as

$$\log p_{\text{model}}(x) = \log p_{\text{prior}}(x_0) + \int_1^0 -\text{Tr}\left(\frac{df}{dx_t}\right) dt$$

where $x(t) = x_1 - \int_1^0 f(t, x)dt$ using the continuous-time change of variables formula. We note that it is important to choose the integration method correctly, as the model is only invertible in continuous time. If too few steps are taken then we observe inaccurate NLL values. Therefore we use the 5th order Dormand-Prince (dopri5) adaptive step size solver with tolerances `atol=rtol=`10^{-3} for GMM and `atol=rtol=`10^{-5} for DW-4, LJ-13, and LJ-55. This keeps the number of function evaluations per integration around 100 in practice, but results in a much more accurate and repeatable NLL value.

For more complicated datasets such as LJ13 and LJ55, we find that training a CNF on the output of some samples degrades the NLL performance of the CNF. We refer to a sampler as *diverged* if the negative log-likelihood of the CNF trained on its output is worse (higher) than an untrained sampler. For LJ13 and LJ55 respectively, an untrained CNF achieves a negative log likelihood of 60.32 and 230.53. In our reported results with mean and standard deviation, we exclude these values from the aggregation. We exclude values for LJ55 for the PIS, DDS, and pDEM models where training the CFM on samples from the sampling models do not improve the NLL of the CFM over random initialization.

There are other methods to compute or approximate NLL. In Tab. 6 we show that CFM is a relatively good estimator of the NLL as compared to the native estimations used in FAB and PIS respectively. This supports our use of a standardized model and training procedure for NLL computation.

In practice, we train CFM on 100k samples on GMM, DW-4, LJ-13 tasks and 10k samples on the LJ-55 task for all models. We train on fewer samples from LJ-55 due to the cost of sampling in this high-dimensional setting. It may be possible to achieve better NLL values with more samples (particularly in LJ-55) or with better architectures, due to our standardized pipeline, this does not affect the comparison between samplers evaluated in this work.

2-Wasserstein distance $\mathcal{W}_2$. We also use the standard 2-Wasserstein distance between empirical samples from the sampler and the ground truth dataset. The 2-Wasserstein distance is defined as

$$\mathcal{W}_2(\mu, \nu) = \left(\inf_{\pi} \int \pi(x, y) d(x, y)^2 dx dy \right)^{\frac{1}{2}} \tag{32}$$

where π is the transport plan with marginals constrained to μ and ν respectively. In practice, we use the Hungarian algorithm as implemented in the Python optimal transport package (POT) (Flamary et al., 2021) to solve this optimization for discrete samples. For simplicity, we use the Euclidean ground distance $d(x, y) = \|x - y\|_2$.

Effective Sample Size (ESS). To measure ESS we first evaluate log importance weights given by using our model $p_{model}(x)$ as a proposal. We estimate the ESS by

$$ESS = \frac{n}{\sum_{i=1}^{n} w_i^2}$$

where $w_i = \frac{\exp(-\mathcal{E}(x_i))/p_{model}(x_i)}{\sum_{j=1}^{n} \exp(-\mathcal{E}(x_j))/p_{model}(x_j)}$, which is equivalent to the Softmax of the log-probability ratios. We note that this is sometimes known as the normalized ESS, as we normalize the effective sample size to the fraction of the test set size.

Total Variation. The total variation metric on the Gaussian mixture model dataset is taken over 200 bins in each dimension (200^2) total bins. This is possible in low dimensions but does not scale well to high dimensions, requiring an exponential number of bins. Therefore for the larger equivariant datasets, we take the total variation distance over the distribution of the interatomic distances of the particles, see *e.g.*, Fig. 4, which scales well with dimension and shows how well the distribution of energies matches that of the test data.

