Training Neural Samplers with Reverse Diffusive KL Divergence

Jiajun He[*,1] **Wenlin Chen**[*,1,2] **Mingtian Zhang**[*,3]
David Barber[3] **José Miguel Hernández-Lobato**[1]

[*]Equal contribution

[1]University of Cambridge [2]MPI for Intelligent Systems [3]University College London

jh2383@cam.ac.uk wc337@cam.ac.uk m.zhang@cs.ucl.ac.uk

Abstract

Training generative models to sample from unnormalized density functions is an important and challenging task in machine learning. Traditional training methods often rely on the reverse Kullback-Leibler (KL) divergence due to its tractability. However, the mode-seeking behavior of reverse KL hinders effective approximation of multi-modal target distributions. To address this, we propose to minimize the reverse KL along diffusion trajectories of both model and target densities. We refer to this objective as the *reverse diffusive KL divergence*, which allows the model to capture multiple modes. Leveraging this objective, we train neural samplers that can efficiently generate samples from the target distribution in *one step*. We demonstrate that our method enhances sampling performance across various Boltzmann distributions, including both synthetic multi-modal densities and n-body particle systems.

1 INTRODUCTION

Sampling from unnormalized distributions is an essential and challenging research problem with wide applications in machine learning, Bayesian inference, and statistical mechanics. Consider a target distribution with an analytical but unnormalized density function:

$$p_d(x) = \exp(-E(x))/Z, \tag{1}$$

where x is the random variable to be sampled, $E : \mathbb{R}^n \to \mathbb{R}$ is a lower-bounded differentiable energy function, and $Z = \int \exp(-E(x))dx$ is the intractable normalization constant. A common approach to sampling from $p_d(x)$ involves designing MCMC samplers (Neal et al., 2011; Chen et al., 2024b). However, for high-dimensional, multi-modal target distributions, MCMC methods often take a long time to converge and need to simulate a very long chain for the samples to be uncorrelated (Pompe et al., 2020). This presents significant challenges in large-scale simulation problems.

Alternatively, one can approximate the target distribution $p_d(x)$ with a generative model $p_\theta(x)$, such as a normalizing flow or a latent variable model $p_\theta(x) \coloneqq \int p_\theta(x|z)p(z)dz$, which is easier to sample from. This model is often referred to as a *neural sampler* (Levy et al., 2017; Wu et al., 2020; Arbel et al., 2021; di Langosco et al., 2021). Training a neural sampler involves learning the model parameters θ, which is usually achieved by minimizing a divergence between $p_\theta(x)$ and $p_d(x)$. A common choice to train the neural sampler is the reverse KL divergence due to its tractability.

However, the most significant limitation of reverse KL is the *mode collapse* phenomenon due to its mode-seeking behavior (Bishop, 2006). This means that when the target distribution $p_d(x)$ contains multiple distant modes, the model $p_\theta(x)$ trained by reverse KL will underestimate the variance of $p_d(x)$ and can only capture a few modes. This is undesirable since the target distributions, such as the Bayesian posteriors (Welling and Teh, 2011) and Boltzmann distributions (Noé et al., 2019), often exhibit multiple modes.

In this paper, we propose to use an alternative objective, the *diffusive KL divergence* (DiKL). This objective convolves both the target and the model distributions with Gaussian diffusion kernels, allowing for better connectivity and merging of distant modes in the

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

noisy space. Notably, DiKL is still a valid divergence between the model density and the *original* target distribution, which allows us to learn the original target with better mass-covering capability. We further introduce a tractable gradient estimator for reverse DiKL, enabling practical training of neural samplers with this divergence. We demonstrate the effectiveness of our approach on both synthetic and n-body system targets, where it matches previous state-of-the-art models with reduced training and sampling costs.

2 BACKGROUND: KL DIVERGENCE

Given access to target samples $\{x_1, \ldots, x_N\} \sim p_d(x)$, it is common to minimize the forward KL divergence to fit a generative model $p_\theta(x)$ to the target $p_d(x)$:

$$\mathrm{KL}(p_d||p_\theta) \approx -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x_n) + \text{const.}, \quad (2)$$

which is equivalent to maximum likelihood estimation (MLE). However, our setting only assumes access to the unnormalized density of $p_d(x)$ without samples, where the reverse KL divergence is typically employed:

$$\begin{aligned}\mathrm{KL}(p_\theta||p_d) &= \int (\log p_\theta(x) - \log p_d(x)) p_\theta(x) dx \\ &= \int (\log p_\theta(x) + E(x)) p_\theta(x) dx + \log Z, \end{aligned} \quad (3)$$

where $\log Z$ is a constant independent of x. The integration over $p_\theta(x)$ can be approximated by the Monte Carlo method with samples from $p_\theta(x)$, and the gradient of the reverse KL w.r.t. the model parameter θ can be obtained by auto differentiation with the reparameterization trick (Kingma and Welling, 2013). This objective is particularly suitable for models with analytically tractable marginal densities, such as normalizing flows (Papamakarios et al., 2019; Rezende et al., 2020; Dinh et al., 2016; Kingma and Dhariwal, 2018).

For other models like a latent variable model $p_\theta(x) = \int p_\theta(x|z)p(z)dz$, the log marginal $\log p_\theta(x)$ is typically intractable. Instead, one can derive a tractable upper bound of the reverse KL (Zhang et al., 2019):

$$\mathrm{KL}(p_\theta||p_d) \leq \mathrm{KL}(p_\theta(x|z)p(z)||q_\phi(z|x)p_d(x)), \quad (4)$$

where $q_\phi(z|x)$ is a learnable variational distribution. This reverse KL upper bound contrasts with the more commonly studied forward KL upper bound, as discussed in Wainwright et al. (2008); Kingma and Welling (2013). While this variational approach circumvents the intractability of $\log p_\theta(x)$, it introduces its own challenges and limitations, such as the limited flexibility of the variational family and the potential looseness of the variational bound.

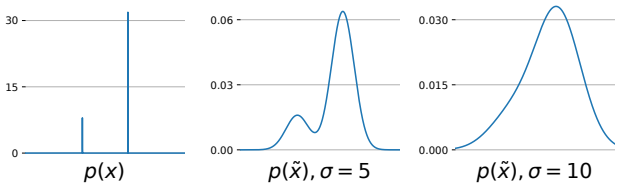


Figure 1: We convolve a Gaussian kernel $\mathcal{N}(\tilde{x}|x, \sigma^2)$ with $\sigma \in \{5, 10\}$ to the original distribution $p(x)$. This demonstrates that Gaussian convolution can bridge modes and even reduce the number of modes as the variance of the Gaussian increases.

Alternatively, Li and Turner (2017); Shi et al. (2017); Song et al. (2020); Luo et al. (2023) directly derive the analytical form of the gradient of the negative entropy for the reverse KL divergence in Equation (3):

$$\nabla_\theta \int \log p_\theta(x) p_\theta(x) dx = \int p_\theta(x) \nabla_x \log p_\theta(x) \frac{\partial x}{\partial \theta} dx, \quad (5)$$

where the score function $\nabla_x \log p_\theta(x)$ of the model can be approximated by training a score network using the model samples with score matching (Hyvärinen, 2005). This enables fitting latent variable models to unnormalized target densities by reverse KL minimization without any variational approximation. However, the mode-seeking behavior of reverse KL typically results in the trained model $p_\theta(x)$ collapsing to a small number of modes in a multi-modal target distribution.

3 DIFFUSIVE KL DIVERGENCE

One effective way to bridge and merge isolated modes is Gaussian convolution, which has been successfully used in training diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2021; Ho et al., 2020) and encouraging exploration of samplers (Lee et al., 2021; Huang et al., 2023; Chen et al., 2024b). It can also potentially reduce the number of modes due to the fact that Gaussian convolution effectively convexifies any functions to its convex envelope (Mobahi and Fisher, 2015). In Figure 1, we provide a toy visualization showing that by increasing the variance of the Gaussian convolution, we can bridge modes or even reduce the number of modes in the original multi-modal distribution.

To construct a valid divergence that leverages Gaussian convolutions, one can convolve two distributions $p(x)$ and $q(x)$ with the same Gaussian kernel $k(\tilde{x}|x) = \mathcal{N}(\tilde{x}|\alpha x, \sigma^2 I)$ and then define the KL divergence between the convolved distributions $\tilde{p}(\tilde{x}) =$

$\int k(\tilde{x}|x)p(x)dx$ and $\tilde{q}(\tilde{x}) = \int k(\tilde{x}|x)q(x)dx$. This type of divergence construction is known as the spread divergence (Zhang et al., 2020).

Definition 3.1 (Spread KL Divergence)**.**

$$\mathrm{SKL}_k(p||q) \equiv \mathrm{KL}(\tilde{p}||\tilde{q}) = \mathrm{KL}(p * k||q * k), \quad (6)$$

where $*$ denotes the convolution operator: $\tilde{\pi} \equiv \pi * k \equiv \int k(\tilde{x}|x)\pi(x)dx$.

The spread KL divergence is a theoretically well-defined divergence as $\mathrm{SKL}_k(p||q) = 0 \Leftrightarrow p = q$ for any Gaussian kernel k, as shown in Zhang et al. (2020). In practice, the choice of k is crucial for model training, and selecting the optimal kernel is a challenging problem. Inspired by the recent success of diffusion models, instead of selecting one k, one can use a sequence of Gaussian kernels with different lengthscales to construct a "multi-level spread KL divergence", which we refer to as *diffusive KL divergence* (DiKL).

Definition 3.2 (Diffusive KL Divergence)**.**

$$\mathrm{DiKL}_{\mathcal{K}}(p||q) \equiv \sum_{t=1}^{T} w(t)\mathrm{KL}(p * k_t||q * k_t), \quad (7)$$

where $w(t)$ is a positive scalar weighting function and $\mathcal{K} = \{k_1, \cdots, k_T\}$ is a set of (scaled) Gaussian convolution kernels denoted as $k_t(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$.

Since the DiKL can be seen as an average of multiple spread KL divergence with different Gaussian kernels, it is straightforward to show that it is a valid divergence, i.e., $\mathrm{DiKL}_{\mathcal{K}}(p||q) = 0 \Leftrightarrow p = q$.

The DiKL has been successfully applied to some important applications such as 3D generative models (Poole et al., 2022; Wang et al., 2024) and diffusion distillation (Luo et al., 2024; Xie et al., 2024). However, in these cases, $p * k_t$ corresponds to a given pre-trained diffusion model. In contrast, our setting only assume access to the unnormalized target density without any samples, and therefore $p * k_t$ is usually intractable. In the next section, we propose a practical gradient estimator of the reverse diffusion KL divergence for training neural samplers to capture diverse modes in unnormalized target densities.

4 TRAINING NEURAL SAMPLERS WITH DIKL

We focus on training neural samplers defined by a latent variable model:

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz, \quad (8)$$

where $p(z) = \mathcal{N}(z|0, I)$ and $p_\theta(x|z)$ is parameterized by a neural network: $p_\theta(x|z) = p(x|g_\theta(z))$.

Unlike the conventional KL divergence, which requires the model $p_\theta(x)$ to have a valid density function (Arjovsky et al., 2017), SKL and DiKL are well-defined even for singular distributions (e.g., delta function), as discussed by Zhang et al. (2020). Therefore, we can let the generator $p_\theta(x|z)$ be a deterministic function g_θ and define the "generalized model density"[1] as:

$$p_\theta(x) = \int \delta(x - g_\theta(z))p(z)dz. \quad (9)$$

This type of model is also referred to as an implicit model (Goodfellow et al., 2014; Huszár, 2017), which is more flexible than the latent variable model defined in Equation (8), as it avoids pre-defining a constrained distribution family for $p(x|g_\theta(z))$.

