BNEM: A BOLTZMANN SAMPLER BASED ON BOOTSTRAPPED NOISED ENERGY MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating independent samples from a Boltzmann distribution is a highly relevant problem in scientific research, *e.g.* in molecular dynamics, where one has initial access to the underlying energy function but not to samples from the Boltzmann distribution. We address this problem by learning the energies of the convolution of the Boltzmann distribution with Gaussian noise. These energies are then used to generate independent samples through a denoising diffusion approach. The resulting method, NOISED ENERGY MATCHING (NEM), has lower variance and only slightly higher cost than previous related works. We also improve NEM through a novel bootstrapping technique called BOOTSTRAP NEM (BNEM) that further reduces variance while only slightly increasing bias. Experiments on a collection of problems demonstrate that NEM can outperform previous methods while being more robust and that BNEM further improves on NEM.

1 INTRODUCTION

A fundamental problem in probabilistic modeling and physical systems simulation is to sample from a target Boltzmann distribution $\mu_{\text{target}} \propto \exp(-\mathcal{E}(x))$ specified by an energy function $\mathcal{E}(x)$. A prominent example is protein folding, which can be formalized as sampling from a Boltzmann distribution (Śledź & Caflisch, 2018) with energies determined by inter-atomic forces (Case et al., 2021). Having access to efficient methods for solving the sampling problem could significantly speed up drug discovery (Zheng et al., 2024) and material design (Komanduri et al., 2000).

However, existing methods for sampling from Boltzmann densities have problems scaling to high dimensions and/or are very time-consuming. As an alternative, Akhound-Sadegh et al. (2024) proposed Iterated Denoising Energy Matching (iDEM), a neural sampler based on denoising diffusion models which is not only computationally tractable but also guarantees good coverage of all modes. iDEM uses a bi-level training scheme that iteratively generates samples from the learned sampler and then does score matching using only the target energy and its gradient. Nevertheless, iDEM requires a large number of samples for its Monte Carlo (MC) score estimate to have low variance and a large number of integration steps even when sampling from simple distributions. Also, its effectiveness highly depends on the choice of noise schedule and score clipping. These disadvantages demand careful hyperparameter tuning and raise issues when working with complicated energies.

To further push the boundary of diffusion-based neural samplers, we propose NOISED ENERGY MATCHING (NEM), which learns a series of noised energy functions instead of the corresponding score functions. Despite a need to differentiate the energy network when simulating the diffusion sampler, NEM targets less noisy objectives as compared with iDEM. Additionally, using an energy-based parametrization enables NEM to use bootstrapping techniques for more efficient training and Metropolis-Hastings corrections for more accurate simulation. By applying the bootstrapping technique, we propose a variant of NEM called BOOTSTRAP NEM (BNEM). BNEM estimates high noise-level energies by bootstrapping from current energy estimates at slightly lower noise levels. BNEM increases bias but reduces variance in its training target.

We conduct experiments on a 2-dimensional 40 Gaussian Mixture Model (GMM), a 4-particle double-welling potential (DW-4), a 13-particle Lennard-Jones potential (LJ-13) and a 55-particle Lennard-Jones potential (LJ-55). We empirically find that our methods lead to state-of-the-art performance on these tasks. Additionally, we found that targeting energies instead of scores is more

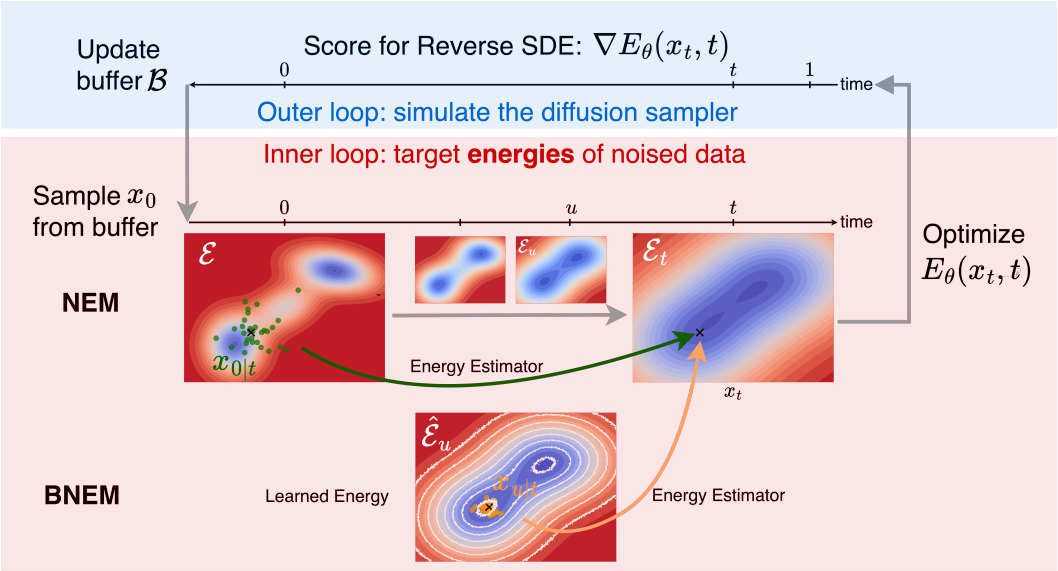


Figure 1: Both EnDEM and BEnDEM parameterize a time-dependent energy network $E_\theta(x_t, t)$ to target the energies of noised data. EnDEM targets an MC energy estimator computed by system energies; BEnDEM targets a Bootstrap energy estimator computed by learned energies at a slightly lower noise level. Contours are the ground truth energies at different noise levels; • represents samples used for computing the MC energy estimator, • represents samples used for computing the Bootstrap energy estimator, and the white contour line represents the learned energy at time u.

robust, requiring fewer Monte Carlo samples during training and fewer integration steps during sampling. This compensates for the need to differentiate through energy networks to obtain scores.

Our contributions are as follows:

- We introduce NEM in section 3.3, including its methodology and theoretical analysis on training target variance and bias, which showcases the advantage of targeting noised energies rather than noised scores.
- We introduce BNEM in section 3.4, where we also theoretically show the Variance-Bias trade-off implied by the bootstrapping energy estimation.
- We present experiment results on four different tasks in section 5, showcasing the advantage of BNEM and NEM compared with DEM. We also conduct ablation studies, which show that NEM is more robust than DEM regarding the number of samples used for training and the number of integration steps used for sampling.

2 Preliminary

We consider learning a generative model for sampling from the Boltzmann distribution

$$\mu_{\text{target}} = \frac{\exp(-\mathcal{E}(x))}{Z}, \quad \text{where} \quad Z = \int \exp(-\mathcal{E}(x))dx, \tag{1}$$

$\mathcal{E}$ is the energy function and Z is the intractable partition function. Generating accurate samples from this type of distribution is highly challenging. The recent success of Diffusion Models provides a promising way to solve this issue.