Log partition function ($\log Z$ **)**. To compute $\log Z$ we use an importance sampling estimate with the proposal density, $q(x)$, given by the OT-CFM model as

$$\log Z = \log \mathbb{E}_{x \sim q(x)} \left[\frac{\exp(-\mathcal{E}(x))}{q(x)} \right] \tag{33}$$

$$\geq \mathbb{E}_{x \sim q(x)} [-\mathcal{E}(x) - \log q(x)] \tag{34}$$

which yields a lower bound on the true log partition function. Note that for GMM we know the true value of $\log Z$, however, this isn't the case for the equivariant tasks. For those tasks, to compare the values of $\log Z$ from the different methods, we use the fact that our estimate is a lower bound of the true value and favour the method with the largest estimate.

F.3. Timing experiment setup

To compute the training times in Table 3 we first measure the time per iteration while excluding all computations used for evaluation. Next, we determined the number of training steps required for each method to converge based on training loss. Finally, we reported the number of iterations taken to converge multiplied by the number of iterations per second for each method in Table 3.

F.4. Task details

F.4.1. Gaussian Mixture Model

We use a 40 Gaussian mixture density in 2 dimensions as proposed by Midgley et al. (2023b). This density consists of a mixture of 40 evenly weighted Gaussians with identical covariances

$$\Sigma = \begin{pmatrix} 40 & 0 \\ 0 & 40 \end{pmatrix} \tag{35}$$

and μ_i are uniformly distributed over the -40 to 40 box, *i.e.*, $\mu_i \sim \mathcal{U}(-40, 40)^2$.

$$p_{\text{gmm}}(x) = \frac{1}{40} \sum_{i=1}^{40} \mathcal{N}(x; \mu_i, \Sigma) \tag{36}$$

We use a test set of size 1000 sampled with TORCH.RANDOM.SEED(0) following prior work.

F.4.2. DW-4

The energy function for the DW-4 dataset was introduced in Köhler et al. (2020) and corresponds to a system of 4 particles in a 2-dimensional space. The system is governed by a double-well potential based on the pairwise distances of the particles. For a system of 4 particles, $x = \{x_1, \dots, x_4\}$, the energy is depicted in Appendix F.4.2 and is given by:

$$\mathcal{E}^{\text{DW}}(x) = \frac{1}{2\tau} \sum_{ij} a(d_{ij} - d_0) + b(d_{ij} - d_0)^2 + c(d_{ij} - d_0)^4 \tag{37}$$

where $d_{ij} = \|x_i - x_j\|_2$ is the Euclidean distance between particles i and j. Following previous work, we set $a = 0$, $b = -4$, $c = 0.9$ and the temperature parameter $\tau = 1$. To evaluate the efficacy of our samples we use a validation and test set from the the MCMC samples in Klein et al. (2023b) as the "Ground truth" samples. We note that this is not necessarily a perfect ground truth, but we believe it is reasonable to use for our purposes. Previous work has evaluated performance based on a dataset from Garcia Satorras et al. (2021). However, we note that this dataset is biased as its test set partition is generated from a single MCMC chain. As such, we omit this dataset and evaluate only on the data from Klein et al. (2023b).

F.4.3. Lennard-Jones Potential

The Lennard-Jones (LJ) potential is an intermolecular potential which models repulsive and attractive interactions of non-bonding atoms or molecules. The energy is based on the distance of interacting particles and is given by:

$$\mathcal{E}^{\text{LJ}}(x) = \frac{\epsilon}{2\tau} \sum_{ij} \left(\left(\frac{r_m}{d_{ij}} \right)^6 - \left(\frac{r_m}{d_{ij}} \right)^{12} \right) \tag{38}$$

where $d_{ij} = \|x_i - x_j\|_2$ is the Euclidean distance between particles i and j, and r_m, τ, ϵ and c are physical constants. As in Köhler et al. (2020), we also use a harmonic potential:

$$\mathcal{E}^{\text{osc}}(x) = \frac{1}{2} \sum_{i} ||x_i - x_{\text{COM}}||^2 \tag{39}$$

where x_{COM} refers to the center of mass of the system. Therefore, the final energy is then $\mathcal{E}^{Tot} = \mathcal{E}^{\text{LJ}}(x) + c\mathcal{E}^{\text{osc}}(x)$, for c the oscillator scale. As in previous work, we use $r_m = 1$, $\tau = 1$, $\epsilon = 1$ and $c = 0.5$. We note that this task is difficult