We now explore how to train such a neural sampler $p_\theta(x)$ to fit the unnormalized target density $p_d(x)$ using the reverse DiKL, denoted as $\mathrm{DiKL}_{\mathcal{K}}(p_\theta||p_d)$. For simplicity, we first consider DiKL with a single kernel k_t; the extension to multiple kernels is straightforward. The reverse DiKL can be expressed as:

$$\begin{aligned}\mathrm{DiKL}_{k_t}(p_\theta||p_d) &\equiv \mathrm{KL}(p_\theta * k_t||p_d * k_t) \\ &= \int p_\theta(x_t)\left(\log p_\theta(x_t) - \log p_d(x_t)\right)dx_t, \end{aligned} \quad (10)$$

where $p_\theta(x_t) = \int k_t(x_t|x)p_\theta(x)dx$ and $p_d(x_t) = \int k_t(x_t|x)p_d(x)dx$[2]. The integration over $p_\theta(x_t)$ can be approximated using Monte Carlo integration. This involves first sampling $x' \sim p_\theta(x)$ and then $x_t \sim k_t(x_t|x')$. Inspired by Poole et al. (2022); Wang et al. (2024); Luo et al. (2024), we can derive the analytical gradient of Equation (10) w.r.t θ as follows:

$$\begin{aligned}\nabla_\theta \mathrm{DiKL}_{k_t}(p_\theta||p_d) &= \nabla_\theta \mathrm{KL}(p_\theta * k_t||p_d * k_t) \\ &= \int p_\theta(x_t)\left(\nabla_{x_t}\log p_\theta(x_t) - \nabla_{x_t}\log p_d(x_t)\right)\frac{\partial x_t}{\partial \theta}dx_t,\end{aligned} \quad (11)$$

The derivation can be found in Appendix A. The Jacobian term $\frac{\partial x_t}{\partial \theta}$ can be efficiently computed by the vector-Jacobian product (VJP) with auto differentiation. However, both score functions, $\nabla_{x_t}\log p_\theta(x_t)$ and $\nabla_{x_t}\log p_d(x_t)$, in Equation (11) are intractable to compute directly. To address this, we approximate these scores using denoising score matching (DSM) (Vincent, 2011) and mixed score identity

[1] In this case, the marginal distribution may not be absolutely continuous (a.c.) w.r.t. the Lebesgue measure, which implies that it may not have a valid density function, e.g., when $\mathrm{Dim}(z) < \mathrm{Dim}(x)$.

[2] To avoid notation overloading, we slightly abuse p_d to represent the density function for both the clean target and the convolved target density $p_d * k_t$. We distinguish them by their arguments: $p_d(x)$ denotes the original target density, while $p_d(x_t)$ refers to the convolved density. The also applies to the kernel k_t and model density p_θ.

(MSI) (De Bortoli et al., 2024; Phillips et al., 2024), respectively. Specifically, we estimate $\nabla_{x_t} \log p_\theta(x_t)$ by training a score network with DSM using samples from the sampler, and estimate $\nabla_{x_t} \log p_d(x_t)$ by MSI with Monte Carlo estimation. Below, we explain these two estimators in detail.

4.1 Estimating $\nabla_{x_t} \log p_\theta(x_t)$ with DSM

Denoising score matching (DSM) (Vincent, 2011) has been successfully used in training score-based diffusion models (Song et al., 2021). DSM is based on the denoising score identity (DSI):

Proposition 4.1 (**Denoising Score Identity**). *For any convolution kernel $k(x_t|x)$, we have*

$$\nabla_{x_t} \log p_\theta(x_t) = \int \nabla_{x_t} \log k(x_t|x) p_\theta(x|x_t) dx, \quad (12)$$

where $p_\theta(x|x_t) \propto k(x_t|x)p_\theta(x)$ is the model posterior.

See Appendix B.1 for a proof. We can then train a time-conditioned score network $s_\phi(x_t)$ to approximate $\nabla_{x_t} \log p_\theta(x_t)$ by minimizing the score matching loss w.r.t. ϕ:

$$\begin{aligned} &\int \| s_\phi(x_t) - \int \nabla_{x_t} \log k(x_t|x) p_\theta(x|x_t) dx \|_2^2 p_\theta(x_t) dx_t \\ &= \iint \| s_\phi(x_t) - \nabla_{x_t} \log k(x_t|x) \|_2^2 p_\theta(x, x_t) dx dx_t + \text{const.}, \end{aligned} \quad (13)$$

where the equivalence can be shown by expanding the L2 norm and ignoring a term that is independent of ϕ. The integration over $p_\theta(x, x_t) = p_\theta(x)k(x_t|x)$ can be approximated by sampling $x' \sim p_\theta(x)$ then $x'_t \sim k(x_t|x')$. Once trained, we plug $s_\phi(x_t)$ into Equation (11) to estimate the gradient.

4.2 Estimating $\nabla_{x_t} \log p_d(x_t)$ with MSI

To estimate the gradient defined in Equation (11), we also need to estimate the noisy target score $\nabla_{x_t} \log p_d(x_t)$. Since no samples from $p_d(x)$ are available, we can no longer use DSM to estimate the score. Fortunately, we have access to the unnormalized target density and its score function $\nabla_x \log p_d(x) = -\nabla_x E(x)$, which allows us to estimate this score by target score identity (TSI, De Bortoli et al., 2024):

Proposition 4.2 (**Target Score Identity**). *For any translation-invariant convolution kernel $k(x_t|x) = k(x_t - \alpha_t x)$, we have*

$$\nabla_{x_t} \log p_d(x_t) = \frac{1}{\alpha_t} \int \nabla_x \log p_d(x) p_d(x|x_t) dx, \quad (14)$$

where $p_d(x|x_t) \propto k(x_t|x)p_d(x)$ is the target posterior.

See Appendix B.2 for a proof. In practice, the TSI estimator has larger variance when the Gaussian kernel $k(x_t|x)$ has larger variance, while the DSI estimator exhibits higher variance when $k(x_t|t)$ has smaller variance. To address this, De Bortoli et al. (2024); Phillips et al. (2024) propose a convex combination of the DSI and TSI to interpolate between them, favoring TSI when $k(x_t|x)$ has smaller variance and DSI when $k(x_t|x)$ has larger variance, thus minimizing the overall variance of the estimator. We refer to this estimator as the mixed score identity (MSI).

Proposition 4.3 (**Mixed Score Identity**). *Using a Gaussian convolution $k(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$ with a variance-preserving (VP) scheme $\sigma_t^2 = 1 - \alpha_t^2$, and a convex combination of TSI and DSI with coefficients α_t^2 and $1 - \alpha_t^2$, respectively, we have*

$$\begin{aligned} &\nabla_{x_t} \log p_d(x_t) \\ &= \int (\alpha_t(x + \nabla_x \log p_d(x)) - x_t) p_d(x|x_t) dx. \end{aligned} \quad (15)$$

See Appendix B.3 for a proof. This identity estimates the score $\nabla_{x_t} \log p(x_t)$ based on the original target score $\nabla_x \log p_d(x)$ which can be directly evaluated. We can plug it into Equation (11) as part of the gradient approximation.

However, to use this estimator, we also need to obtain samples from the denoising posterior $p_d(x|x_t)$ to approximate the integration over x in Equation (15). We notice that the posterior $p_d(x|x_t)$ is proportional to the joint $p(x, x_t) = k(x_t|x)p(x)$, taking the form

$$p_d(x|x_t) \propto \exp\left(-E(x) - \|\alpha_t x - x_t\|^2 / 2\sigma_t^2\right), \quad (16)$$

which has a tractable score function (Gao et al., 2020; Huang et al., 2023; Chen et al., 2024b; Grenioux et al., 2024):

$$\nabla_x \log p_d(x|x_t) = -\nabla_x E(x) - \frac{\alpha_t(\alpha_t x - x_t)}{\sigma_t^2}. \quad (17)$$

Therefore, common score-based sampler such as HMC (Duane et al., 1987; Neal et al., 2011), MALA (Roberts and Tweedie, 1996; Roberts and Stramer, 2002), and AIS (Neal, 2001) can be directly employed to sample from the denoising posterior $p(x|x_t)$. Notably, compared to sampling from the original target distribution $p(x) \propto \exp(-E(x))$ using standard score-based samplers, incorporating the additional quadratic term in Equation (16) improves the Log-Sobolev conditions, which significantly enhances the convergence speed of samplers like ULA (Vempala and Wibisono, 2019; Huang et al., 2023).

It is worth noting that it is not crucial to have a perfect posterior sampler. This is because our method

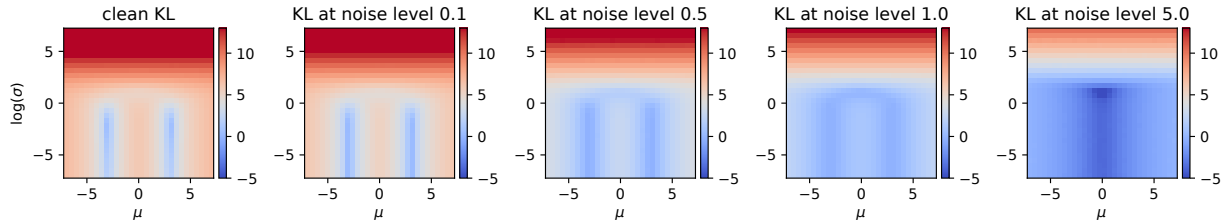


Figure 2: Heatmap of (log scale) KL divergence at different noise levels between a Gaussian model (with mean parameter μ and standard deviation parameter σ) and a two-mode MoG target in 1D. At lower noise levels (or in the extreme case, the standard reverse KL), the divergence is highly mode-seeking, with the model favoring either one of the two modes in the target distribution. However, perhaps surprisingly, the KL divergence becomes more mass-covering at a higher noise level, encouraging the model to cover both modes of the target.

essentially works in a bootstrapping manner: the posterior samples improve the model, and a better model in turn brings the posterior samples closer to the true target. Having said that, accurate posterior sampling may improve the convergence rate of the model.

We summarize the whole procedure of training neural samplers with DiKL in Algorithm 1[3]. In short, our training algorithm forms a nested loop: in the inner loop, we train a score network $s_\phi(x_t)$ to estimate the model score $\nabla_{x_t} \log p_\theta(x_t)$ with DSM; in the outer loop, we first estimate the noisy target score $\nabla_{x_t} \log p_d(x_t)$ with MSI, and then update the neural sampler with the gradient as in Equation (11) using our estimated noisy target and model scores. One might think that this nested training procedure imposes a high computational burden. Fortunately, we found that the inner loop typically converged within 50-100 steps in practice, minimally affecting the overall training cost. In the following, we give an empirical illustration of how DiKL can encourage mode covering.

Algorithm 1 Training Neural Samplers with DiKL

Input: Target $p_d(x) \propto \exp(-E(x))$, Gaussian kernels $\{(\alpha_t, \sigma_t)\}_{t=1}^T$; score network training step N_ϕ; weighting function $w(t)$; Randomly initialized θ, ϕ.
repeat
  # Train the score network $s_\phi(x_t)$ by DSM:
  for $i \in [1, \cdots, N_\phi]$ **do**
    $z \sim p(z)$, $x \leftarrow g_\theta(z)$, $t \sim \mathcal{U}\{1, \cdots, T\}$
    $\epsilon \sim \mathcal{N}(0, I)$, $x_t \leftarrow \alpha_t x + \sigma_t \epsilon$
    Update ϕ with $\nabla_\phi \| s_\phi(x_t) - \nabla_{x_t} \log k(x_t|x) \|_2^2$
  end for
  # Train the neural sampler $g_\theta(z)$ by DiKL:
  $z \sim p(z)$, $x \leftarrow g_\theta(z)$
  $t \sim \mathcal{U}\{1, \cdots, T\}$, $\epsilon \sim \mathcal{N}(0, I)$, $x_t \leftarrow \alpha_t x + \sigma_t \epsilon$
  $x'^{(1:K)} \sim p_d(x|x_t)$ ▷ posterior sampling
  $d_p \leftarrow \frac{1}{K} \sum_{k=1}^K (\alpha_t (x'^{(k)} + \nabla \log p_d(x'^{(k)})) - x_t)$ ▷ MSI estimator
  $\ell \leftarrow w(t) \texttt{stopgrad}(s_\phi(x_t) - d_p)^\top x_t$ ▷ surrogate loss for VJP
  Update θ with $\nabla_\theta \ell$
until convergence

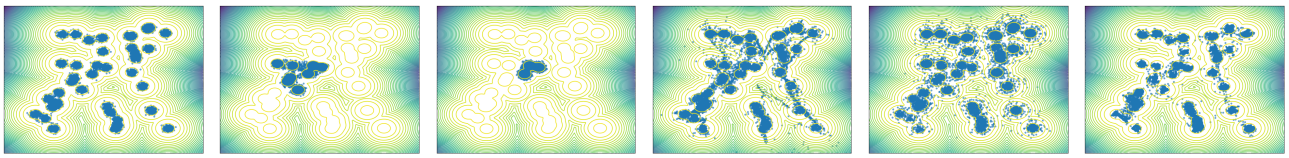
4.3 DiKL Encourages mass-covering

Unlike the mode-seeking nature of reverse KL (R-KL), DiKL promotes better mode coverage. In this section, we provide an intuitive explanation to illustrate how this is achieved. Assume we have a 1D Mixture of Gaussian (MoG) target with two components $p_d(x) = \frac{1}{2}\mathcal{N}(x|-3, 0.01) + \frac{1}{2}\mathcal{N}(x|3, 0.01)$. For simplicity, we fit a 1D Gaussian model $p_\theta(x) = \mathcal{N}(x|\mu, \sigma^2)$ to this target. As this model only contains two parameters $\theta = \{\mu, \sigma\}$, we visualize log KL and log DiKL at different noise levels against these two parameters to develop a better understanding of the reverse DiKL objective, as shown in Figure 2.