Diffusion Models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) learn a generative process that starts from a known and tractable base distribution, *a.k.a.* denoising process, which is the inverse of a tractable noising process that starts from the target distribution. Formally, given samples from the target distribution, $x_0 \sim \mu_{\text{target}}$, the noising process is an SDE towards a known

base distribution p_1:

$$dx_t = f(x_t, t)dt + g(t)dw_t, \quad \text{where} \quad t \in [0, 1], \tag{2}$$

$f(x_t, t)$ is called drift coefficient, $g(t)$ is the diffusion coefficient and w_t is standard Brownian Motion. Diffusion Models work by approximately solving the following inverse SDE:

$$dx_t = [f(x_t, t) - g^2(t)\nabla \log p_t(x_t)]dt + g(t)d\tilde{w}_t \tag{3}$$

where $\tilde{w}_t$ is again standard Brownian Motion. In the example of the Variance Exploding (VE) noising process, $f(x_t, t) \equiv 0$ and the perturbation kernel of the noising process is given by $q_{t|0}(x_t|x_0) = \mathcal{N}(x_t; x_0, \sigma_t^2)$, where $\sigma_t^2 := \int g^2(s)ds$. Then the learning objective of DMs is obtained by using Tweedie's formula (Efron, 2011):

$$\mathcal{L}_{DM} = \mathbb{E}_{x_0 \sim p_0, t \sim [0,1], x_t \sim q_{t|0}} \left[\left\| \frac{x_0 - x_t}{\sigma_t^2} - s_\theta(x_t, t) \right\|^2 \right] \tag{4}$$

which allows us to approximate the marginal scores $\nabla \log p_t(x_t)$ with a score network $s_\theta(x_t, t)$ that is parameterized by θ and that targets the conditional scores $\nabla \log p_{t|0}(x_t|x_0) = \nabla \log \mathcal{N}(x_t; x_0, \sigma_t^2 I)$.

3 METHODS

In this section, we first provide an overview before presenting the formalization and training paradigm of simulation-free energy matching. We then discuss the theoretical advantages of our energy matching over score matching. Finally, we describe how bootstrapping is employed to gain improvement.

3.1 OVERVIEW OF NEM FRAMEWORK

This work intends to train a diffusion-based neural sampler that enables diffusion sampling to draw samples from $\mu_{\text{target}}(x) = \frac{\exp(-\mathcal{E}(x))}{Z}$, where we only have access to the energy function $\mathcal{E}$ without any known data from the target distribution.

As in Figure 1, our methods apply the iterative training paradigm, where the inner loop updates the buffer for training the neural sampler, and the outer loop uses the neural sampler to collect new pseudo data to update the buffer. In the inner loop, our model is trained on varied forms of Monte Carlo estimation of the energy.

3.2 DENOISING DIFFUSION-BASED BOLTZMANN SAMPLER

We consider training an energy-based diffusion sampler corresponding to a variance exploding (VE) noising process defined by $dx_t = g(t)dw_t$, where $t \in [0, 1]$, $g(t)$ is a function of time and w_t is Brownian motion. The reverse SDE with Brownian motion $\bar{w}_t$ is $dx_t = -g^2(t)\nabla \log p_t(x_t)d_t + g(t)d\bar{w}_t$, where p_t is the marginal of the diffusion process starting at $p_0 := \mu_{\text{target}}$.

Given the energy $\mathcal{E}(x)$ and the perturbation kernel $q_t(x_t|x_0) = \mathcal{N}(x_t; x_0, \sigma_t^2)$, where $\exp(-\mathcal{E}(x)) \propto p_0(x)$ and $\sigma_t^2 := \int_s^t g^2(s)ds$, one can obtain the marginal noised density p_t as

$$p_t(x_t) \propto \int \exp(-\mathcal{E}(x_0))\mathcal{N}(x_t; x_0, \sigma_t^2 I)dx_0 = \mathbb{E}_{\mathcal{N}(x; x_t, \sigma_t^2 I)}[\exp(-\mathcal{E}(x))]. \tag{5}$$

Going a step further, the RHS of Eq. 5 defines a Boltzmann distribution over the noise-perturbed distribution p_t. The noised energy is defined as the negative logarithm of this unnormalized density

$$\mathcal{E}_t(x) := -\log \mathbb{E}_{\mathcal{N}(x; x_t, \sigma_t^2 I)}[\exp(-\mathcal{E}(x))], \quad \text{where} \quad \exp(-\mathcal{E}_t(x_t)) \propto p_t(x_t). \tag{6}$$

Training on MC estimated targets We can approximate the gradient of $\log p_t$ by fitting a score network $s_\theta(x_t, t)$ to the gradient of Monte Carlo (MC) estimates of Eq. 6, leading to iDEM. The MC score estimator S_K and the training objective can be written as

$$S_K(x_t, t) := \nabla \log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)})), \quad x_{0|t}^{(i)} \sim \mathcal{N}(x; x_t, \sigma_t^2 I), \tag{7}$$

$$\mathcal{L}_{\text{DEM}}(x_t, t) := \|S_K(x_t, t) - s_\theta(x_t, t)\|^2. \tag{8}$$

Alternatively, we can fit an energy network $E_\theta(x_t, t)$ to MC estimates of Eq. 6. The gradient of this energy network w.r.t. input x_t, *i.e.* $\nabla E_\theta(x_t, t)$, can then be used to estimate the score required for diffusion-based sampling. The MC energy estimator E_K and the training objective can be written as

$$E_K(x_t, t) := -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)})), \quad x_{0|t}^{(i)} \sim \mathcal{N}(x; x_t, \sigma_t^2 I), \tag{9}$$

$$\mathcal{L}_{\text{NEM}}(x_t, t) := \|E_K(x_t, t) - E_\theta(x_t, t)\|^2. \tag{10}$$

To enable diffusion-based sampling, one is required to differentiate the energy network to obtain the marginal scores, *i.e.* $\nabla E_\theta(x_t, t)$, which doubles the computation of evaluating $E_\theta(x_t, t)$. Notice that $S_K(x_t, t) = -\nabla E_K(x_t, t)$, regressing the MC energy estimator E_K doesn't need to compute the gradient of target energy $\mathcal{E}$ during training but it is required to compute the gradient of the energy network E_θ during sampling. In other words, it moves the need for differentiation from given energy function $\mathcal{E}$ to neural networks E_θ, which can be beneficial for training on complicated energy functions.

Bi-level Iterative Training Scheme To train the diffusion on the estimated targets, we should obtain noising exact samples from the target. Previous works (Akhound-Sadegh et al., 2024; Midgley et al., 2023) used data points generated by a current learned denoising procedure.

We follow their approach and use a bi-level iterative training scheme for noised energy matching. This involves

- An outer loop that simulates the diffusion sampling process to generate more informative samples. These samples are then used to update a replay buffer $\mathcal{B}$.
- A simulation-free inner loop that matches the noised energies (NEM) or scores (DEM) evaluated at noised versions of the samples stored in the replay buffer.

The significance of this iterated training scheme is proven by Akhound-Sadegh et al. (2024). We, therefore, stick to using it for all relevant samplers' training. The iterated training procedure of NEM is illustrated in Algorithm 1, and its training pipeline is visualized in Figure 1.

3.3 Energy-based learning VS Score-based learning

Both the MC score estimator S_K and the MC energy estimator E_K are biased estimators, where the bias of S_K (Eq. 12) is characterized by Akhound-Sadegh et al. (2024) that it can decrease to 0 when the number of MC samples K increases. We first characterize the bias of E_K in the following Proposition 1, which shows the advantage of NEM in terms of the smaller bias of its regression target.