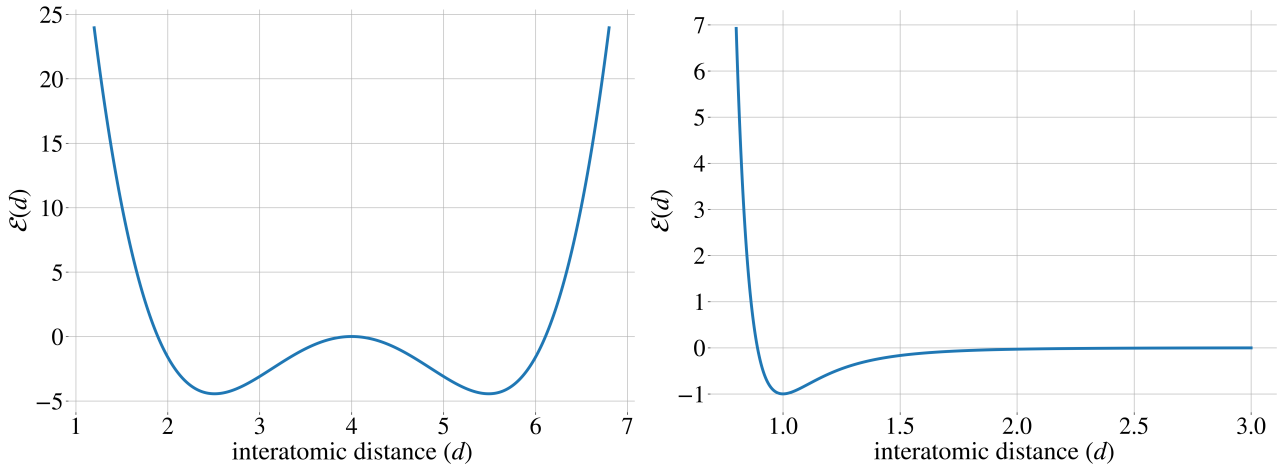

Figure 7. Energies of (left) double-well potential and (right) Lennard-Jones potential ($\mathcal{E}^{LJ}$) as a function of the interatomic distance between the particles.

not only because of its dimensionality but also because of the high magnitude of its score. As depicted in Fig. 7 (left) and apparent from Eq. 38, the score explodes as any $d_{ij} \to 0$, making this task particularly challenging in higher dimensions.

LJ-13 refers to the system of 13 particles, $x = \{x_1, \dots, x_{13}\}$ with 3 dimensions each, resulting in a task with dimensionality $d = 39$. LJ-55 meanwhile refers to a system of 55 particles, $x = \{x_1, \dots, x_{55}\}$ with 3 dimensions each, resulting in a high dimensional task with dimensionality $d = 165$. For the experimental results, we evaluate using the MCMC samples from Klein et al. (2023b). As with DW-4, previous work evaluated performance based on a dataset from Garcia Satorras et al. (2021). However, this dataset is also biased with the test set partition generated from a single MCMC chain and is generated with $\mathcal{E}^{LJ}/2$ as the sum is only calculated for $i < j$. Therefore, we only evaluate the models on the data from Klein et al. (2023b).

G. Additional results

G.1. Additional metrics

We report here additional quantitative metrics to supplement the main results presented in §4.1. In Tab. 5, we report the Total Variation (TV) as well as the log partition function ($\log Z$) values.

Table 5. Sampler performance with mean ± standard deviation over 3 seeds for Total Variation (TV), and log partition function ($\log Z$). $*$ indicates divergent training and entries with † refer to settings where only 1 of 3 runs converged. For TV, we **bold** via Welch's two-sample t-test $p < 0.1$. For $\log Z$, we **bold** the method with the largest value, as our estimate is a lower bound on the true $\log Z$.