At lower noise levels (or in the extreme case, R-KL), the divergence is highly mode-seeking, with the model favoring either one of the two modes in the target distribution. However, perhaps surprisingly, at higher noise levels, DiKL becomes more mass-covering, forcing μ to converge toward the mean of the two modes and σ to cover both modes. This behavior explains why DiKL encourages the model to cover more modes: higher noise levels push the model to explore adjacent modes, while lower noise levels prevent the model from forgetting previously discovered modes.

Below, we demonstrate this mass-covering property on a MoG-40 target before proceeding to more complex Boltzmann distributions. Before presenting the results, we first outline other approaches to encouraging mode coverage, which we will use as baselines.

[3] Algorithm 1 presents the training procedure with a batch size of 1 for clarity.



(a) Ground Truth (b) R-KL SM (c) R-KL Bound (d) FAB (e) iDEM (f) DiKL (ours)

Figure 3: Samples on MoG-40. We train each method for 2.5 hours, which allows all to converge. FAB and iDEM use replay buffers as in Midgley et al. (2023); Akhound-Sadegh et al. (2024). The high-density regions of this target are within $[-50, 50]$. All methods were trained on the original scale, except for iDEM, which is normalized to $[-1, 1]$ following Akhound-Sadegh et al. (2024). This normalization may simplify the task.

Table 1: Comparison of the log-density of samples generated by various methods, evaluated on the target density of MoG-40. "True" indicates the log density of true samples from the target distribution. We only report the evaluation methods that can cover all the modes, see Figure 3 for the sample visualization.

	True	FAB	iDEM	DiKL (Ours)
$\log p_d(x)$	*-6.85*	-10.74	-8.33	**-7.21**

5 RELATED WORKS

Several different types of neural samplers have been proposed in the literature, which we summarize below.

Latent-variable-model samplers. Training latent variable models as neural samplers with Fisher divergence, reverse KL (R-KL-SM) or its upper bound (R-KL Bound) has been explored in the literature (Li and Turner, 2017; Shi et al., 2017; Zhang et al., 2019; Song et al., 2020; Luo et al., 2023; Yin and Zhou, 2018; Hu et al., 2018); See Equations (3) to (5). Such methods typically struggle for multi-modal target distributions.

Flow-based samplers. Flow AIS bootstrap (FAB) (Midgley et al., 2023) is the state-of-the-art (SOTA) flow-based sampler, which is trained by minimizing the α-2 divergence, which exhibits mass covering property. Note that FAB employs a prioritized replay buffer to memorize the regions that have been explored.

Diffusion/control-based samplers. Several works have explored diffusion/control-based samplers. For example, the Gibbs-style sampler (Grenioux et al., 2024; Chen et al., 2024b; Zhang et al., 2023) constructs a forward-backward sampling procedure between the clean data space and the diffusion space. The path integral sampler (PIS, Zhang and Chen, 2022) and the denoising diffusion sampler (DDS, Vargas et al., 2023) align the forward and backward paths by optimizing the KL divergence over the entire path measure. GFlowNet-based sampler (Bengio et al., 2023; Zhang et al., 2024) extends these approaches to objectives with local information, like sub-trajectory balance and detailed balance. Controlled Monte Carlo diffusions (CMCD, Nusken et al., 2024) learns an escorted transport between interpolants from the prior distribution to the target distribution by matching the KL or log-variance divergence between the forward and backward path measures. Non-equilibrium transport sampler (NETS, Albergo and Vanden-Eijnden, 2024) learns a similar escorted transport with PINN (Sun et al., 2024) or Action Matching (Neklyudov et al., 2023) loss. Further improvements including combining these samplers with SMC (Chen et al., 2024a), or incorporating MCMC to improve buffer samplers (Sendera et al., 2024). However, these methods typically involve simulating the SDE by numerical integration during training, which is not scalable. Iterated denoising energy matching (iDEM, Akhound-Sadegh et al., 2024) is one of the SOTA samplers that trains a score network to approximate the noisy score of the target estimated by TSI. iDEM also employs a reply buffer to balance exploration and exploitation.

We compare our methods with each type of SOTA neural sampler on a mixture of 40 Gaussians (MoG-40) target in 2D following Midgley et al. (2023), which allows us to visually examine their mass-covering properties. As shown in Figure 3, our approach achieves better sample quality than all other compared neural samplers. R-KL-based samplers struggle to capture the majority of modes due to the mode-seeking property. FAB captures all modes but exhibits heavy density connections between modes due to the flow architecture. This is in contrast to our sampler which only requires using standard neural networks, which is more flexible. As for iDEM, while it does not exhibit such connections, its samples look noisy. This is because iDEM requires score estimation across all noise levels from the target towards a pure Gaussian distribution, which leads to high variance at larger noise levels due to TSI. In contrast, our approach samples directly from a generator g_θ and uses a Gaussian kernel only to connect adjacent modes, allowing for a much smaller noise level and more manageable variance.

Distillation for Diffusion Models. Variational score distillation (VSD) is a promising approach to distill knowledge from pre-trained diffusion models. The concept of DiKL has been successfully applied in these approaches for 3D generative models (Poole et al., 2022; Wang et al., 2024) and diffusion distillation (Luo et al., 2024; Xie et al., 2024). Different from our application, they have a pre-trained diffusion model to provide the score of $p * k_t$. On the other hand, KL-based neural samplers (Li and Turner, 2017; Shi et al., 2017; Luo et al., 2023) provide a KL estimator when there is no Gaussian convolution. From this perspective, we can see that our approach lies conceptually between VSD and KL-based neural samplers.

6 APPLICATION TO BOLTZMANN GENERATORS

One important application of neural samplers is to generate samples from Boltzmann distributions, where the target distribution defines the probability density that a system will be in a certain state as a function of that state's energy and the temperature of the system. This type of neural sampler is also known as Boltzmann Generator (Noé et al., 2019). In the following, we will omit the temperature for simplicity, as it can be absorbed into the energy function.

In this section, we use our neural sampler g_θ as a Boltzmann Generator to generate samples from n-body systems[4], where the energy is defined over the pairwise distances between n particles. These systems can be defined in either internal or Cartesian coordinates. Note that, for Cartesian coordinates, the energy of the system will remain invariant if we apply rotation, reflection, translation, and permutation to the entire system. Formally, representing each configuration of the system by a matrix $X \in \mathbb{R}^{n\times d}$, our target distribution $p_d(X)$ is invariant to the product group of the Euclidean group and the Symmetric group of degree n, i.e. $G = \mathrm{E}(d) \times \mathbb{S}_n$.

This invariance presents a challenge when training the neural sampler. Recall the sampler learns a mapping $g_\theta : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X}$ represents the space of system configurations, and $\mathcal{Z}$ represents the latent space. If $\mathcal{X}$ includes configurations with symmetries but $\mathcal{Z}$ does not account for these symmetries, the network would need to model every equivariant configuration separately (for example, the model would need to assign same density for the configurations in one equivariant class respect to G), leading to inefficient training.

On the other hand, we can parameterize the neural sampler g_θ with an Equivariant Graph Neural Networks (EGNN, Satorras et al., 2021; Hoogeboom et al., 2022), ensuring that g_θ is G-equivariant.

Proposition 6.1. *Let the neural sampler $g_\theta : \mathcal{Z} \to \mathcal{X}$ be an G-equivariant mapping. If the distribution $p(Z)$ over the latent space $\mathcal{Z}$ is G-invariant, then $p_\theta(X) = \int \delta(X - g_\theta(Z))p(Z)dZ$ is G-invariant.*

The proof can be found in Appendix C.1. Therefore, our neural sampler does not need to explicitly model the invariance, simplifying the training process.

However, another challenge arises from translation. As noted by Midgley et al. (2024), there does not exist a translation invariant probability measure in Euclidean space. Therefore, following Midgley et al. (2024); Satorras et al. (2021); Hoogeboom et al. (2022); Akhound-Sadegh et al. (2024), we constrain both $\mathcal{X}$ and $\mathcal{Z}$ to be the subspace of $\mathbb{R}^{n\times d}$ with zero center of mass, i.e., $X^\top 1 = Z^\top 1 = 0$. This allows us to embed the product group in $n \times d$ into an orthogonal group in nd-dimensional space: $\mathrm{E}(d) \times \mathbb{S}_n \hookrightarrow \mathrm{O}(nd)$.

Having decided on the architecture for the neural sampler g_θ and tackled the translation invariance, we now consider the scoring network s_ϕ for the neural sampler. According to Papamakarios et al. (2021, Lemma 2), *if G is a subgroup of the orthogonal group, then the gradient of a G-invariant function is G-equivariant.* Therefore, the score network for the neural sampler which defines the G-invariant distribution is G-equivariant. To achieve this, we train an EGNN score network with denoising score matching (DSM) within the zero-centered subspace, following Hoogeboom et al. (2022).

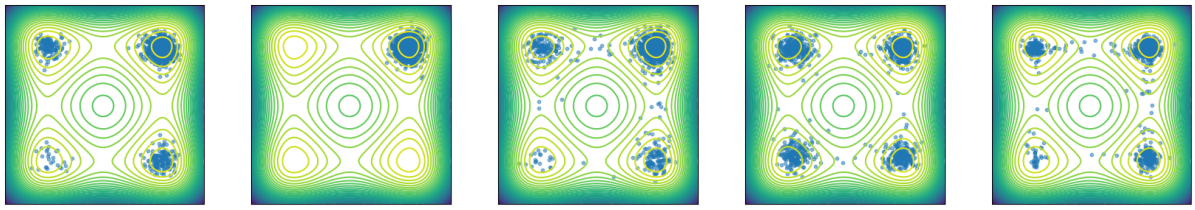
Additionally, when both the model and target density are G-invariant, the identity $\nabla \log p_\theta(X_t) - \nabla \log p_d(X_t)$ in the gradient of reverse DiKL as shown in Equation (11) should also be G-equivariant. This necessitates a G-equivariant MSI estimator for $\int(\alpha_t(X+\nabla \log p_d(X))-X_t)p_d(X|X_t)dx$. Fortunately, this holds true for a broad class of estimators under mild conditions, including importance sampling and AIS estimators, with different choices of samplers for $p_d(X|X_t)$, such as MALA and HMC. Detailed discussion can be found in Appendix C.

6.1 Experiments and Results

We evaluate our approach on three distinct Boltzmann distributions: Many-Well-32 in the internal coordinate, and Double-Well-4 and Lennard-Jones-13 in the Cartesian coordinate. These tasks aim to provide a comprehensive evaluation of highly multi-modal targets, encompassing both balanced and imbalanced modes, as well as energy functions exhibiting invariance. Detailed setups can be found in Appendix D.

Internal MW-32. Midgley et al. (2023) introduced

[4]Code for all experiments is available at `https://github.com/jiajunhe98/DiKL`.



(a) Ground Truth (b) KL (c) FAB (d) iDEM (e) DiKL (ours)

Figure 4: 2D marginal (1st and 3rd dimensions) of samples from MW-32. Our approach and FAB manage to find all the modes with correct weights, iDEM finds all modes but with wrong weights, and the neural sampler trained with standard KL divergence only capture one mode.