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

$$\|E_K(x_t, t) - \mathcal{E}_t(x_t)\| \leq \frac{\tilde{c}(x_t)\sqrt{\log(1/\delta)}}{\sqrt{K}} \tag{11}$$

with $c(x_t)/\tilde{c}(x_t) = 2(1 + \|\nabla \mathcal{E}_t(x_t)\|)$*, where*

$$\|S_K(x_t, t) - S_t(x_t)\| \leq \frac{c(x_t)\sqrt{\log(1/\delta)}}{\sqrt{K}}. \tag{12}$$

Proposition 1 shows that the training target of NEM has a smaller error bound (Eq. 11) than the one of DEM (Eq. 12), especially in regions with a steep gradient, *i.e.* large $\|\nabla \mathcal{E}_t(x))\|$. Specifically, the bias of E_K can be charecterized through the above error bound, which is provided in the following corollary:

Corollary 1 *If* $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ *is sub-Gaussian, then the bias of* E_K *can be approximated as*

$$\text{Bias}[E_K(x_t, t)] = \frac{v_{0t}(x_t)}{2m_t^2(x_t)K} \tag{13}$$

where $m_t(x) = \exp(-\mathcal{E}_t(x_t))$ *and* $v_{0t}(x_t) = \text{Var}_{\mathcal{N}(x; x_t, \sigma_t^2 I)}[\exp(-\mathcal{E}_t(x))]$.

Algorithm 1 Iterated training for Noised Energy Matching

Require: Network E_θ, Batch size b, Noise schedule σ_t^2, Base distribution p_1, Num. integration steps L, Replay buffer $\mathcal{B}$, Max Buffer Size $|\mathcal{B}|$, Num. MC samples K

while Outer-Loop **do**
  $\{x_1\}_{i=1}^b \sim p_1(x_1)$
  $\{x_0\}_{i=1}^b \leftarrow \text{sde.int}(\{x_1\}_{i=1}^b, -\nabla E_\theta, L)$ ▷ Simulate the reverse SDE for sampling
  $\mathcal{B} = (\mathcal{B} \cup \{x_0\}_{i=1}^b)$ ▷ Update Buffer $\mathcal{B}$
  while Inner-Loop **do**
    $x_0 \leftarrow \mathcal{B}.\text{sample}()$ ▷ Uniform sampling from $\mathcal{B}$
    $t \sim \mathcal{U}(0,1), x_t \sim \mathcal{N}(x_0, \sigma_t^2)$
    $\mathcal{L}_{\text{NEM}}(x_t, t) = \|E_K(x_t, t) - E_\theta(x_t, t)\|^2$
    $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{NEM}})$
  end while
end while

Ensure: s_θ

Algorithm 2 Inner-loop of Bootstrap Noised Energy Matching training

Require: Network E_θ, Batch size b, Noise schedule σ_t^2, Replay buffer $\mathcal{B}$, Num. MC samples K

while Inner-Loop **do**
  $x_0 \leftarrow \mathcal{B}.\text{sample}()$ ▷ Uniform sampling from $\mathcal{B}$
  $t \sim \mathcal{U}(0,1), x_t \sim \mathcal{N}(x_0, \sigma_t^2)$
  $n \leftarrow \arg\{i : t \in [t_i, t_{i+1}]\}$ ▷ Identify the time split range of t
  $s \sim \mathcal{U}(t_{n-1}, t_n), x_s \sim \mathcal{N}(x_0, \sigma_s^2)$
  $l_s(x_s) \leftarrow \|E_K(x_s, s) - E_\theta(x_s, s)\|^2/\sigma_s^2$
  $l_t(x_t) \leftarrow \|E_K(x_t, t) - E_\theta(x_t, t)\|^2/\sigma_t^2$
  $\alpha \leftarrow \min(1, l_t(x_t)/l_s(x_s))$
  with probability α,
    $\mathcal{L}_{\text{BNEM}}(x_t, t) = \|E_K(x_t, t, s; \text{StopGrad}(\theta)) - E_\theta(x_t, t)\|^2$
  Otherwise, ▷ Use MC estimator if the model is not well trained
    $\mathcal{L}_{\text{BNEM}}(x_t, t) = \|E_K(x_t, t) - E_\theta(x_t, t)\|^2$
  $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{BNEM}})$
end while

Ensure: E_θ

The complete proofs of Proposition 1 and Corollary 1 are given in Appendix A. Additionally, we characterize the variances of S_K and E_K as follows.

Proposition 2 *If $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ is sub-Gaussian and $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$ is bounded, the total variance of the MC score estimator S_K is consistently larger than that of the MC energy estimator E_K in regions associated with low energies, with*

$$\frac{\textbf{tr}\left(Cov[S_K(x_t, t)]\right)}{Var[E_K(x_t, t)]} = 4(1 + \|\nabla \mathcal{E}_t(x_t)\|)^2. \tag{14}$$

In regions associated with high energies, $Var[E_K(x_t, t)] < \textbf{tr}(Cov[S_K(x_t, t)])$ holds when the target energy $\mathcal{E}(x_t)$ is positively related to at least one element of the score $\nabla \mathcal{E}(x_t)$.

This shows that the MC energy estimator can provide a less noisy training signal than the score one, showcasing the theoretical advantage of NEM compared with DEM. One might wonder if the differentiation of the energy network could amplify errors in NEM. The answer is that, even though the energy errors are amplified, these errors are typically very small, and therefore the errors in the differentiated energy are still smaller than the score errors. Our experiments illustrate this by supporting NEM over DEM. The complete proof for the above results is provided in Appendix B.

3.4 Improvement with bootstrapped energy estimation

Using an energy network that directly models the noisy energy landscape has additional advantages. Intuitively, the variances of E_K and S_K explode at high noise levels as a result of the VE noising

process. However, we can reduce variance of the training target in NEM by using the learned noised energies at just slightly lower noise levels rather than using the target energy at time $t = 0$. Based on this, we propose Bootstrap NEM, or **BNEM**, which uses a novel MC energy estimator at high noise levels that is bootstrapped from the learned energies at slightly lower noise levels. Suppose that $E_\theta(\cdot, s)$ is an energy network that already provides an accurate estimate of the energy at a low noise level s, we can then construct a bootstrap energy estimator at a higher noise level $t > s$ by using

$$E_K(x_t, t, s; \theta) := -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-E_\theta(x_{s|t}^{(i)}, s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}\left(x; x_s, (\sigma_t^2 - \sigma_s^2)I\right), \tag{15}$$

The loss used by BNEM is then

$$\mathcal{L}_{\text{BNEM}}(x_t, t|s) := \left\| E_K\left(x_t, t, s; \text{StopGrad}(\theta)\right) - E_\theta(x_t, t) \right\|^2, \tag{16}$$

where the gradient of (Eq. 15) with respect to θ is stopped. We show that this bootstrap energy estimator is trading off variance to bias, as characterized by the following proposition:

Proposition 3 *Given a bootstrap trajectory $\{s_i\}_{i=0}^n$ such that $\sigma_{s_i}^2 - \sigma_{s_{i-1}}^2 \leq \kappa$, where $s_0 = 0$, $s_n = 1$ and $\kappa > 0$ is a small constant. Suppose E_θ is incrementally optimized from $t \in [0, 1]$ as follows: if $t \in [s_i, s_{i+1}]$, $E_\theta(x_t, t)$ targets an energy estimator bootstrapped from $\forall s \in [s_{i-1}, s_i]$ using Eq. 15. For $\forall 0 \leq i \leq n$ and $\forall s \in [s_{i-1}, s_i]$, the variance of the bootstrap energy estimator is given by*

$$\textit{Var}[E_K(x_t, t, s; \theta)] = \frac{v_{st}(x_t)}{v_{0t}(x_t)} \textit{Var}[E_K(x_t, t)] \tag{17}$$

and the bias of $E_K(x_t, t, s; \theta)$ is given by

$$\text{Bias}[E_K(x_t, t, s; \theta)] = \frac{v_{0t}(x_t)}{2m_t^2(x_t)K^{i+1}} + \sum_{j=1}^{i} \frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j}. \tag{18}$$

where $m_z(x_z) = \exp(-\mathcal{E}_z(x_z))$ and $v_{yz}(x_z) = \text{Var}_{\mathcal{N}(x; x_z, (\sigma_z^2 - \sigma_y^2)I)}[\exp(-\mathcal{E}_y(x))]$ for $\forall 0 \leq y < z \leq 1$.