Energy →	GMM ($d = 2$)		DW-4 ($d = 8$)		LJ-13 ($d = 39$)		LJ-55 ($d = 165$)	
Algorithm ↓	TV	$\log Z$	TV	$\log Z$	TV	$\log Z$	TV	$\log Z$
FAB (Midgley et al., 2023b)	0.88±0.02	-1.165 ±0.164	**0.09**±0.00	29.602 ±0.019	**0.04**±0.00	4.35±0.01	0.24±0.09	32.809†
PIS (Zhang & Chen, 2022)	0.92±0.01	-2.243 ±0.070	**0.09**±0.00	29.599 ±0.009	0.25±0.01	46.685 ±1.471	$*$	$*$
DDS (Vargas et al., 2023)	**0.82**±0.02	-0.358 ±0.209	0.16±0.01	28.382 ±0.158	$*$	$*$	$*$	$*$
pDEM (ours)	**0.82**±0.02	-0.370 ±0.005	0.13±0.00	29.191 ±0.036	0.06±0.02	32.450 ±3.191	$*$	$*$
iDEM (ours)	**0.82**±0.01	**-0.340** ±0.075	**0.10**±0.01	**29.567** ±0.014	**0.04**±0.01	**49.969** ±2.784	**0.09**±0.01	**273.167** ±22.226

In Tab. 6 we report the NLL values that are computed between the original method and the CFM model that is trained using the samples of each method.

In Fig. 8 we plot the energy histogram of the DW-4 system in comparison to all the methods. We observe that iDEM and FAB achieve similar performance and match the ground truth energy. PIS and DDM also are able to learn using this energy but are noticeably worse than iDEM and FAB.

Table 6. This table compares native vs. negative log-likelihood estimation by retraining a flow matching model on samples. FAB-Native directly admits a likelihood calculation through its invertible architecture. PIS-Native uses a stochastic estimation of an upper bound on the NLL. 231 is the negative log-likelihood of a randomly initialized CFM model, therefore any value larger than this is worse than random. We report NLL values worse than the initialization for CFM as ≥ 231.

Algorithm ↓ Energy →	GMM	DW-4	LJ-13	LJ-55
FAB-Native	$7.12_{\pm 0.12}$	$7.19_{\pm 0.06}$	$17.40_{\pm 0.10}$	$1355_{\pm 1379}$
FAB-CFM	$7.14_{\pm 0.01}$	$7.16_{\pm 0.01}$	$17.52_{\pm 0.17}$	$200.32_{\pm 62.30}$
PIS-Native	$7.92_{\pm 0.06}$	$7.31_{\pm 0.02}$	$47.783_{\pm 2.283}$	$449.794_{\pm 476.535}$
PIS-CFM	$7.72_{\pm 0.03}$	$7.19_{\pm 0.01}$	$47.05_{\pm 12.46}$	≥ 231

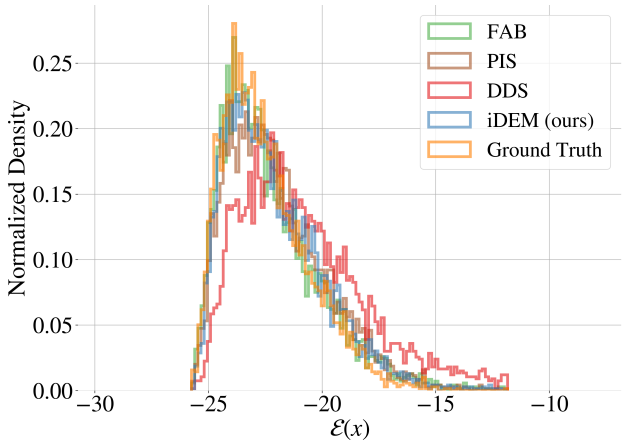

Figure 8. Comparison of the ground truth energy histograms of DW-4 in relation to energies of samples generated from iDEM and baseline methods.

G.2. Interatomic distances

We report the interatomic distances of the ground truth system and model-generated samples for the DW-4, LJ-13, and LJ-55 tasks in Fig. 9. We find the highest agreement with the ground truth method and iDEM with the differences between baselines increasing with the complexity of the dataset.