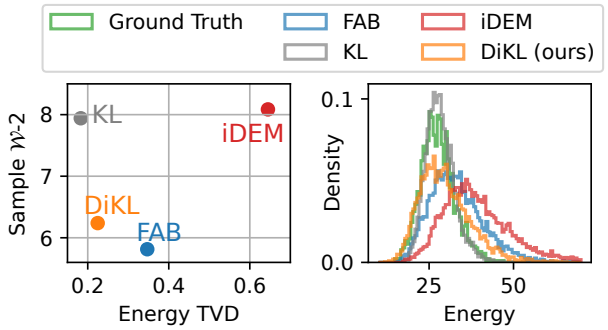
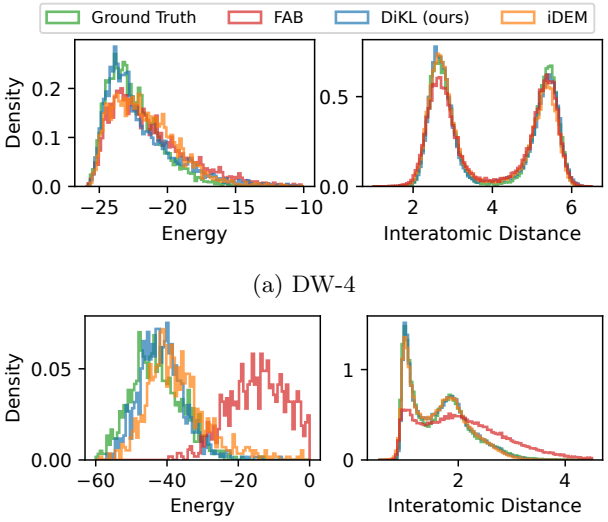


Figure 5: **Left.** Wasserstein-2 ($\mathcal{W}$-2) distance of samples and total variation distance (TVD) of energy on MW-32. Our method and FAB clearly outperform iDEM and KL in this evaluation. **Right.** Histogram of sample energy. Our approach outperforms both FAB and iDEM. Note that although the KL approach yields better energy, it captures only one mode, as shown in Figure 4.

this target by stacking 2D Double-Well 32 times, forming a distribution with 2^{32} modes in total. These modes carry different weights. Therefore, this task can assess whether each method successfully covers all modes and accurately captures their weights.

We report the Wasserstein-2 ($\mathcal{W}$-2) distances between samples yielded by these methods and ground truth samples obtained by MCMC. We also evaluate the sample energy and report the total-variant distance (TVD) between the distribution of the energy of samples. We note that both metrics have their limitations: $\mathcal{W}$-2 tends to be less sensitive to noisy samples, which can be particularly detrimental in some n-body systems. Conversely, the energy TVD is less sensitive to missing modes. To provide a comprehensive evaluation, we plot both metrics together in Figure 5, which shows that our method and FAB clearly outperform iDEM and KL in this evaluation. We also visualize the samples along two selected axes to assess mode coverage in Figure 4, showing that only our approach and



(a) DW-4

(b) LJ-13

Figure 6: Histogram of sample energy and interatomic distance on Cartesian DW-4 and LJ-13. Our approach achieves comparable performance on both tasks to iDEM, with only 1 number of function evaluation (NFE), while iDEM requires 1,000 NFEs.

FAB can find all modes with roughly correct weights.

Cartesian DW-4 and LJ-13. Köhler et al. (2020) introduced these two tasks to access the model under invariance target distributions. Specifically, the target energy is invariant to the product group $G = \mathrm{E}(d) \times \mathbb{S}_n$ where $d = 2$ and 3 for DW and LJ, respectively.

We compare our approach with FAB and iDEM. We report the $\mathcal{W}$-2 distance for samples, TVD for energy, and TVD for interatomic distance in Table 2. We also visualize histograms for sample energy and interatomic distance in Figure 6. As shown, our method achieves competitive performance with both FAB and iDEM on DW-4, and notably outperforms FAB on LJ-13.

Training and sampling time. We report the training and sampling times for our approach and baselines

Table 2: Comparison of our approach with FAB and iDEM on Cartisian DW-4 and LJ-13. We report the Wasserstein-2 ($\mathcal{W}$-2) distance of samples, and total variation distances (TVDs) of energy and atomic distances. We evaluate each metric using 5,000 samples repeated ten times and report the mean and standard deviation.

	Cartesian DW-4 (8D)			**Cartesian LJ-13 (39D)**		
	Sample $\mathcal{W}$-2	Energy TVD	Distance TVD	Sample $\mathcal{W}$-2	Energy TVD	Distance TVD
FAB	**1.554 ± 0.015**	0.224 ± 0.008	**0.097 ± 0.005**	4.938 ± 0.009	0.902 ± 0.010	0.252 ± 0.002
iDEM	1.593 ± 0.012	0.197 ± 0.010	0.103 ± 0.005	**4.172 ± 0.007**	0.306 ± 0.013	0.044 ± 0.001
DiKL (ours)	1.581 ± 0.026	**0.167 ± 0.012**	0.101 ± 0.006	4.233 ± 0.008	**0.239 ± 0.019**	**0.042 ± 0.002**

Table 3: Training and sampling wall-clock times for FAB, iDEM and our sampler. We measure this on a single NVIDIA A100 (80GB) GPU. We omit the sampling times for FAB on DW-4 and LJ13 as it is implemented in JAX with JIT compilation, making direct comparison with the other methods implemented in PyTorch not feasible. However, we expect FAB to have slightly slower sampling times than our method due to its large flow network.

		FAB	**iDEM**	**DiKL (ours)**
Training	MW-32	3.5h	3.5h	**2.5h**
	DW-4	4.5h	4.5h	**0.9h**
	LJ-13	21.5h	**6.5h**	**6.5h**
Batch Sampling (1,000 samples)	MW-32	**0.01s**	7.2s	**0.01s**
	DW-4	-	2.6s	**0.01s**
	LJ-13	-	19.7s	**0.02s**

in Table 3. Notably, our method shows faster training and sampling times compared to both FAB and iDEM. FAB depends on a large and limited normalizing flow, which leads to significantly longer training times, particularly for complex tasks like LJ. In contrast, our approach maintains consistent training times across different tasks. iDEM is diffusion-based and requires intensive computation for sampling, which is 1,000 times slower than our approach.

Discussion. Our method is comparable to SOTA Boltzmann generators, including FAB and iDEM, and has both faster training and sampling times. Additionally, we highlight that, unlike iDEM and FAB, which use replay buffers for exploration-exploitation balance, our approach achieves this *without* relying on any replay buffer, offering a more clean, straightforward and easy-to-extend solution.

7 CONCLUSION

In this work, we proposed a new training paradigm for neural samplers using reverse diffusive KL divergence, providing a simple yet efficient method to achieve the mass-covering property. We demonstrated its effectiveness on both synthetic and n-body system targets. Our approach matches or outperforms SOTA methods like FAB and iDEM, with improved training and sampling efficiency and without relying on replay buffer.

While our method achieves comparable performance with SOTA flow-based and diffusion-based samplers with faster training and sampling speed, it has a few limitations as discussed below.

Model density. The density $p_\theta(x)$ of our neural sampler is intractable since we have a latent variable model with a nonlinear and non-invertible generator $g_\theta(z)$. Compared to FAB, although we have a more flexible generator, we cannot use importance re-weighting to correct the potential bias of generated samples since we do not have access to the density of our samples.

Model flexibility. Although our one-step generator $g_\theta(z)$ has significantly faster sampling speed than diffusion-based samplers, it has limited model flexibility compared to multi-step diffusion models such as iDEM. It is, therefore, more difficult for our model to handle more complicated energy functions. For example, DiKL cannot capture more complicated target LJ-55 well, as shown in Appendix E.3. Therefore, a promising direction is to design approaches to balance the sampling speed and model expressiveness.

Training stability. For the two n-body system targets in the Cartesian coordinate (i.e., DW-4 and LJ-13), training can be unstable near convergence. We employed some criteria on the energies of model samples to perform early stopping. For future work, it might be beneficial to employ a replay buffer similar to the one used in FAB or iDEM to balance exploration and exploitation and stabilize training.

Posterior sampling. Posterior sampling could be a bottleneck in our training procedure, which requires running MCMC to sample from the denoising posterior distribution $p_d(x|x_t)$ at each training iteration. The MCMC sampling procedure might have a slow mixing speed for complicated target distributions. In fact, we note that most diffusion-based samplers have this bottleneck since there is no data available to train a neural network denoiser, unlike diffusion or score-based generative models.

Acknowledgments

JH is supported by the University of Cambridge Harding Distinguished Postgraduate Scholars Programme. JH and JMHL acknowledge support from a Turing AI Fellowship under grant EP/V023756/1. MZ and DB acknowledge funding from AI Hub in Generative Models, under grant EP/Y028805/1.

Part of this work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (`www.csd3.cam.ac.uk`), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (`www.dirac.ac.uk`).

References

Akhound-Sadegh, T., Rector-Brooks, J., Bose, J., Mittal, S., Lemos, P., Liu, C.-H., Sendera, M., Ravanbakhsh, S., Gidel, G., Bengio, Y., et al. (2024). Iterated denoising energy matching for sampling from boltzmann densities. In *Forty-first International Conference on Machine Learning*.

Albergo, M. S. and Vanden-Eijnden, E. (2024). Nets: A non-equilibrium transport sampler. *arXiv preprint arXiv:2410.02711*.

Arbel, M., Matthews, A., and Doucet, A. (2021). Annealed flow transport monte carlo. In *International Conference on Machine Learning*, pages 318–330. PMLR.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023). Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Chen, J., Richter, L., Berner, J., Blessing, D., Neumann, G., and Anandkumar, A. (2024a). Sequential controlled langevin diffusions. *arXiv preprint arXiv:2412.07081*.

Chen, W., Zhang, M., Paige, B., Hernández-Lobato, J. M., and Barber, D. (2024b). Diffusive gibbs sampling. In *International Conference on Machine Learning*, pages 7731–7747. PMLR.

De Bortoli, V., Hutchinson, M., Wirnsberger, P., and Doucet, A. (2024). Target score matching. *arXiv preprint arXiv:2402.08667*.

di Langosco, L. L., Fortuin, V., and Strathmann, H. (2021). Neural variational gradient descent. *arXiv preprint arXiv:2107.10731*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222.

Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. (2020). Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Grenioux, L., Noble, M., Gabrié, M., and Durmus, A. O. (2024). Stochastic localization via iterative posterior sampling. In *International Conference on Machine Learning*, pages 16337–16376. PMLR.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR.

Hu, T., Chen, Z., Sun, H., Bai, J., Ye, M., and Cheng, G. (2018). Stein neural sampler. *arXiv preprint arXiv:1810.03545*.

Huang, X., Hanze Dong, Y. H., Ma, Y., and Zhang, T. (2023). Reverse Diffusion Monte Carlo. *arXiv preprint arXiv:2307.02037*.

Huszár, F. (2017). Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*.

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

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Köhler, J., Klein, L., and Noé, F. (2020). Equivariant flows: exact likelihood generative learning for symmetric densities. In *International conference on machine learning*, pages 5361–5370. PMLR.

Lee, Y. T., Shen, R., and Tian, K. (2021). Structured logconcave sampling with a restricted gaussian oracle. In *Conference on Learning Theory*, pages 2993–3050. PMLR.

Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. (2017). Generalizing hamiltonian monte carlo with neural networks. *arXiv preprint arXiv:1711.09268*.

Li, Y. and Turner, R. E. (2017). Gradient estimators for implicit models. *arXiv preprint arXiv:1705.07107*.

Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., and Zhang, Z. (2024). Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36.

Luo, W., Zhang, B., and Zhang, Z. (2023). Entropy-based training methods for scalable neural implicit samplers. *Advances in Neural Information Processing Systems*, 36.

Midgley, L., Stimper, V., Antorán, J., Mathieu, E., Schölkopf, B., and Hernández-Lobato, J. M. (2024). Se (3) equivariant augmented coupling flows. *Advances in Neural Information Processing Systems*, 36.

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

Mobahi, H. and Fisher, J. W. (2015). On the link between gaussian homotopy continuation and convex envelopes. In *Energy Minimization Methods in Computer Vision and Pattern Recognition: 10th International Conference, EMMCVPR 2015, Hong Kong, China, January 13-16, 2015. Proceedings 10*, pages 43–56. Springer.

Moore, J. H., Cole, D. J., and Csanyi, G. (2024). Computing hydration free energies of small molecules with first principles accuracy. *arXiv preprint arXiv:2405.18171*.

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

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

Neklyudov, K., Brekelmans, R., Severo, D., and Makhzani, A. (2023). Action matching: Learning stochastic dynamics from samples. In *International conference on machine learning*, pages 25858–25889. PMLR.

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

Nusken, N., Vargas, F., Padhy, S., and Blessing, D. (2024). Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations: ICLR 2024*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Phillips, A., Dau, H.-D., Hutchinson, M. J., De Bortoli, V., Deligiannidis, G., and Doucet, A. (2024). Particle denoising diffusion sampler. *arXiv preprint arXiv:2402.06320*.