A detailed discussion and proof are given in Appendix C. Proposition 3 demonstrates that the bootstrap energy estimator, which estimates the noised energies by sampling from a x_t-mean Gaussian with smaller variance, can reduce the variance of the training target while this new target can introduce accumulated bias. In general, the high variance of training target introduces high bias of the learned value; Bootstrapping reduces the variance of training target, therefore reduces bias of the learned value; However, bootstrapping is based on the learned value at a slightly lower noise level, which can be biased as well, introducing extra bias that can be accumulated; Therefore, bootstrapping trades the extra accumulated bias to the high bias introduced by the high variance of training target. Proposition 3 shows that, within proper choice of number of MC samples K and bootstrap trajectory, the bias of BNEM (Eq. 13) can be smaller than that of NEM (Eq. 18). This provides theoretical guarantee for improving performance via bootstrapping. In the following paragraph, we define how we select s and t using a novel variance-controlled bootstrap schedule and training scheme.

Variance-Controlled Bootstrap Schedule BNEM aims to trade the bias of the learning target to its variance. To ensure that the variance of the Bootstrap energy estimator at t bootstrapped from s is controlled by a predefined bound β, *i.e.* $\sigma_t^2 - \sigma_s^2 \leq \beta$, we first split the time range $[0, 1]$ with $0 = t_0 < t_1 < ... < t_N = 1$ such that $\sigma_{t_{i+1}}^2 - \sigma_{t_i}^2 \leq \beta/2$; then we uniformly sample s and t from adjacent time splits during training for variance control.

Training of BNEM To train BNEM, it is crucial to account for the fact that bootstrap energy estimation at t can only be accurate when the noised energy at s is well-learned. To favor this, we first use NEM to obtain an initial energy network and then apply training with the bootstrapped estimator. We then aim to use bootstrapping only when the energy network is significantly better at time s than at time t. We quantify this by evaluating the NEM losses at times t and s given a clean data point x_0. In particular, we could compare $\tilde{l}_s(x_s) = \mathcal{L}_{NEM}(x_s, s)$ and $\tilde{l}_t(x_t) = \mathcal{L}_{NEM}(x_t, t)$, where $x_s \sim \mathcal{N}(x_0, \sigma_s^2 I)$ and $x_t \sim \mathcal{N}(x_0, \sigma_t^2 I)$. However, the variance of training targets increases with

Table 1: Neural sampler performance comparison for 4 different energy functions. We measured the performance using data Wasserstein-2 distance (x-$\mathcal{W}_2$), Energy Wasserstein-2 distance ($\mathcal{E}$-$\mathcal{W}_2$), and Total Variation (TV). * indicates divergent training. Each sampler is evaluated with 3 random seeds and we report the mean $\pm$ standard deviation for each metric. † indicates a large standard deviation of metric ($> 20\%$) and we report the best value.

Energy →	**GMM-40** ($d = 2$)			**DW-4** ($d = 8$)			**LJ-13** ($d = 39$)			**LJ-55** ($d = 165$)		
Sampler ↓	**x-$\mathcal{W}_2$**	**$\mathcal{E}$-$\mathcal{W}_2$**	**TV**	**x-$\mathcal{W}_2$**	**$\mathcal{E}$-$\mathcal{W}_2$**	**TV**	**x-$\mathcal{W}_2$**	**$\mathcal{E}$-$\mathcal{W}_2$**	**TV**	**x-$\mathcal{W}_2$**	**$\mathcal{E}$-$\mathcal{W}_2$**	**TV**
DDS	$15.04_{\pm 2.97}$	$305.13_{\pm 186.06}$	$0.96_{\pm 0.01}$	$0.82_{\pm 0.21}$	$558.79_{\pm 787.92}$	$0.38_{\pm 0.14}$	*	*			*	
PIS	$6.58_{\pm 1.68}$	$79.86_{\pm 7.79}$	$0.95_{\pm 0.01}$	*	*	*	*		*	*	*	*
FAB	$9.08_{\pm 1.41}$	$47.60_{\pm 7.24}$	$0.79_{\pm 0.07}$	$0.62_{\pm 0.02}$	$112.70_{\pm 20.33}$	$0.38_{\pm 0.02}$	*	*	*	*	*	*
iDEM	$8.21_{\pm 5.43}$	$60.49_{\pm 70.12}$	$0.82_{\pm 0.03}$	$0.50_{\pm 0.03}$	$2.80_{\pm 1.72}$	$0.16_{\pm 0.01}$	$0.87_{\pm 0.00}$	6770†	$0.06_{\pm 0.01}$	$2.06_{\pm 0.04}$	17651†	$0.16_{\pm 0.02}$
NEM (ours)	$5.28_{\pm 0.89}$	$44.56_{\pm 39.56}$	$0.91_{\pm 0.02}$	$\mathbf{0.48}_{\pm 0.02}$	$0.85_{\pm 0.52}$	$\mathbf{0.14}_{\pm 0.01}$	$0.87_{\pm 0.01}$	$5.01_{\pm 3.14}$	$\mathbf{0.03}_{\pm 0.00}$	$1.90_{\pm 0.01}$	$\mathbf{118.58}_{\pm 106.63}$	$0.10_{\pm 0.02}$
BNEM (ours)	$\mathbf{3.66}_{\pm 0.30}$	$\mathbf{1.87}_{\pm 1.00}$	$\mathbf{0.79}_{\pm 0.04}$	$0.49_{\pm 0.02}$	$\mathbf{0.38}_{\pm 0.09}$	$\mathbf{0.14}_{\pm 0.01}$	$\mathbf{0.86}_{\pm 0.00}$	$\mathbf{1.02}_{\pm 0.69}$	$\mathbf{0.03}_{\pm 0.00}$	$\mathbf{1.88}_{\pm 0.01}$	$601.61_{\pm 784.19}$	$\mathbf{0.08}_{\pm 0.01}$

time, meaning that a direct comparison of these losses is not reliable. To avoid this, we normalize these losses according to the noise schedule variance and compare instead $l_s(x_s) = \mathcal{L}_{NEM}(x_s, s)/\sigma_s^2$ and $l_t(x_t) = \mathcal{L}_{NEM}(x_t, t)/\sigma_t^2$. We then adopt a rejection training scheme according to the ratio of these normalized NEM losses:

(a) given s, t and x_0, we first noise x_0 to s and t respectively, and compute the normalized losses defined above, *i.e.* $l_s(x_s)$ and $l_t(x_t)$;

(b) these losses indicate how well the energy network fits the noised energies at different times; we then compute $\alpha = \min(1, l_s(x_s)/l_t(x_t))$;

(c) with probability α, we accept targeting an energy estimator at t bootstrapped from s and otherwise, we stick to targeting the original MC energy estimator.