G.3. Further ablations

Different choices of $\mathcal{S}_K$. While in practice we use the biased estimate (8) one could consider using other estimates of $\nabla_{x_t} \log p_t(x_t)$ as a regression target. For example, instead of applying the LogSumExp trick as is done in (8) one could work directly with the ratio estimate (7), namely

$$\begin{aligned}\nabla_{x_t} \log p_t(x_t) &= \frac{\mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\nabla \exp(-\mathcal{E}(x_{0|t}))]}{\mathbb{E}_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\exp(-\mathcal{E}(x_{0|t}))]} \\ &\approx \frac{\frac{1}{K}\sum_i \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \\ x_{0|t}^{(1)}, \ldots, x_{0|t}^{(K)} &\sim \mathcal{N}(x_t, \sigma_t^2),\end{aligned}$$

Inherent in this estimate are drawbacks, chief among them that we must work with the non-log scale energies $\exp(-\mathcal{E}(x_{0|t}))$ directly which can often be either extremely large or extremely small depending on the landscape of $\mathcal{E}$. This can cause large variance estimates and numerical instability if one is unlucky.

Another possible estimate for $\nabla \log p_t(x_t)$ is inspired by Jensen's inequality. In particular, if we write $\nabla \log p_t(x_t) = \nabla \log E_{x_{0|t}\sim\mathcal{N}(x_t,\sigma_t^2)}[\exp(-\mathcal{E}(x_{0|t})]$ we can push the log inside the expectation and observe that

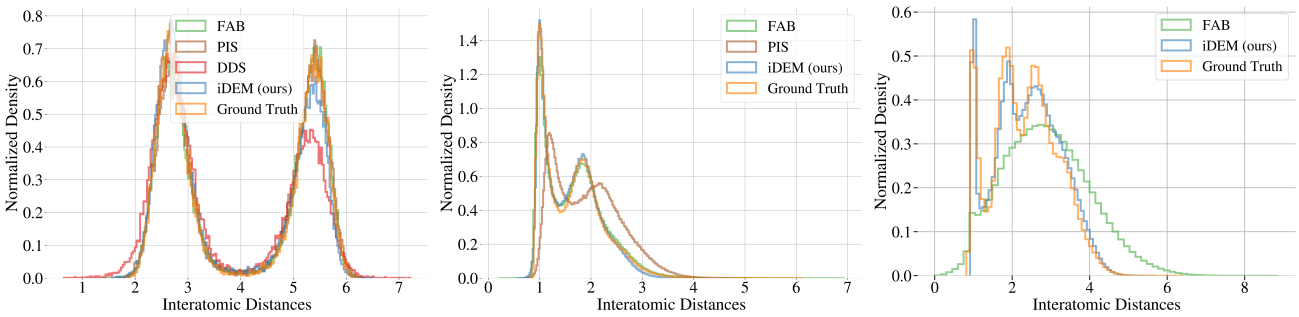

Figure 9. Interatomic distances for DW-4 (**Left**), LJ-13 (**mid**), and LJ-55 (**right**) of the ground truth versus the model-generated samples. Note that PIS is omitted from the LJ-55 plot and DDS is omitted from both LJ-13 and LJ-55 plots as the samples of these models diverged in these settings.

$$
\begin{aligned}
\nabla_{x_t} \log p_t(x_t) &\approx \nabla_{x_t} \mathbb{E}_{x_{0|t} \sim \mathcal{N}(x_t, \sigma_t^2)}[-\mathcal{E}(x_{0|t})] \\
&= \nabla_{x_t} \int -\mathcal{N}(x_{0|t}; x_t, \sigma_t^2) \mathcal{E}(x_{0|t}) dx_{0|t} \\
&= -\int \nabla_{x_t} \mathcal{N}(x_{0|t}; x_t, \sigma_t^2) \mathcal{E}(x_{0|t}) dx_{0|t} \\
&= -\int \mathcal{N}(x_{0|t}; x_t, \sigma_t^2) \mathcal{E}(x_{0|t}) \nabla_{x_t} \log \mathcal{N}(x_{0|t}; x_t, \sigma_t^2) dx_{0|t} \\
&= -\frac{1}{\sigma_t^2} \mathbb{E}_{x_{0|t} \sim \mathcal{N}(x_t, \sigma_t^2)} \left[\mathcal{E}(x_{0|t})(x_{0|t} - x_t) \right],
\end{aligned}
$$