Pompe, E., Holmes, C., and Łatuszyński, K. (2020). A framework for adaptive MCMC targeting multimodal distributions. *The Annals of Statistics*, 48(5):2930–2952.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. (2022). Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*.

Rezende, D. J., Papamakarios, G., Racanière, S., Albergo, M. S., Kanwar, G., Shanahan, P. E., and Cranmer, K. (2020). Normalizing flows on tori and spheres. *arXiv preprint arXiv:2002.02428*.

Roberts, G. O. and Stramer, O. (2002). Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4:337–357.

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

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR.

Sendera, M., Kim, M., Mittal, S., Lemos, P., Scimeca, L., Rector-Brooks, J., Adam, A., Bengio, Y., and Malkin, N. (2024). Improved off-policy training of diffusion samplers. *Advances in Neural Information Processing Systems*, 37:81016–81045.

Shi, J., Sun, S., and Zhu, J. (2017). Kernel implicit variational inference. *arXiv preprint arXiv:1705.10119*.

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

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

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

Sun, J., Berner, J., Richter, L., Zeinhofer, M., Müller, J., Azizzadenesheli, K., and Anandkumar, A. (2024). Dynamical measure transport and neural pde solvers for sampling. *arXiv preprint arXiv:2407.07873*.

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

Vempala, S. and Wibisono, A. (2019). Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. *Advances in neural information processing systems*, 32.

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

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

Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., and Zhu, J. (2024). Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

Wu, H., Köhler, J., and Noé, F. (2020). Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944.

Xie, S., Xiao, Z., Kingma, D. P., Hou, T., Wu, Y. N., Murphy, K. P., Salimans, T., Poole, B., and Gao, R. (2024). Em distillation for one-step diffusion models. *arXiv preprint arXiv:2405.16852*.

Yin, M. and Zhou, M. (2018). Semi-implicit variational inference. In *International conference on machine learning*, pages 5660–5669. PMLR.

Zhang, D., Chen, R. T., Liu, C.-H., Courville, A., and Bengio, Y. (2024). Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *The Twelfth International Conference on Learning Representations*.

Zhang, M., Bird, T., Habib, R., Xu, T., and Barber, D. (2019). Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*.

Zhang, M., Hawkins-Hooker, A., Paige, B., and Barber, D. (2023). Moment matching denoising gibbs sampling. *Advances in Neural Information Processing Systems*, 36:23590–23606.

Zhang, M., Hayes, P., Bird, T., Habib, R., and Barber, D. (2020). Spread divergence. In *International Conference on Machine Learning*, pages 11106–11116. PMLR.

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

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Code for all experiments is available at `https://github.com/jiajunhe98/DiKL`.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes] The proof can be found in Appendix.
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] Code for all experiments is available at `https://github.com/jiajunhe98/DiKL`.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] We provide the training details and hyperparameters in Appendix.

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes]

(b) The license information of the assets, if applicable. [Yes]

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Training Neural Samplers with Reverse Diffusive KL Divergence: Supplementary Materials

Contents

A DERIVATION OF ANALYTICAL GRADIENT FOR REVERSE DIKL

The gradient of reverse DiKL w.r.t. the model parameter θ is given by

$$\nabla_\theta \mathrm{DiKL}_{k_t}(p_\theta||p_d) = \int p_\theta(x_t) \left(\nabla_{x_t} \log p_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)\right) \frac{\partial x_t}{\partial \theta} dx_t. \tag{18}$$

Proof. The reverse DiKL at time t is defined as

$$\mathrm{DiKL}_{k_t}(p_\theta||p_d) = \int \left(\log p_\theta(x_t) - \log p_d(x_t)\right) p_\theta(x_t) dx_t. \tag{19}$$

We first reparameterize x_t as a function of z and ϵ:

$$x_t = \alpha_t g_\theta(z) + \sigma_t \epsilon_t \equiv h_\theta(z, \epsilon_t), \tag{20}$$

where $z \sim p(z) \equiv \mathcal{N}(z|0, I)$ and $\epsilon_t \sim p(\epsilon_t) \equiv \mathcal{N}(\epsilon_t|0, I)$. It then follows that

$$\begin{aligned}
\nabla_\theta \mathrm{DiKL}_{k_t}(p_\theta||p_d) &= \nabla_\theta \int \left(\log p_\theta(x_t) - \log p_d(x_t)\right) p_\theta(x_t) dx_t && (21)\\
&= \nabla_\theta \iiint \left(\log p_\theta(x_t) - \log p_d(x_t)\right) \delta(x_t - h_\theta(z, \epsilon_t)) p(z) p(\epsilon_t) dx_t dz d\epsilon && (22)\\
&= \nabla_\theta \iint \left(\log p_\theta(x_t) - \log p_d(x_t)\right) |_{x_t = h_\theta(z,\epsilon_t)} p(z) p(\epsilon_t) dz d\epsilon && (23)\\
&= \iint \left(\nabla_\theta \log p_\theta(x_t) + \nabla_{x_t} \log p_\theta(x_t) \frac{\partial x_t}{\partial \theta} - \nabla_{x_t} \log p_d(x_t) \frac{\partial x_t}{\partial \theta}\right)\Big|_{x_t = h_\theta(z,\epsilon_t)} p(z) p(\epsilon_t) dz d\epsilon && (24)\\
&= \int \left(\nabla_\theta \log p_\theta(x_t) + \nabla_{x_t} \log p_\theta(x_t) \frac{\partial x_t}{\partial \theta} - \nabla_{x_t} \log p_d(x_t) \frac{\partial x_t}{\partial \theta}\right) p_\theta(x_t) dx_t && (25)\\
&= \int \left(\nabla_{x_t} \log p_\theta(x_t) \frac{\partial x_t}{\partial \theta} - \nabla_{x_t} \log p_d(x_t) \frac{\partial x_t}{\partial \theta}\right) p_\theta(x_t) dx_t, && (26)
\end{aligned}$$

where the last line follows since

$$\int \nabla_\theta \log p_\theta(x_t) p_\theta(x_t) dx_t = \int \nabla_\theta p_\theta(x_t) dx_t = \nabla_\theta \int p_\theta(x_t) dx_t = \nabla_\theta 1 = 0. \tag{27}$$

This completes the proof. □

B DERIVATIONS OF SCORE IDENTITIES

B.1 Derivation of Denoising Score Identity (DSI)

Proposition 4.1 (**Denoising Score Identity**). *For any convolution kernel $k(x_t|x)$, we have*

$$\nabla_{x_t} \log p_\theta(x_t) = \int \nabla_{x_t} \log k(x_t|x) p_\theta(x|x_t) dx, \tag{28}$$

where $p_\theta(x|x_t) \propto k(x_t|x) p_\theta(x)$ is the model posterior.

Proof. It follows that

$$\nabla_{x_t} \log p_\theta(x_t) = \frac{\nabla_{x_t} p_\theta(x_t)}{p_\theta(x_t)} \tag{29}$$

$$= \frac{\nabla_{x_t} \int k(x_t|x) p_\theta(x) dx}{p_\theta(x_t)} \tag{30}$$

$$= \frac{\int \nabla_{x_t} k(x_t|x) p_\theta(x) dx}{p_\theta(x_t)} \tag{31}$$

$$= \frac{\int \nabla_{x_t} \log k(x_t|x) k(x_t|x) p_\theta(x) dx}{p_\theta(x_t)} \tag{32}$$

$$= \int \nabla_{x_t} \log k(x_t|x) p_\theta(x|x_t) dx. \tag{33}$$

Note that the same argument can be used to derive DSI for the target distribution $p_d(x)$:

$$\nabla_{x_t} \log p_d(x_t) = \int \nabla_{x_t} \log k(x_t|x) p_d(x|x_t) dx. \tag{34}$$

□

B.2 Derivation of Target Score Identity (TSI)

Proposition 4.2 (**Target Score Identity**). *For any translation-invariant convolution kernel $k(x_t|x) = k(x_t - \alpha_t x)$, we have*

$$\nabla_{x_t} \log p_d(x_t) = \frac{1}{\alpha_t} \int \nabla_x \log p_d(x) p_d(x|x_t) dx, \tag{35}$$

where $p_d(x|x_t) \propto k(x_t|x) p_d(x)$ is the target posterior.

Proof. Since $k(x_t|x) = k(x_t - \alpha_t x)$ is translation-invariant, we have

$$\nabla_{x_t} \log k(x_t|x) = -\alpha^{-1} \nabla_x \log k(x_t|x). \tag{36}$$

But by Bayes rule, we have

$$\nabla_x \log k(x_t|x) = \nabla_x \log p_d(x|x_t) - \nabla_x \log p_d(x). \tag{37}$$

Using DSI, it then follows that

$$\nabla_{x_t} \log p_d(x_t) = \int \nabla_{x_t} k(x_t|x) p_d(x|x_t) dx \tag{38}$$

$$= -\alpha^{-1} \int \nabla_x k(x_t|x) p_d(x|x_t) dx \tag{39}$$

$$= \alpha^{-1} \int \left(\nabla_x \log p_d(x) - \nabla_x \log p_d(x|x_t) \right) p_d(x|x_t) dx \tag{40}$$

$$= \alpha^{-1} \int \nabla_x \log p_d(x) p_d(x|x_t) dx, \tag{41}$$

where the last equality follows since

$$\int \nabla_x \log p_d(x|x_t) p_d(x|x_t) dx = \int \nabla_x p_d(x|x_t) dx = \nabla_x \int p_d(x|x_t) dx = \nabla_x 1 = 0. \tag{42}$$

This completes the proof. □

B.3 Derivation of Mixed Score Identity (MSI)

Proposition 4.3 (**Mixed Score Identity**). *Using a Gaussian convolution $k(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$ with a variance-preserving (VP) scheme $\sigma_t^2 = 1 - \alpha_t^2$, and a convex combination of TSI and DSI with coefficients α_t^2 and $1 - \alpha_t^2$, respectively, we have*

$$\nabla_{x_t} \log p_d(x_t) = \int (\alpha_t(x + \nabla_x \log p_d(x)) - x_t) p_d(x|x_t) dx. \tag{43}$$

Proof. For a Gaussian convolution kernel $k(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$, DSI becomes

$$\begin{aligned}
\nabla_{x_t} \log p_d(x_t) &= \int \nabla_{x_t} \log k(x_t|x) p_d(x|x_t) dx && (44)\\
&= \int \nabla_{x_t} \left(-\frac{\|x_t - \alpha x\|^2}{2\sigma_t^2} \right) p_d(x|x_t) dx && (45)\\
&= \int \left(\frac{\alpha x - x_t}{\sigma_t^2} \right) p_d(x|x_t) dx. && (46)
\end{aligned}$$

Since $\sigma_t^2 = 1 - \alpha_t^2$, it then follows that

$$\begin{aligned}
\nabla_{x_t} \log p_d(x_t) &= \int \left(\alpha_t^2 \frac{\nabla_x \log p_d(x)}{\alpha} + (1 - \alpha_t^2) \frac{\alpha_t x - x_t}{\sigma_t^2} \right) p_d(x|x_t) dx && (47)\\
&= \int (\alpha_t(x + \nabla_x \log p_d(x)) - x_t) p_d(x|x_t) dx. && (48)
\end{aligned}$$

This completes the proof. □

C PROOFS REGARDING INVARIANCES AND EQUIVARIANCES

C.1 Proof of Proposition 6.1

We first prove Proposition 6.1 in the main text. This proposition tells us for is a G-equivariant neural sampler which maps from latent space to sample space, if the latent space is invariant then the sample space is also invariant. We restate the formal statement below.

Proposition 6.1. *Let the neural sampler $g_\theta : \mathcal{Z} \to \mathcal{X}$ be an G-equivariant mapping. If the distribution $p(Z)$ over the latent space $\mathcal{Z}$ is G-invariant, then $p_\theta(X) = \int \delta(X - g_\theta(Z)) p(Z) dZ$ is G-invariant.*

Proof. For a transformation G[5], we have

$$\begin{aligned}
p_\theta(G \circ X) &= \int \delta(G \circ X - g_\theta(Z)) p(Z) dZ && (49)\\
&= \int \delta(X - g_\theta(G^{-1} \circ Z)) p(G^{-1} \circ Z) dZ && (50)\\
&= \int \delta(X - g_\theta(Z')) p(Z') dZ && (51)\\
&= \int \delta(X - g_\theta(Z')) p(Z') dZ' && (52)\\
&= p_\theta(X), && (53)
\end{aligned}$$

where $Z' = G^{-1} \circ Z$, and the penultimate line follows since the transformation in $\mathrm{E}(d) \times \mathbb{S}_n$ preserves the volume. □

[5]We slightly abuse the notation by using G to represent both the set $G = \mathrm{E}(3) \times \mathbb{S}_n$ and its elements.