We provide a full description of the inner-loop of BNEM training in Algorithm 2.

4 Related Works

Boltzmann Generator. To learn a neural sampler for Boltzmann distribution, unlike data-driven tasks where a sufficient amount of data is available, simply minimizing the reverse Kullback-Leibler (KL) divergence, *i.e.*$\min_\theta \mathbb{D}_{KL}(\mu_{target} \| q_\theta)$, can lead to mode-seeking behavior. Boltzmann Generator(Noé et al., 2019) addresses this problem by minimizing the combination of forward and reverse KL divergence.

PIS, DDS, and FAB. Inspired by the rapid development of deep generative models, *e.g.* diffusion models (Song & Ermon, 2019; Ho et al., 2020), pseudo samples could be generated from an arbitrary prior distribution. Then, we can train the neural samplers by matching these sample trajectories, as in Path Integral Sampler (PIS)(Zhang & Chen, 2022) and Denoising Diffusion Sampler (DDS) (Vargas et al., 2023). Midgley et al. (2023) further deploy a replay buffer for the trajectories while proposing an α-divergence as the objective to avoid mode-seeking. However, these methods require simulation during training, which still poses challenges for scaling up to higher dimensional tasks.

iDEM. To further boost scalability and mode-coverage, iDEM (Akhound-Sadegh et al., 2024) is proposed to target an MC score estimator that estimates scores of noised data, enabling the usage of the efficient diffusion sampling. iDEM is trained with a bi-level scheme: (1) a simulation-free inner-loop that targets time-involved scores of buffer data; (2) an outer-loop that simulates the learned diffusion sampler to generate more informative data in the buffer. It achieves previous state-of-the-art performance on the tasks above.

iEFM. To regress x_t directly, one can leverage Flow Matching Lipman et al. (2023). Woo & Ahn (2024) proposes iEFM, a variant of iDEM that targets the MC estimated vector fields in a Flow Matching fashion, which is empirically found to outperform iDEM in GMM and DW-4. In fact, we found that iEFM and iDEM can be linked through Tweedie's formula (Efron, 2011) shown in Appendix G with supplementary experiments provided in Appendix I.8.

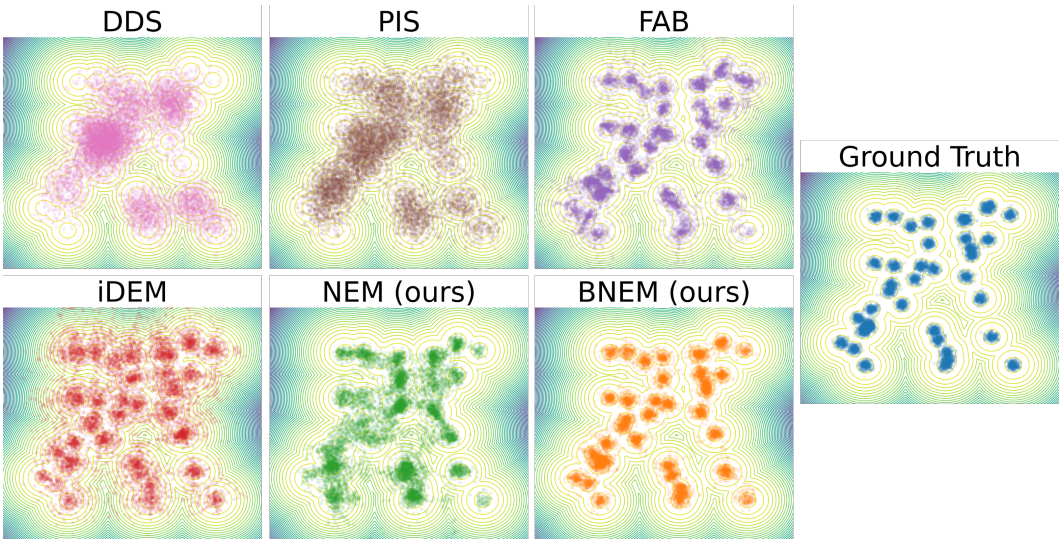
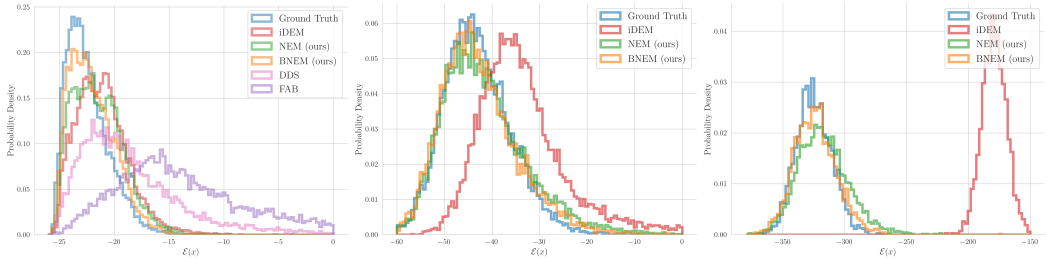


Figure 2: Sampled points from samplers applied to GMM-40 potentials, with the ground truth represented by contour lines.



(a) DW-4 (b) LJ-13 (c) LJ-55

Figure 3: Histogram for energies of samples generated by each sampler.

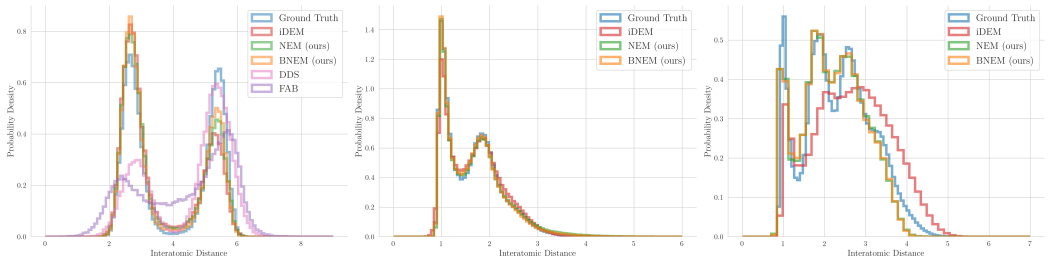
5 Experiments

We evaluate our methods and baseline models on 4 potentials. A complete description of all energy functions, metrics, and experiment setups is in Appendix H. Supplementary experiments can be found in Appendix I. We provide an anonymous link to our code for implementation reference[1].

Datasets. We evaluate all neural samplers on 4 different datasets: a GMM with 40 modes ($d = 2$), a 4-particle double-well (DW-4) potential ($d = 8$), a 13-particle Lennard-Jones (LJ-13) potential ($d = 39$) and a 55-particle Lennard-Jones (LJ-55) potential ($d = 165$). For LJ-n potentials, the energy can be extreme when particles are too close to each other, creating problems for estimating noised energies. To overcome this issue, we smooth the Lennard-Jones potential through the cubic spline interpolation, according to Moore et al. (2024).

Baseline. We compare NEM and BNEM to following recent works: Denoising Diffusion Sampler (DDS)(Vargas et al., 2023), Path Integral Sampler (PIS)(Zhang & Chen, 2022), Flow Annealed Bootstrap (FAB)(Midgley et al., 2023) and Iterated Denoising Energy Matching (iDEM)(Akhound-Sadegh et al., 2024). Due to the high complexity of DDS and PIS training due to their simulation-based nature, we limit their integration step when sampling to 100. Also, as iDEM (Akhound-Sadegh et al., 2024) shows excellent performance of GMM and DW-4, we consider showing its robustness together with NEM and BNEM by limiting both the integration step and number of MC samples to 100. For complex tasks, *i.e.* LJ-13 and LJ-55, we stick to using 1000 steps for reverse SDE integration and 1000 MC samples in the estimators. We train all samplers using an NVIDIA-A100 GPU.