This estimate is reminiscent of the estimate used in typical score matching where we have access to samples from p_0. Moreover, the estimate does not require access to the gradient of $\mathcal{E}$ making it an attractive option if it is indeed faithful to the true score $\nabla_{x_t} \log p_t(x_t)$.

We investigate the behaviour of each estimate in Figure 10, examining the MSE between the score estimated by each technique and the true score on the GMM task as a function of the number of MC samples. Unfortunately, we observe that despite the Jensen estimate's attractiveness the score estimates it yields have a large MSE which does not readily reduce as the number of MC samples increases indicating a significant bias. On the other hand, the unbiased estimate suffers from extremely large variance due to it having to work directly with the exponentiated energies $\exp(-\mathcal{E}(x_t))$. This results in every estimate before 500 MC samples resulting in exclusively NaN or infinite estimates, rendering the estimate unusable in practice. The only estimate which is well-behaved and approaches the true score is the LogSumExp estimate, whose use we advocate throughout this work.

Finally, we note that ours is not the only biased objective which achieves good performance in practice: *e.g.*, the current SOTA FAB's objective is also biased due to its use of finite samples for importance sampling as well as its sampling without replacement from the replay buffer.

Bias vs. Energy for different K and Total Variation Distance.

In Fig. 11 (left) we study the log-bias as a function of $\mathcal{E}$, by taking a point from a GMM mode and linearly following a direction away from the data itself which corresponds to $\exp(-\mathcal{E}(x)) \approx 0$. As observed, we notice we need a few MC samples in high-data regions. Additionally, the log-bias increases further from data for a fixed K and drops when we increase K. This result empirically confirms Prop. 1 where we need more MC samples to have a proper estimation of the score.

In Fig. 11 (right) we visualize the progression of the TV metric during training across MC samples on DW-4. In line with expectations, increasing the number of MC samples K leads to a lower biased estimate and a better performance of the model.

Estimator quality as a function of dimension. We examine the quality of the estimator as a function of dimensions, with a

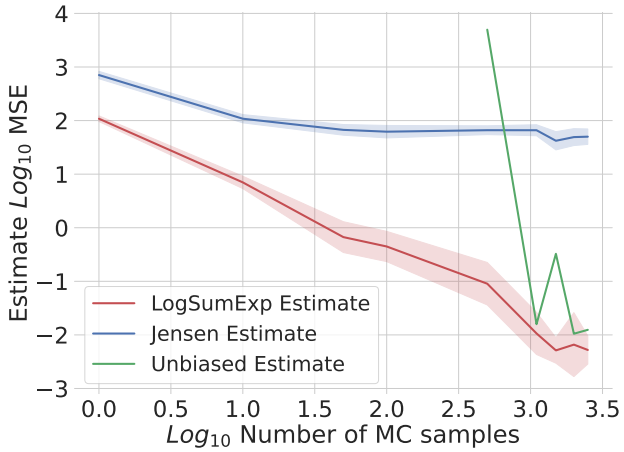

Figure 10. Comparison of different methods to compute $\mathcal{S}_K$, namely the LogSumExp estimate we use, as well as the Jensen estimate and ratio estimate. As the ratio estimator must work with non-log scale energies $\exp(-\mathcal{E}(x_t))$ it frequently results in NaN estimates which we have omitted from the plot, along with the standard deviations of the ratio estimates (which are also NaN).