C.2 Monte Carlo Score Estimators are G-equivariant

Recall that, in Section 6, we discussed that we need a G-equivariant MSI estimator for $\int(\alpha_t(X+\nabla \log p_d(X))-X_t)p_d(X|X_t)dx$, and we mentioned that this holds true for a broad class of estimators under mild conditions, including importance sampling and AIS estimators, with different choices of samplers for $p_d(X|X_t)$, such as MALA and HMC. We now provide a detailed discussion below.

Before tackling the equivariance, first recall that we embed the product group in $n \times d$ into an orthogonal group in nd-dimensional space: $\mathrm{E}(d) \times \mathbb{S}_n \hookrightarrow \mathrm{O}(nd)$ by constrain both $\mathcal{X}$ to be the subspace of $\mathbb{R}^{n\times d}$ with zero center of mass. Therefore, both X and X_t are zero-centered, i.e., $X^\top 1 = X_t^\top 1 = 0$. Additionally, the score $\nabla \log p_d(X)$ is also zero-centered. The following Proposition provides a formal statement.

Proposition C.1. *Let $X \in \mathbb{R}^{n\times d}$ be a random variable representing a d-dimensional n-body system. The gradient of a translation invariant energy function f, i.e, $f(X) = f(X + 1t^\top)$ is (1) translation invariant; and (2) 0-centered, i.e., $\nabla f(X)^\top 1 = 0$.*

Proof. (1) We first prove that the gradient is translation invariant. Given the fact that the function is translation invariant:

$$f(X) = f(X + 1t^\top), \tag{54}$$

taking gradient w.r.t. X on both sides, we have

$$\nabla_X f(X) = \nabla_X f(X + 1t^\top). \tag{55}$$

By chain rule, we also have

$$\nabla_X f(X + 1t^\top) = \nabla_{X+1t^\top} f(X + 1t^\top)\nabla_X (X + 1t^\top) = \nabla_{X+1t^\top} f(X + 1t^\top). \tag{56}$$

But let $X' = X + 1t^\top$, we have

$$\nabla_X f(X) = \nabla_{X'} f(X'), \tag{57}$$

which shows that the gradient of a translation invariant function is also translation invariant.

(2) Now we show that the gradient is always 0-centered. Applying first-order Taylor expansion at X_0 to both sides of $f(X) = f(X + 1t^\top)$, we have

$$f(X_0) + \mathrm{tr}\left(\nabla f(X_0)(X - X_0)^\top\right) \tag{58}$$

$$= \underbrace{f(X_0 + 1t^\top)}_{=f(X_0)} + \mathrm{tr}\left(\underbrace{\nabla f(X_0 + 1t^\top)}_{=\nabla f(X_0)}(X - X_0 - 1t^\top)^\top\right). \tag{59}$$

Therefore, we have

$$\mathrm{tr}\left(\nabla f(X_0)(X - X_0)^\top\right) = \mathrm{tr}\left(\nabla f(X_0 + 1t^\top)(X - X_0 - 1t^\top)^\top\right). \tag{60}$$

It then follows that

$$\mathrm{tr}\left(\nabla f(X_0)t1^\top\right) = 0. \tag{61}$$

By the property of the trace operator, we have

$$\mathrm{tr}\left(t^\top \nabla f(X_0)^\top 1\right) = t^\top \nabla f(X_0)^\top 1 = 0. \tag{62}$$

Note that the equation holds for any t. Therefore, we have

$$\nabla f(X_0)^\top 1 = 0. \tag{63}$$

This completes the proof. □

Therefore, the entire MSI: $\nabla_{X_t} \log p_d(X_t) = \int(\alpha_t(X+\nabla_X \log p_d(X))-X_t)p_d(X|X_t)dx$ is **(1) zero-centered** and **(2) G-equivariant w.r.t** X_t. We now need to prove that the aforementioned MC estimators and samplers (IS/AIS with HMC/MALA) meet these requirements.

C.2.1 Importance Sampling

We begin with the simplest case, where we estimate $\int(\alpha_t(X+\nabla \log p_d(X))-X_t)p_d(X|X_t)dx$ with Importance Sampling (IS) with zero-centered Gaussian proposal, as employed in Akhound-Sadegh et al. (2024). Formally, we use the IS estimator (or more precisely, self-normalized IS estimator) as follows:

$$\int(\alpha_t(X+\nabla \log p_d(X))-X_t)p_d(X|X_t)dx \tag{64}$$

$$=\alpha_t\frac{\int(X+\nabla \log p_d(X))p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X,\sigma_t^2)}{\int p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X,\sigma_t^2)dx}dx - X_t \tag{65}$$

$$=\alpha_t\frac{\int(X+\nabla \log p_d(X))p_d(X)\bar{\mathcal{N}}(X|X_t/\alpha_t,\sigma_t^2/\alpha_t^2)}{\int p_d(X)\bar{\mathcal{N}}(X|X_t/\alpha_t,\sigma_t^2/\alpha_t^2)dx}dx - X_t \tag{66}$$

$$\approx\alpha_t\frac{\sum_n(X^{(n)}+\nabla \log p_d(X^{(n)}))p_d(X^{(n)})}{\sum_n p_d(X^{(n)})} - X_t,\quad X^{(1:N)}\sim\bar{\mathcal{N}}(X|X_t/\alpha_t,\sigma_t^2/\alpha_t^2). \tag{67}$$

Here, we slightly abuse the notation of the Gaussian distribution: the random variable and its mean are both in matrix form, while the variance is a scalar. We use this notation to represent an isotropic Gaussian for matrices, i.e., $\mathcal{N}(X|Y,v)=\mathcal{N}(\mathrm{vec}(X)|\mathrm{vec}(Y),vI)$. Unless otherwise specified, we will adhere to this notation throughout our proof. We also use $\bar{\mathcal{N}}$ to denote Gaussian in the zero-centered subspace, i.e.,

$$\bar{\mathcal{N}}(X|\cdot)\propto\begin{cases}\mathcal{N}(X|\cdot), & \text{if } X^\top 1=0,\\ 0, & \text{otherwise.}\end{cases} \tag{68}$$

In other words, we draw samples from the proposal $\bar{\mathcal{N}}(X|X_t/\alpha_t,\sigma_t^2/\alpha_t^2)$, and target at $p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X,\sigma_t^2)$. The importance weight is given by

$$w(X)=\frac{p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X,\sigma_t^2)}{\bar{\mathcal{N}}(X|X_t/\alpha_t,\sigma_t^2/\alpha_t^2)}\propto p_d(X). \tag{69}$$

It is easy to check that this estimator is zero-centered. We now prove that this estimator is G-equivariant. Our proof follows Akhound-Sadegh et al. (2024) closely. We mostly restate their proof here just for completeness.

Proof. Assume we apply some transformation G to X_t, the estimator becomes

$$\alpha_t\frac{\sum_n(G\circ X^{(n)}+\nabla \log p_d(G\circ X^{(n)}))p_d(G\circ X^{(n)})}{\sum_n p_d(G\circ X^{(n)})} - G\circ X_t,\quad G\circ X^{(1:N)}\sim\bar{\mathcal{N}}(G\circ X_t/\alpha_t,\sigma_t^2/\alpha_t^2) \tag{70}$$

$$=\alpha_t\frac{\sum_n(G\circ X^{(n)}+G\circ\nabla \log p_d(X^{(n)}))p_d(G\circ X^{(n)})}{\sum_n p_d(G\circ X^{(n)})} - G\circ X_t \tag{71}$$

$$=G\circ\alpha_t\frac{\sum_n(X^{(n)}+\nabla \log p_d(X^{(n)}))p_d(X^{(n)})}{\sum_n p_d(X^{(n)})} - G\circ X_t \tag{72}$$

$$=G\circ\left(\alpha_t\frac{\sum_n(X^{(n)}+\nabla \log p_d(X^{(n)}))p_d(X^{(n)})}{\sum_n p_d(X^{(n)})} - X_t\right),\quad X^{(1:N)}\sim\bar{\mathcal{N}}(X_t/\alpha_t,\sigma_t^2/\alpha_t^2). \tag{73}$$

The last line follows since $G\circ X^{(1:N)}\sim\bar{\mathcal{N}}(G\circ X_t/\alpha_t,\sigma_t^2/\alpha_t^2)$ is equivalent to $=X^{(1:N)}\sim\bar{\mathcal{N}}(X_t/\alpha_t,\sigma_t^2/\alpha_t^2)$. □

C.2.2 Sampling Importance Resampling

Instead of estimating the integral by IS, we can also perform Sampling Importance Resampling (SIR) using the importance weight. Specifically, we can draw one sample X^* from the Categorical distribution according to the IS weights:

$$\begin{aligned}&X^*=X^{(n^*)},\\ &\text{where } n^*\sim\mathrm{Cat}\left(\frac{p_d(X^{(1)})}{\sum_n p_d(X^{(n)})},\frac{p_d(X^{(2)})}{\sum_n p_d(X^{(n)})},\cdots,\frac{p_d(X^{(N)})}{\sum_n p_d(X^{(n)})}\right),\\ &\text{and } X^{(1:N)}\sim\bar{\mathcal{N}}(X_t/\alpha_t,\sigma_t^2/\alpha_t^2).\end{aligned} \tag{74}$$

The sample obtained by SIR is G-equivariant to X_t.

Proof. If we apply G to X_t, SIR becomes

$$\begin{aligned} X^{*\prime} &= G \circ X^{(n^*)}, \\ \text{where } n^* &\sim \mathrm{Cat}\left(\frac{p_d(G \circ X^{(1)})}{\sum_n p_d(G \circ X^{(n)})}, \frac{p_d(G \circ X^{(2)})}{\sum_n p_d(G \circ X^{(n)})}, \cdots, \frac{p_d(G \circ X^{(N)})}{\sum_n p_d(G \circ X^{(n)})}\right), \\ \text{and } G \circ X^{(1:N)} &\sim \bar{\mathcal{N}}(G \circ X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \end{aligned} \tag{75}$$

Since the target density is G-invariant and $G \circ X^{(1:N)} \sim \bar{\mathcal{N}}(G \circ X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$ is equivalent to $X^{(1:N)} \sim \bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$, we have

$$\begin{aligned} X^{*\prime} &= G \circ X^{(n^*)}, \\ \text{where } n^* &\sim \mathrm{Cat}\left(\frac{p_d(X^{(1)})}{\sum_n p_d(X^{(n)})}, \frac{p_d(X^{(2)})}{\sum_n p_d(X^{(n)})}, \cdots, \frac{p_d(X^{(N)})}{\sum_n p_d(X^{(n)})}\right), \\ \text{and } X^{(1:N)} &\sim \bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \end{aligned} \tag{76}$$

Comparing Equations (74) and (76), we conclude $X^{*\prime} = G \circ X^*$, and hence the sample obtained by SIR is G-equivariant to X_t. □

Additionally, the score at the sample obtained by SIR is also G-equivariant to X_t.

C.2.3 Hamiltonian Monte Carlo

We now look at more complicated cases, where we run Hamiltonian Monte Carlo (HMC) or Langevin Dynamics (LG, including ULA and MALA) to obtain samples from $p_d(X|X_t)$. We start with HMC in this section and look at LG in the next section. Our conclusion will require the following two assumptions:

- Assumption 1: the initial guess in HMC is zero-centered and G-equivariant w.r.t. to X_t. This is a reasonable assumption, as the initial guess can simply be a sample from $\bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$, or can be a sample from Sampling Importance Resampling.
- Assumption 2: the momentum variable in HMC follows zero-centered and G-invariant Gaussian, which also most holds true since the most common choice is standard or isotropic Gaussian.

Under these assumptions, the samples obtained by HMC are zero-centered and G-equivariant to X_t. In the following, we prove that this holds for the first step. The other steps can be simply proved by viewing the sample from the previous guess as the initial guess.