[1] https://anonymous.4open.science/r/nem-D664/README.md



(a) DW-4 (b) LJ-13 (c) LJ-55

Figure 4: Histogram for interatomic distance of samples generated by each sampler.

Architecture. We use the same network architecture (MLP for GMM and EGNN for particle systems, *i.e.* DW-4, LJ-13, and LJ-55) in DDS, PIS, iDEM, NEM and BNEM. To ensure a similar number of parameters for each sampler, if the score network is parameterized by $s_\theta(x,t) = f_\theta(x,t)$, the energy network is set to be $E_\theta(x,t) = \mathbf{1}^\top f_\theta(x,t) + c$ with a learnable scalar c. Furthermore, this setting ensures SE(3) invariance for the energy network. However, FAB requires an invertible architecture. For a fair comparison, we replace the neural network architecture in FAB with a continuous flow matching, to ensure a similar number of parameters for each sampler.

Metrics. We use data 2-Wasserstein distance (x-$\mathcal{W}_2$), energy 2-Wasserstein distance ($\mathcal{E}-\mathcal{W}_2$), and Total Variation (TV) as metrics. TV is computed from data in GMM; For equivariant systems, *i.e.* DW-4, LJ-13, and LJ-55, TV is based on the interatomic distance. To compute $\mathcal{W}_2$ and TV, we use pre-generated samples as datasets: (a) For GMM, we sample from the ground truth distribution; (b) For DW-4, LJ-13 and LJ-55, we use samples from Klein et al. (2023b).

5.1 Main Results

We report x-$\mathcal{W}_2$, $\mathcal{E}-\mathcal{W}_2$, and TV for all tasks in Table 1. The table demonstrates that by targeting less noisy objectives, NEM outperforms DEM on most metrics, particularly for complex tasks such as LJ-13 and LJ-55. Figure 2 visualizes the generated samples from each sampler in the GMM benchmark. When the compute budget is constrained—by reducing neural network size for DDS, PIS, and FAB, and limiting the number of integration steps and MC samples to 100 for all baselines—none of them achieve high-quality samples with sufficient mode coverage. In particular, iDEM produces samples that are not concentrated around the modes in this setting. Conversely, NEM generates samples with far fewer outliers, focusing more on the modes and achieving the best performance on all metrics. BNEM can further improve on top of NEM, generating data that are most similar to the ground truth ones.

For the equivariant tasks, *i.e.* DW-4, LJ-13, and LJ-55, we compute the energies and interatomic distances for each generated sample. Empirically, we found that smoothing the energy-distance function by a cubic spline interpolation (Moore et al., 2024) to avoid extreme values (i.e. when the particles are too close) is a key step when working with the Lennard Jones potential. Furthermore, this smoothing technique can be applied in a wide range of many-particle systems (Pappu et al., 1998). Therefore, NEM and BNEM can be applied without significant modeling challenges. We also provide an ablation study on applying this energy-smoothing technique to score-based iDEM. The results in Table 4 suggest that it could help to improve the performance of iDEM but NEM still outperforms it.

Figures 3 and 4, show the histograms of the energies and interatomic distances, respectively. It shows that both NEM and BNEM closely match the ground truth densities, outperforming all other baselines. Moreover, for the most complex task, LJ-55, NEM demonstrates greater stability during training, generating more low-energy samples, unlike iDEM, which is more susceptible to instability and variance from different random seeds.

For $\mathcal{E}$-$\mathcal{W}_2$, it is susceptible to outliers with high energy, especially in complex tasks like LJ-13 and LJ-55, where an outlier that corresponds to a pair of particles that are close to each other can result in an extremely large value of this metric. We find that for LJ-n tasks, NEM and BNEM tend to generate

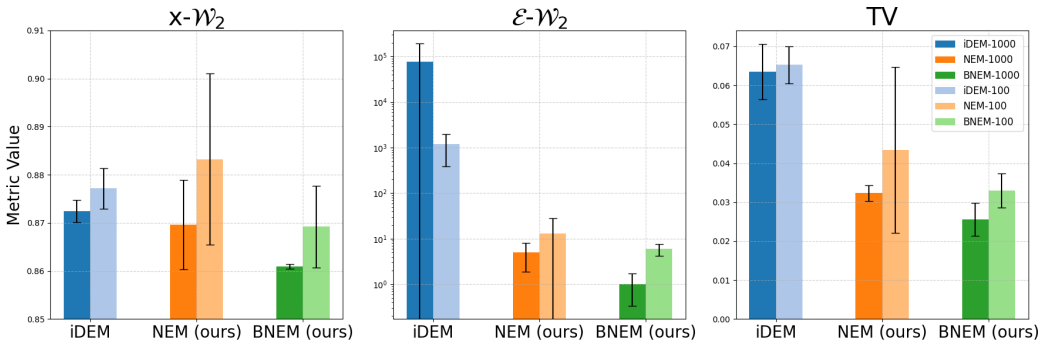


Figure 5: Barplots comparing DEM, NEM, and BNEM evaluations with 1000 vs. 100 integration steps and MC samples on the LJ-13 benchmark.

samples with low energies and result in low $\mathcal{E}$-$\mathcal{W}_2$, while iDEM can produce high energy outliers and therefore corresponds to the extremely high values of this metric.

We notice that for LJ-55, NEM is more stable than BNEM, with no divergent training. However, a well-converged BNEM can be better than NEM as we expected, with generated samples that are highly similar to the ground truth data, achieving state-of-the-art performance in this task.

5.2 Robustness of DEM, NEM, and BNEM

Table 1 partially shows the better robustness of NEM and BNEM in terms of fewer integration steps and MC samples, compared with iDEM and other baselines. We further explore the robustness in a more complex benchmark, LJ-13, to demonstrate the advantage of our energy-based models, *i.e.* NEM and BNEM. Figure 5 visualizes the difference of metrics when reducing the integration steps and number of MC samples from 1000 to 100. A complete comparison between iDEM, NEM, and BNEM under different computational budgets is provided in Table 5 (Appendix I). It shows that when limiting the computing budgets, iDEM can degrade significantly in GMM and DW-4 potentials, while NEM is less affected. In LJ-13 potential, both iDEM and NEM can degrade, while NEM-100 is still better than DEM-100 and even better than DEM-1000 in $\mathcal{E}$-$\mathcal{W}_2$ and TV. Furthermore, BNEM can achieve better performance. BNEM-100 is less affected and matches the performance of iDEM-1000 in GMM and DW-4, and even outperforms iDEM-1000 in the more complex LJ-13 potential, showcasing its capability.

6 Conclusion

In this work, we propose NEM and BNEM, neural samplers for Boltzmann distribution and equilibrium systems like many-body systems. NEM uses a novel Monte Carlo energy estimator with reduced bias and variance. BNEM builds on NEM, employing an energy estimator bootstrapped from lower noise-level data, theoretically trading bias for variance. Empirically, BNEM achieves state-of-the-art results on 4 different benchmarks, GMM, DW-4, LJ-13, and LJ-55.