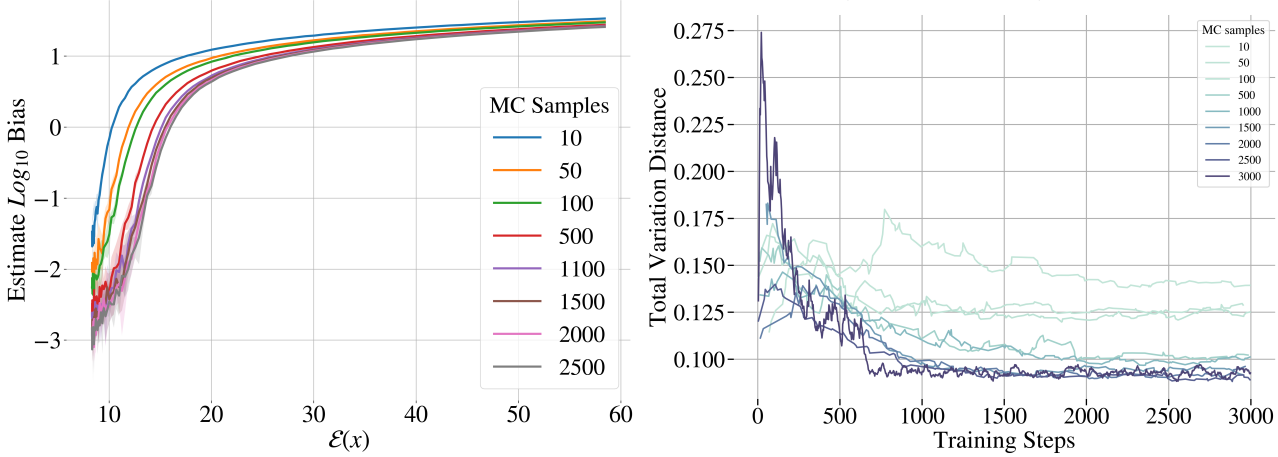

Figure 11. **Left:** Plot of log bias vs. energy for different K. The MSE and bias are calculated for GMM with a linear noise schedule. The standard deviations for the log-transformed values are over 10 seeds with the variance estimated over 256 samples. For the plot on the right, the values are averaged over $x_0 \sim p_0$ **Right:** Ablation of Total Variation distance as a function training steps for various numbers of MC samples on DW-4.

closed-form score for the 40-GMM task. Specifically, we analyze the bias and the MSE (capturing variance) of the estimator as a function of increasing number of dimensions, across various noise levels, $\sigma(t)$ in Fig. 12 (left, middle). As expected, we find that the bias and MSE increase as a function of dimensions. In line with our expectations and previous findings shown in Fig. 5 (right), we note that the bias and MSE are both higher at higher noise levels. In Fig. 12 (right), we measure the cosine similarity of the mean estimated score to the true score. As this value is very close to 1 for the dimensions we analyze, we conclude that, interestingly, the estimator error is almost exclusively in the magnitude and not in the direction. This interesting finding is also aligned with our experiments, where we observed that we were able to get good samples, even in high-dimensional tasks, simply by clipping the estimated score. Overall, we hypothesize that overestimating the magnitude of the score early in the diffusion process (i.e., at large values of $\sigma(t)$) does not pose a significant problem as long as the estimated scores still point towards the data density and that later in the diffusion process (i.e., at smaller values of $\sigma(t)$) we are able to get accurate estimates. The reason is that although using the overestimated scores we might move towards the data density too quickly or even overshoot it, by taking small enough steps we will eventually enter a region where our estimates are accurate and we can converge towards the data density.

Finally, to demonstrate that the errors in the magnitude of the estimated score are manageable even in a very high-dimensional setting, we scale up the 40-GMM task from 2 to 10000 dimensions. As we want to examine the quality of the estimator itself, we use the estimator directly, instead of a network, to generate samples. Although clipping is necessary to achieve