Proof. **(1) Zero-centered.** We first note that the gradient of $\log p_d(X|X_t)$ is zero-centered if both X and X_t are zero-centered. This is easy to check:

$$\nabla_X \log p_d(X|X_t) = \underbrace{\nabla_X \log p_d(X)}_{\text{zero-centered by Proposition C.1}} + \underbrace{\nabla_X \log \bar{\mathcal{N}}(X_t|X)}_{\propto X - X_t}. \tag{77}$$

We now look at the HMC transition kernel (with frog-leap):

$$\begin{aligned} &P \sim \bar{\mathcal{N}}(0, m), \\ &P_{t/2} \leftarrow P + \underbrace{\frac{t}{2}\nabla_X \log p_d(X|X_t)}_{\text{zero-centered by Equation (77)}}, \\ &X' \leftarrow X + tP_{t/2}, \\ &P' \leftarrow P_{t/2} + \underbrace{\frac{t}{2}\nabla_{X'} \log p_d(X'|X_t)}_{\text{zero-centered by Equation (77)}}. \end{aligned} \tag{78}$$

This proposed X' will be accepted according to the Metropolis-Hastings criterion:

$$\alpha = \min\left\{1, \frac{\bar{\mathcal{N}}(P'|0,m)p_d(X'|X_t)}{\bar{\mathcal{N}}(P|0,m)p_d(X|X_t)}\right\} = \min\left\{1, \frac{\bar{\mathcal{N}}(P'|0,m)p_d(X')\bar{\mathcal{N}}(X_t|X')}{\bar{\mathcal{N}}(P|0,m)p_d(X)\bar{\mathcal{N}}(X_t|X)}\right\}. \tag{79}$$

Since P is zero-centered by Assumption 2, all operations will maintain the zero-centered property.

(2) G-equivariant. It is easy to check α is G-invariant. We, therefore, only focus on proving the frog-leap step is G-equivariance. We simplify the frog-leap steps involving X as:

$$X' \leftarrow X + tP + \frac{t}{2}\nabla \log p_d(X|X_t), \text{ where } P \sim \bar{\mathcal{N}}(0,m). \tag{80}$$

Notice that, after applying G to X and hence X_t (by Assumption 1 that the initial guess of X is G-equivariant to X_t), the frog-leap steps involving X become:

$$X'' \leftarrow G\circ X + t\left(G\circ P + \frac{t}{2}\underbrace{\nabla \log p_d(G\circ X|G\circ X_t)}_{=G\circ\nabla \log p_d(X|X_t)}\right), \text{ where } G\circ P \sim \bar{\mathcal{N}}(0,m) \tag{81}$$

$$= G\circ\left(X + tP + \frac{t}{2}\nabla \log p_d(X|X_t)\right), \text{where } P \sim \bar{\mathcal{N}}(G^{-1}\circ 0, M) = \bar{\mathcal{N}}(0,m) \tag{82}$$

$$= G\circ X'. \tag{83}$$

This completes the proof. □

C.2.4 Langevin Dynamics

We now look at LG (i.e., ULA and MALA). By the same argument as HMC, the Metropolis-Hastings acceptance is G-invariant and will not influence our conclusions. We, therefore, simply look at a single step in ULA but note that the same holds for MALA as well. We also take the two assumptions made in HMC with a small modification:

- Assumption 1: the initial guess in LG is zero-centered and G-equivariant w.r.t. to X_t.
- Assumption 2: the Brownian motion we take in LG is izero-centered.

Under these assumptions, the samples obtained by LG are zero-centered and G-equivariant to X_t.

Proof. **(1) Zero-centeredness.** This is trivial by the LG updating formula:

$$X' \leftarrow X + \gamma \underbrace{\nabla_X \log p_d(X|X_t)}_{\text{zero-centered by Equation (77)}} + \sqrt{2\gamma}\mathcal{E}, \tag{84}$$

where $\mathcal{E}$ is a matrix of standard Gaussian noise in the zero-centered subspace, i.e., $\mathcal{E} \sim \bar{\mathcal{N}}(0,1)$.

(2) G-equivariance. Applying G to both X_t and X, and noticing that the Gaussian distribution over $\mathcal{E}$ is G-invariant, we obtain the new LG updating formula:

$$X'' \leftarrow G\circ X + \gamma\underbrace{\nabla \log p_d(G\circ X|G\circ X_t)}_{=G\circ\nabla \log p_d(X|X_t)} + \sqrt{2\gamma}\mathcal{E}, \quad \mathcal{E} \sim \bar{\mathcal{N}}(0,1) \tag{85}$$

$$= G\circ\left(X + \gamma\nabla \log p_d(X|X_t) + \sqrt{2\gamma}\mathcal{E}\right), \quad \mathcal{E} \sim \bar{\mathcal{N}}(G^{-1}\circ 0, 1) = \bar{\mathcal{N}}(0,1) \tag{86}$$

$$= G\circ X'. \tag{87}$$

This completes the proof. □

C.2.5 Annealed Importance Sampling

Another possible choice for the score estimator is Annealed Importance Sampling (AIS, Neal, 2001) or AIS followed by importance resampling. Recall that in IS, we draw samples from the proposal $\bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$, and the target as given by $p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)$. For AIS, we introduce a sequence of intermediate distributions that interpolate between the proposal and the target:

$$\pi_{(k)}(X) \propto \left(\underbrace{\bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)}_{\text{proposal}}\right)^{1-\beta_k} \left(\underbrace{p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)}_{\text{target}}\right)^{\beta_k} \tag{88}$$

$$\propto (p_d(X))^{\beta_k} \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2). \tag{89}$$

where $\beta_0 = 0$ and $\beta_K = 1$. AIS follows an iterative process: The AIS algorithm proceeds iteratively as follows:

- Draw $X_{(0)} \sim \pi_{(0)}$;
- For $k = 1, 2, \ldots, K-1$, run a MCMC using a transition kernel $T(X_{(k)}|X_{(k-1)})$, with $\pi_{(k)}$ as the stationary distribution, to obtain $X_{(k)}$.

In the end, we calculate the IS weight in the joint space defined over $X_{(1:K-1)}$. This will yield the AIS weight:

$$w_{\text{AIS}}(X_{(1:K-1)}) = \frac{\pi_{(1)}(X_{(0)})}{\pi_{(0)}(X_{(0)})}\frac{\pi_{(2)}(X_{(1)})}{\pi_{(1)}(X_{(1)})} \cdots \frac{\pi_{(K)}(X_{(K-1)})}{\pi_{(K-1)}(X_{(K-1)})} \tag{90}$$

$$= \prod_{k=1}^{K} \left(p_d(X_{(k-1)})\right)^{\beta_k - \beta_{k-1}}. \tag{91}$$

Therefore, MSI can be estimated by

$$\int (\alpha_t(X + \nabla \log p_d(X)) - X_t) p_d(X|X_t) dx \tag{92}$$

$$\approx \alpha_t \frac{\sum_n (X^{(n)} + \nabla \log p_d(X^{(n)})) w_{\text{AIS}}(X^{(n)}_{(1:K-1)})}{\sum_n w_{\text{AIS}}(X^{(n)}_{(1:K-1)})} - X_t, \quad X^{(1:N)}_{(1:K-1)} \sim \text{AIS}. \tag{93}$$

We use superscripts to represent the sample index and subscripts to denote the intermediate step index in AIS.

Assume we use HMC or LG with the assumptions discussed in Appendices C.2.3 and C.2.4 as the transition kernel in AIS. The sequence of samples we obtain $(X^{(1:N)}_{(1:K-1)})$ will be G-equivariant w.r.t. X_t. Additionally, notice that $\nabla \log p_d(X^{(n)}))$ is G-invariance to $X^{(n)}$ and w_{AIS} is G-invariance as p_d is G-invariance. We can conclude that the AIS estimator for MSI is G-equivariant w.r.t. X_t.

Following the same argument as in Appendix C.2.2, if we perform AIS followed by a resampling, the result will also be G-equivariant w.r.t. X_t.

D EXPERIMENT DETAILS

D.1 Mixture of 40 Gaussians (MoG-40)

We employ the mixture of 40 Gaussians (MoG-40) target distribution in 2D proposed in Midgley et al. (2023) to visually examine the mass-covering property of different models. We train all methods for 2.5h, which allows all of them to converge.

For our approach, we choose the total number of diffusion steps $T = 30$. We use a variance-preserving (VP) scheme (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-4} to 0.7. We choose the weighting function to be $w(t) = 1/\alpha_t$. For the score network $s_\phi(x_t)$, we use a 5-layer MLP with hidden dimension 400 and SiLU activation. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM

with learning rate 10^{-4} and batch size $1,024$. For the neural sampler $g_\theta(z)$, we use a 5-layer MLP with latent dimension 2, hidden dimension 400 and SiLU activation. We use Adam to train the neural sampler $g_\theta(z)$ using MSI with learning rate 10^{-3}, batch size $1,024$, and gradient norm clip 10.0. Regarding posterior sampling for Equation (16), we use AIS with 10 importance samples, 15 AIS steps. For each AIS step, we use HMC transition kernel with 1 frog-leap step and step size 1.0. We resample one of those 10 AIS samples according to the AIS weights and use that sample as initialization for 5 steps of MALA with step size 10^{-2}. In the end, we estimate the MSI using this single sample (i.e., we perform Monte Carlo estimation with one sample).

For R-KL-based approaches, we align their experiment setups with that for our approach. Specifically, we use a 5-layer MLP with hidden dimension 400 and SiLU activation for the score network $s_\phi(x_t)$. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with learning rate 10^{-4} and batch size $1,024$. For the neural sampler $g_\theta(z)$, we use a 5-layer MLP with latent dimension 2, hidden dimension 400 and SiLU activation. We use Adam to train the neural sampler $g_\theta(z)$ with learning rate 10^{-3}, batch size $1,024$, and gradient norm clip 10.0.

For FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024), we use exactly the same setups as described in the respective papers. Note that both of them use replay buffers. In addition, all methods except iDEM work under the original scale $[-50, 50]$ of the target. iDEM normalizes the target to the range $[-1, 1]$, which may simplify the task.

D.2 Many-Well-32 (MW-32, Internal Coordinate)

We employ the Many-Well target distribution in 32D proposed in Midgley et al. (2023) to examine the mass-covering property of different models, as this target contains 2^{16} modes. We train all models until convergence. Training and sampling time for each model can be found in Table 3 in the main text.

For our approach, we choose the total number of diffusion steps $T = 30$. We use a variance-preserving (VP) scheme (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-4} to 0.15. We choose the weighting function to be $w(t) = 1/\alpha_t$. For the score network $s_\phi(x_t)$, we use a 5-layer MLP with hidden dimension 400 and SiLU activation. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with learning rate 10^{-4} and batch size $1,024$. For the neural sampler $g_\theta(z)$, we use a 5-layer MLP with latent dimension 32, hidden dimension 400 and SiLU activation. We use Adam to train the neural sampler $g_\theta(z)$ using MSI with learning rate 10^{-3}, batch size $1,024$, and gradient norm clip 10.0. Regarding posterior sampling for Equation (16), we use AIS with 10 importance samples, 15 AIS steps. For each AIS step, we use HMC transition kernel with 1 frog-leap step and step size 0.3. We resample one of those 10 AIS samples according to the AIS weights and use that sample as initialization for 5 steps of MALA with step size 5×10^{-2}. In the end, we estimate the MSI using this single sample.

For R-KL-based approaches, we align their experiment setups with that for our approach. Specifically, we use a 5-layer MLP with hidden dimension 400 and SiLU activation for the score network $s_\phi(x_t)$. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with learning rate 10^{-4} and batch size $1,024$. For the neural sampler $g_\theta(z)$, we use a 5-layer MLP with latent dimension 32, hidden dimension 400 and SiLU activation. We use Adam to train the neural sampler $g_\theta(z)$ with learning rate 10^{-3}, batch size $1,024$, and gradient norm clip 10.0.

For FAB, we use exactly the same setup as described in Midgley et al. (2023). For iDEM, we use the same setup as that for experiments in internal coordinates as described in (Akhound-Sadegh et al., 2024) but change the maximum score norm clip threshold to $1,000$, increase the number of MC samples to $1,000$, and reduce σ_{max} in the noise schedule to 1.0. Note that both of FAB and iDEM use replay buffers.

D.3 Double-Well-4 (DW-4, Cartesian Coordinate)

We employ the Double-Well target with 4 particles in 2D. This target was originally introduced by Köhler et al. (2020) and also used in Midgley et al. (2024); Akhound-Sadegh et al. (2024) to evaluate model performance on invariant targets. We train all models until convergence. Training and sampling time for each model can be found in Table 3 in the main text.