Limitations and future work. Even though NEM can outperform DEM with fewer integration steps, the requirement of differentiating the neural network w.r.t. the input poses a memory issue for high-dimensional tasks, and therefore would require further improvement for scalability in terms of memory in the future; Secondly, though BNEM demonstrates its potential to achieve improvement on top of NEM, its training process is yet not stable enough. Therefore, improving its training stability is of interest in the future; Besides, the cubic-spline interpolation technique is applied for the LJ-n potentials, and therefore a more general learning-based technique that can fix the extreme energy issue, such as contrastive learning, is more desired; Furthermore, the well-learned noised energies allow us more possible ways to improve generation quality beyond normal denoising diffusion approach, such as integrating Metropolis-Hastings correction inside the denoising process to generate more accurate samples.

REFERENCES

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Alexander Tong. Iterated denoising energy matching for sampling from boltzmann densities, 2024. URL .

David A Case, H Metin Aktulga, Kellon Belfon, Ido Ben-Shalom, Scott R Brozell, David S Cerutti, Thomas E Cheatham III, Vinícius Wilian D Cruzeiro, Tom A Darden, Robert E Duke, et al. *Amber 2021*. University of California, San Francisco, 2021.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

RÃ©mi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, AurÃ©lie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, LÃ©o Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78): 1–8, 2021. URL .

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Leon Klein, Andreas Krämer, and Frank Noe. Equivariant flow matching. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 59886–59910. Curran Associates, Inc., 2023a. URL .

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching, 2023b. URL .

Jonas Köhler, Leon Klein, and Frank Noe. Equivariant flows: Exact likelihood generative learning for symmetric densities. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5361–5370. PMLR, 13–18 Jul 2020. URL .

R Komanduri, N Chandrasekaran, and LM Raff. Md simulation of nanometric cutting of single crystal aluminum–effect of crystal orientation and direction of cutting. *Wear*, 242(1-2):60–88, 2000.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL .

Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap, 2023. URL .

J. Harry Moore, Daniel J. Cole, and Gabor Csanyi. Computing hydration free energies of small molecules with first principles accuracy, 2024. URL .

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

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators – sampling equilibrium states of many-body systems with deep learning, 2019. URL .

Art B. Owen. *Monte Carlo theory, methods and examples*. , 2013.

Art B. Owen. *Practical Quasi-Monte Carlo Integration*. , 2023.

Rohit V Pappu, Reece K Hart, and Jay W Ponder. Analysis and application of potential energy smoothing and search methods for global optimization. *The Journal of Physical Chemistry B*, 102 (48):9725–9742, 1998.

Paweł Śledź and Amedeo Caflisch. Protein structure-based drug design: from docking to molecular dynamics. *Current opinion in structural biology*, 48:93–102, 2018.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL .

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers, 2023. URL .

Dongyeop Woo and Sungsoo Ahn. Iterated energy-based flow matching for sampling from boltzmann densities, 2024. URL .

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling, 2022. URL .

Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, et al. Predicting equilibrium distributions for molecular systems with deep learning. *Nature Machine Intelligence*, pp. 1–10, 2024.

APPENDIX

CONTENTS

1 Introduction **1**

2 Preliminary **2**

3 Methods **3**

3.1 Overview of NEM framework 3

3.2 denoising diffusion-based Boltzmann sampler 3

3.3 Energy-based learning VS Score-based learning 4

3.4 Improvement with bootstrapped energy estimation 5

4 Related works **7**

5 Experiments **8**

5.1 Main Results 9

5.2 Robustness of DEM, NEM, and BNEM 10

6 Conclusion **10**

A Proof of Proposition 1 **13**

B Proof of Proposition 2 **14**

C Proof of Proposition 3 **16**
C.1 Bootstrap(1) estimator 16
C.2 Bootstrap(n) estimator 17

D Incorporating Symmetry Using NEM **18**

E Generalizing NEM **19**
E.1 NEM for General SDEs 19
E.2 MC Energy Estimator as an Importance-Weighted Estimator 20

F Memory-Efficient NEM **20**

G TweeDEM: a middle point between Denoising Energy Matching and Energy Flow Matching through Tweedie's formula **21**

H Experimental Details **22**
H.1 Energy functions 22
H.2 Evaluation Metrics 23
H.3 Experiment Settings 23

I Supplementary Experients **24**
I.1 Main Results 24
I.2 Comparing the Robustness of Energy-Matching and Score-Matching 24
I.3 Empirical Analysis of the Variance of E_K and S_K 26
I.4 Empirical Analysis of the Bias of Bootstrapping 27
I.5 Complexity Analysis 29
I.6 Performance Gain Under the Same Computational Budget 30
I.7 Experiments for Memory-Efficient NEM 30
I.8 Experiments for TweeDEM 30

A Proof of Proposition 1

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

$$\|E_K(x_t,t) - \mathcal{E}_t(x_t)\| \leq \frac{\tilde{c}(x_t)\sqrt{\log(1/\delta)}}{\sqrt{K}} \tag{19}$$

with $c(x_t)/\tilde{c}(x_t) = 2(1+\|\nabla\mathcal{E}_t(x_t)\|)$, where

$$\|S_K(x_t,t) - S_t(x_t)\| \leq \frac{c(x_t)\sqrt{\log(1/\delta)}}{\sqrt{K}} \tag{20}$$

Proof. We first introduce the error bound of the MC score estimator S_K, where $S_K = \nabla E_K$, proposed by Akhound-Sadegh et al. (2024) as follows

$$\|S_K(x_t,t) - S(x_t,t)\| \leq \frac{2C\sqrt{\log(\frac{2}{\delta})}(1+\|\nabla\mathcal{E}_t(x_t)\|)\exp(\mathcal{E}_t(x_t))}{\sqrt{K}} \tag{21}$$

which assumes that $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ is sub-Gaussian. Let's define the following variables

$$m_t(x_t) = \exp(-\mathcal{E}_t(x_t)) \tag{22}$$

$$v_{st}(x_t) = \text{Var}_{\mathcal{N}(x_t,(\sigma_t^2-\sigma_s^2)I)}[\exp(-\mathcal{E}(x))] \tag{23}$$

By the sub-Gaussianess assumption, it's easy to show that the constant term C in Equation 21 is $C = \sqrt{2v_{0t}(x_t)}$. Notice that E_K is a logarithm of an unbiased estimator . By the sub-Gaussian assumption, one can derive that E_K is also sub-Gaussian. Furthermore, it's mean and variance can be derived by employing a first-order Taylor expansion:

$$\mathbb{E}E_K(x_t,t) \approx \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_t^2(x_t)K} \tag{24}$$

$$\text{Var}[E_K(x_t,t)] \approx \frac{v_{0t}(x_t)}{m_t^2(x_t)K} \tag{25}$$

And one can obtain its concentration inequality by incorporating the sub-Gaussianess

$$\|E_K(x_t,t) - \mathbb{E}[E_K(x_t,t)]\| \leq \sqrt{2\frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K}\log\frac{2}{\delta}} \tag{26}$$