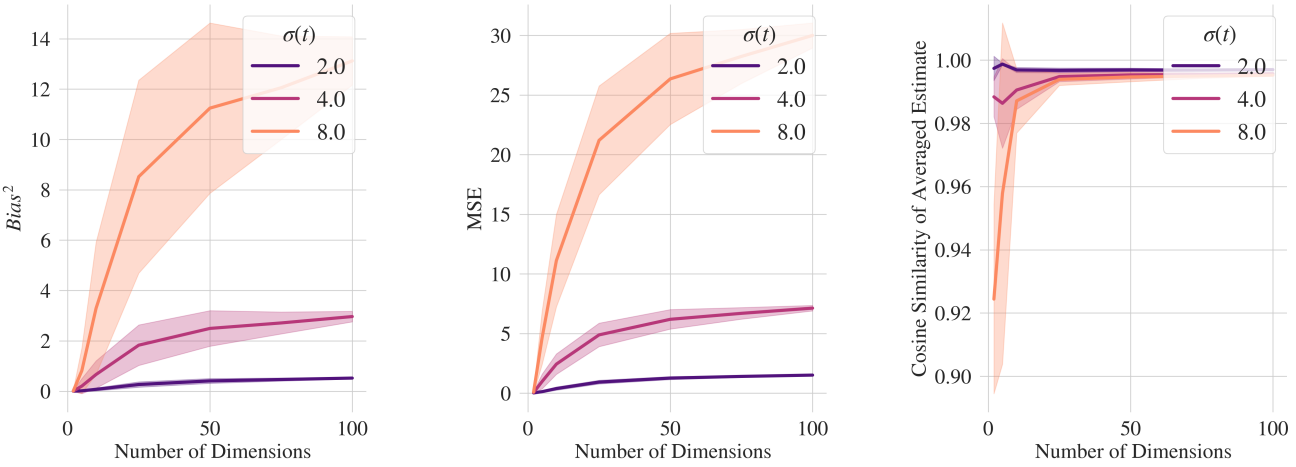

Figure 12. Plot of the **Left:** bias-squared **Middle:** mean squared error (MSE) and **Right:** cosine similarity of the mean estimated score to the ground truth score as a function of the number of dimensions of the 40-GMM. These are plotted for x_t sampled at three noise levels $\sigma(t) \in \{2, 4, 8\}$. The standard deviations for all the values are over 10 seeds with the variance estimated over 128 samples.

good samples, the estimator is able to achieve good performance over a wide range of clipping values, from 10^4 to 10^8. In Fig. 13, we demonstrate this by plotting the true and predicted distributions on the first two principal components using the estimator, with norm clipped to 10^4 and 10^8 respectively.

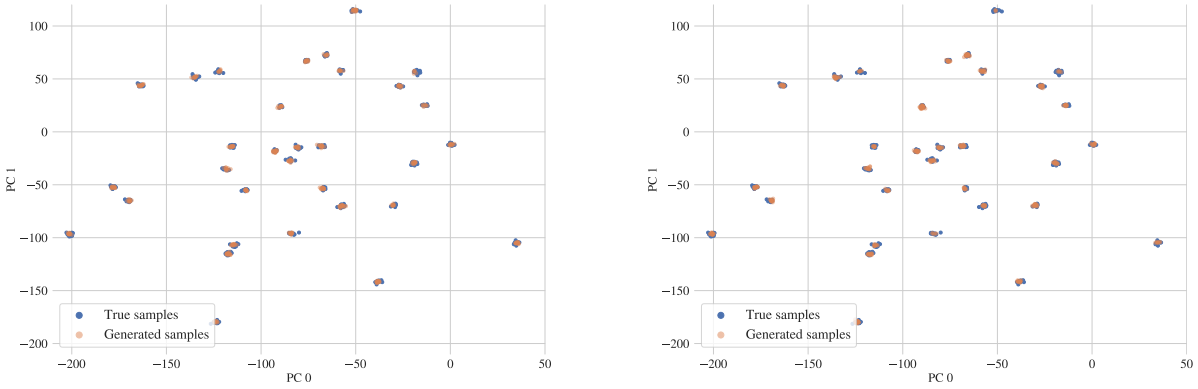

Figure 13. Generated and true samples for the 40-GMM task in 10,000 dimensions plotted over the first two principal components. The samples are generated using the score estimator with norm clipped to **Left:** 10^4 and **Right:** 10^8.