For our approach, we choose the total number of diffusion steps $T = 30$. We use a variance-preserving (VP) scheme (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-6} to 0.05. We found that using a

constant weighting function, $w(t) = 1$, is beneficial for handling these complex targets. We use EGNN following Hoogeboom et al. (2022) for both the score network s_ϕ and the neural sampler g_θ. The neural sampler has 8 layers with a hidden dimension of 144 and ReLU activation. The score network has 4 layers with the same width, and it is additionally conditioned on t. We use Adam to train the score network s_ϕ for 100 iterations using DSM with learning rate 10^{-4} and batch size 1024. We use Adam to train the neural sampler $g_\theta(z)$ using MSI with learning rate 5×10^{-4}, batch size 1024, and gradient norm clip 10.0. Regarding posterior sampling for Equation (16), we use AIS with 20 importance samples, 10 AIS steps. For each AIS step, we use 1-step MALA transition kernel with step size 0.01. We resample one of those 20 AIS samples according to the AIS weights and use that sample as initialization for 50 steps of MALA. We dynamically adjust the MALA step size to maintain an acceptance rate between 0.5 and 0.6. Specifically, we increase the step size by a factor of 1.5 when the acceptance rate exceeds 0.6 and decrease it by a factor of 1.5 when the acceptance rate drops below 0.5. In the end, we estimate the MSI using this single sample.

Additionally, we employ early stopping during training. Specifically, we generate 2,000 samples using the neural sampler, which serve as the *predictions*. These samples are then used as the initialization for 50 MALA steps, targeting the target energy. The 2,000 samples obtained after MALA are treated as the *validation* set. We evaluate the energy of both the predictions and the validation set, then calculate the total variation (TV) distances between their energy histograms. We save the model with the lowest TVD. This criterion can be interpreted as asking: how much improvement can be achieved by applying a small number of Langevin dynamics to the model samples? The less improvement we can achieve, the better the model is.

For FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024), we use exactly the same setups as described in the respective papers. Note that both of them use replay buffers.

D.4 Lennard-Johns-13 (LJ-13, Cartesian Coordinate)

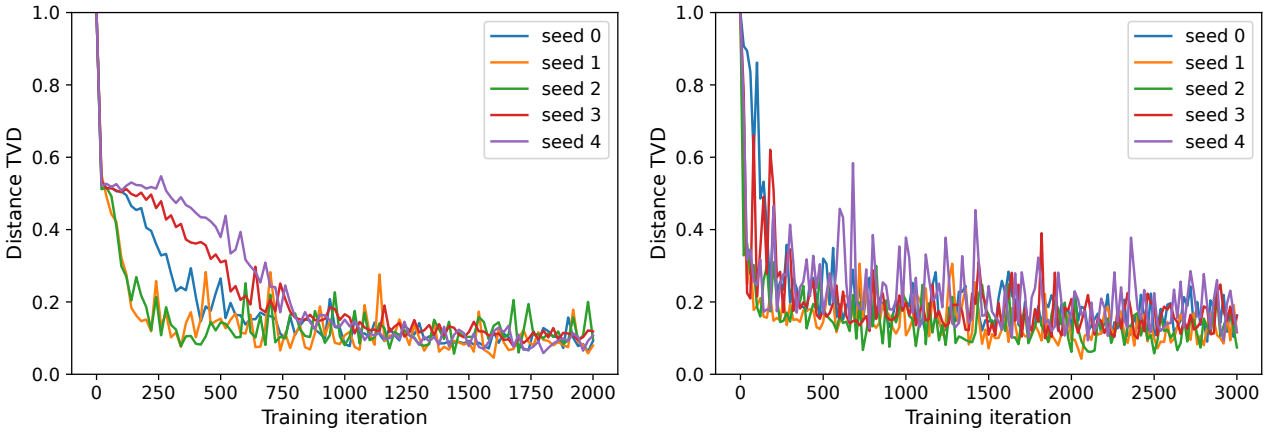
We employ the Lennard-Jones target with 13 particles in 3D, as introduced by Köhler et al. (2020) and later used in Midgley et al. (2024); Akhound-Sadegh et al. (2024) to assess model performance on invariant targets. This target is more complex than DW-4 in the sense that its energy landscape includes prohibitive regions that can destabilize training. However, DW-4 poses its own challenges, as it has two modes, and balancing these modes can be more difficult than handling the Lennard-Jones target. We therefore evaluate our approach and baselines on both to evaluate its behavior comprehensively. We train all models until convergence. Training and sampling time for each model can be found in Table 3 in the main text.

For our approach, we choose the total number of diffusion steps $T = 30$. We use a variance-preserving (VP) scheme (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-6} to 0.05. We also use a constant weighting function, $w(t) = 1$. Both the score network s_ϕ and the neural sampler network g_θ share the same 8-layer architecture with a hidden dimension of 192 and ReLU activation. We use Adam to train the score network s_ϕ for 100 iterations using DSM with learning rate 10^{-4} and batch size 256. We use Adam to train the neural sampler g_θ using MSI with learning rate 5×10^{-4}, batch size 256, and gradient norm clip 10.0. Regarding posterior sampling for Equation (16), we use IS with 500 importance samples. We then resample one of those 20 AIS samples according to the AIS weights and use that sample as initialization for 1,000 steps of MALA. We also dynamically adjust the MALA step size to maintain an acceptance rate between 0.5 and 0.6. Specifically, we increase the step size by a factor of 1.5 when the acceptance rate exceeds 0.6 and decrease it by a factor of 1.5 when the acceptance rate drops below 0.5. Unlike previous tasks, we found that using only the last sample from MALA sometimes leads to suboptimal performance. To improve stability, we track the samples and their gradients from the last 500 steps of MALA, and estimate the MSI using these 500 samples. It's important to note that since the samples and their scores are already computed during MALA, using more samples in the Monte Carlo estimator does not incur any additional computational cost. We found smoothing the LJ target following Moore et al. (2024) can help to stabilize the training. However, our approach works well even without this smoothing. Additionally, we employ early stopping in the same way as in DW-4.

For FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024), we use exactly the same setups as described in the respective papers. Note that both of them use replay buffers.

D.5 Summary and Guidance for Hyperparameter Tuning in DiKL

Below, we summarize some key hyperparameters for our method:



(a) DW-4

(b) LJ-13

Figure 7: Distance TVD along the training process with different seeds.

(1) β_t should not be too large, as this can lead to inaccurate posterior sampling and worse performance. The neural sampling tends to favor the mean of the global mass, which can be seen in Figure 2 in the main text: as the noise level increases, the model is biased towards the mean.

(2) For DW and LJ tasks, it is important to use a large EGNN for the neural sampler. Unlike diffusion models with EGNNs (Akhound-Sadegh et al., 2024; Hoogeboom et al., 2022), our approach involves learning a one-step sampling generator, requiring greater model capacity. We tested smaller networks, such as an EGNN with 6 layers and 128 hidden dimensions. However, these yielded worse performance compared to the larger architecture used in our experiments. On the other hand, the score network does not need to have the same capacity. For example, for DW4, we found a shallower one that suffices for optimal performance. Additionally, interestingly, we found using ReLU yields better performance than SiLU in EGNNs.

(3) The weight function $w(t)$ also requires careful consideration. While other weighting functions commonly used in diffusion models include σ_t^2/α_t or σ_t^2/α_t^2, we found using $1/\alpha_t$ or uniform weighting is more stable in our approach. An empirical guideline for choosing between these is as follows: for more complex targets like DW and LJ, a uniform weighting function can better at encouraging exploitation. On the other hand, for highly multi-modal targets, using $1/\alpha_t$ can accelerate exploration.

(4) The batch size cannot be too small. Empirically, we found that training could be unstable if a small batch size is used. In general, we recommend using a large batch size like 1,024 if it fits into the GPU memory.

On the other hand, perhaps surprisingly, it is not crucial to have a perfect posterior sampler. Our method essentially works in a bootstrapping manner: the posterior samples improve the model, and a better model in turn brings the posterior samples closer to the true target. Having said that, accurate posterior sampling may improve the convergence rate of the model.

E ADDITIONAL EXPERIMENTS AND RESULTS

E.1 Results with Different Random Seeds

Our training process can be subject to randomness. Therefore, in this appendix, we provide results obtained with different random seeds in Figure 7. As we can see, different seeds can have different converge rates and also slightly influence the final performance (for example, the green line in LJ-13 achieves lower TVD than others). However, we observe that (1) a longer training process consistently yields better and more stable results, regardless of the seed; (2) at the end of training, performance may still exhibit slight fluctuations, making the early stopping we employed necessary; and (3) while different seeds introduce minor variations in final performance, these discrepancies are negligible and smaller than the observed fluctuations.

E.2 Additionally Illustrations of Figure 2

As suggested by anonymous reviewers, in Figure 8, we provide an additional visualization of Figure 2, illustrating DiKL alongside KL divergence at different noise levels. This visualization highlights how DiKL balances model-seeking and mode-covering. This visualization also addresses a potential concern: KL divergence at high noise levels can make the loss landscape flat, potentially hindering optimization. We observe that this is indeed the case when the noise level is large (e.g., 5.0). However, by summing over all noise levels in DiKL, this issue is mitigated, resulting in a smoother yet non-flat landscape that is easier to optimize.

However, it is important to acknowledge the limitation of this visualization: we use a Gaussian model to fit a two-mode Gaussian Mixture Model (GMM). As a result, even at optimal performance, the model cannot perfectly fit the target distribution. Therefore, when interpreting this plot, the focus should be on whether the model tends to converge to a single local mode or achieves a global coverage of both modes.

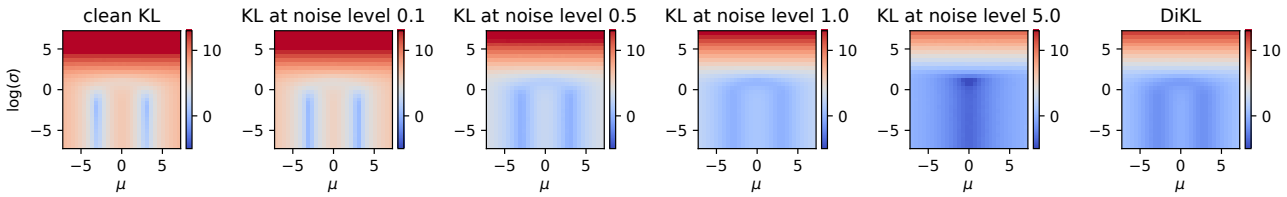


Figure 8: Heatmap of (log scale) KL divergence at different noise levels between a Gaussian model (with mean parameter μ and standard deviation parameter σ) and a two-mode MoG target in 1D. In the last plot, we show the landscape of DiKL, where we sum over the KL divergence on all noise levels.

E.3 Results on LJ-55

As we discussed in the main text, although our one-step generator $g_\theta(z)$ offers significantly faster sampling compared to diffusion-based samplers, it has limited model flexibility relative to multi-step diffusion models such as iDEM. This limitation becomes more pronounced when the target distribution is complex and has a larger Lipschitz constant. To have a comprehensive evaluation of the our approach's weakness, we present the results of DiKL on the LJ-55 potential well in Figure 9. A promising direction for future work is to combine DiKL with multi-step samplers. One possible approach is to introduce several interpolants between the prior and the target distribution and train DiKL sequentially to transport between adjacent interpolants. We leave this for future exploration.

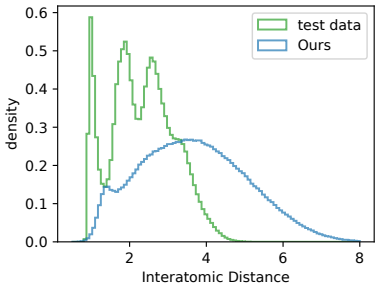


Figure 9: Performance on complex system LJ-55.

F ASSETS AND LICENSES

We use the following codebases for baselines and benchmarks in our experiments:

- FAB (MIT license): PyTorch implementation for MoG-40 and MW-32 (`https://github.com/lollcat/fab-torch`) ; JAX implementation for DW-4 and LJ-13 (`https://github.com/lollcat/se3-augmented-coupling-flows`).

- iDEM (MIT license): PyTorch implementation for all experiments (`https://github.com/jarridrb/DEM`).
- DW-4 and LJ-13 target energy functions (MIT license): bgflow (`https://github.com/noegroup/bgflow`).