By using the above Inequality 26 and the triangle inequality

$$\|E_K(x_t,t) - \mathcal{E}_t(x_t)\| \leq \|\log E_K(x_t,t) - \mathbb{E}[E_K(x_t,t)]\| + \|\mathbb{E}[E_K(x_t,t)] - \mathcal{E}_t(x_t)\| \tag{27}$$

$$= \|\log E_K(x_t,t) - \mathbb{E}[E_K(x_t,t)]\| + \frac{v_{0t}(x_t)}{2m_t^2(x_t)K} \tag{28}$$

$$\leq \sqrt{2\frac{v_{0t}(x_t)}{m_t^2(x_t)K}\log\frac{2}{\delta}} + \frac{v_{0t}(x_t)}{2m_t^2(x_t)K} \tag{29}$$

$$= \frac{C\sqrt{\log\frac{2}{\delta}}\exp(\mathcal{E}_t(x_t))}{\sqrt{K}} + O(1/K) \tag{30}$$

$$= \frac{c(x_t)}{2(1+\|\nabla\mathcal{E}_t(x_t)\|)}\frac{\sqrt{\log\frac{1}{\delta}}}{\sqrt{K}} \tag{31}$$

Therefore, we have $c(x_t) = 2(1+\|\nabla\mathcal{E}_t(x_t)\|)\tilde{c}(x_t)$. It demonstrates a less biased estimator, which, what's more, doesn't require a sub-Gaussianess assumption over $\|\nabla\mathcal{E}(x_{0|t}^{(i)})\|$. □

B Proof of Proposition 2

Proposition 2 *If $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ is sub-Gaussian and $\|\nabla\exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$ is bounded, the total variance of the MC score estimator S_K is consistently larger than that of the MC energy estimator E_K in low-energy regions, with*

$$\frac{\textbf{tr}\,(Cov[S_K(x_t,t)])}{Var[E_K(x_t,t)]} = 4(1+\|\nabla\mathcal{E}_t(x_t)\|)^2 \tag{32}$$

In high-energy regions, $C[E_K(x_t,t)] < \textbf{tr}(Cov[S_K(x_t,t)])$ holds when the system energy $\mathcal{E}(x_t)$ is positively related to at least one element of the score $\nabla\mathcal{E}(x_t)$.

Proof. We split the proof into two parts: low-energy region and high-energy one. The proof in the low-energy region requires only the aforementioned sub-Gaussianess and bounded assumptions, while the one in the high-energy region requires an additional constraint which will be clarified later. Review that S_K can be expressed as an importance-weighted estimator as follows:

$$S_K(x_t,t) = \frac{\frac{1}{K}\sum_{i=1}^K \nabla\exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\frac{1}{K}\sum_{i=1}^K \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \tag{33}$$

Let $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\| \leq M$, where $M > 0$. Since a bounded variable is sub-Gaussian, this assumption resembles a sub-Gaussianess assumption of $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$. Then each element of $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$, i.e. $\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))[j]$, is bounded by M. And therefore $\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))[j]$ are sub-Gaussian.

In low-energy regions. $\exp(-\mathcal{E}(x))$ is concentrated away from 0 as $\mathcal{E}(x)$ is small. Then, there exists a constant c such that $\exp(-\mathcal{E}(x_{0|t}^{(i)})) \geq c > 0$ and thus for each element $j = 1, .., d$:

$$\|S_K(x_t, t)[j]\| = \left\| \frac{\frac{1}{K}\sum_{i=1}^{K} \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))[j]}{\frac{1}{K}\sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \right\| \tag{34}$$

$$\leq \frac{\|\sum_{i=1}^{K} \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))[j]\|}{Kc} \leq M/c \tag{35}$$

therefore, the jth element of S_K, i.e.$S_K[j]$, is bounded by M/c, suggesting it is sub-Gaussian. While Inequality 21 can be expressed as

$$\sqrt{\sum_{j=1}^{d} (\mathcal{S}_K(x_t, t)[j] - S_t(x_t)[j])^2} \leq \frac{2\sqrt{2v_{0t}(x_t)\log(\frac{2}{\delta})}(1 + \|\nabla \mathcal{E}_t(x_t)\|)}{m_t(x_t)\sqrt{K}} \tag{36}$$

We can roughly derive a bound elementwisely

$$|\mathcal{S}_K(x_t, t)[j] - S_t(x_t)[j]| \leq \frac{2\sqrt{2v_{0t}(x_t)\log(\frac{2}{\delta})}(1 + \|\nabla \mathcal{E}_t(x_t)\|)}{m_t(x_t)\sqrt{Kd}} \tag{37}$$

which suggests that we can approximate the variance of $S_K(x_t, t)[j]$ by leveraging its sub-Gaussianess

$$\text{Var}(S_K(x_t, t)[j]) \approx \frac{4v_{0t}(x_t)(1 + \|\nabla \mathcal{E}_t(x_t)\|)^2}{m_t^2(x_t)Kd} \tag{38}$$

Therefore, according to Equation 25 we can derive that

$$\mathbf{tr}(\text{Cov}[S_K(x_t, t)]) = \sum_{j=1}^{d} \text{Var}[S_K(x_t, t)[j]] \tag{39}$$

$$= \frac{4v_{0t}(x_t)(1 + \|\nabla \mathcal{E}_t(x_t)\|)^2}{m_t^2(x_t)K} \tag{40}$$

$$= 4(1 + \|\nabla \mathcal{E}_t(x_t)\|)^2 \text{Var}[E_K(x_t, t)] \tag{41}$$

In high-energy region. we assume that there exists a direction with a large norm pointing to low energy regions, i.e. $\exists j$ such that $\mathcal{E}(x)$ are positively related to $\nabla \mathcal{E}(x)[j]$. According to Section 9.2 in Owen (2023), the asymptotic variance of a self-normalized importance sampling estimator is given by:

$$\mu = \mathbb{E}_q[f(X)] \tag{42}$$

$$\tilde{\mu}_q = \frac{\sum_{i=1}^{K} w_i f_i}{\sum_{i=1}^{K} w_i} \tag{43}$$

$$\text{Var}(\tilde{\mu}_q) \approx \frac{1}{K}\mathbb{E}_q[w(X)]^{-2}\mathbb{E}_q[w(X)^2(f(X) - \mu)^2] \tag{44}$$

By substituting $\tilde{\mu}_q = S_K(x_t, t)[j]$, $f(X) = -\nabla \mathcal{E}(X)[j]$, $w(X) = \exp(-\mathcal{E}(X))$, $q = N(x; x_t, \sigma_t^2)$, $\mathbb{E}_q[w(X)] = m_t(x_t)$ and $\mathbb{E}_q[w^2(X)] = v_{0t}(x_t) + m_t^2(x_t)$, as well as using $w(X)$ and $f(X)$ are positive related, we have:

$$\text{Var}[S_K(x_t, t)[j]] \geq \frac{1}{K}\mathbb{E}_q[w(X)]^{-2}\mathbb{E}_q[w^2(X)]\mathbb{E}_q[(f(X) - \mu)^2] \tag{45}$$

$$= \frac{v_{0t}(x_t) + m_t^2(x_t)}{m_t^2(x_t)K}\text{Var}_q[\nabla \mathcal{E}(x)[j]] \tag{46}$$

$$\tag{47}$$

Therefore, if we further have a large variance over the system score at this region, i.e. $\text{Var}_q[\nabla\mathcal{E}(x)[j]] > 1$, then we have

$$\text{Var}[S_K(x_t,t)[j]] > \frac{v_{0t}(x_t)}{m_t^2(x_t)K} = \text{Var}[E_K(x_t,t)] \tag{48}$$

and thus $\mathbf{tr}(\text{Cov}[S_K(x_t,t]))$ > $\text{Var}[E_K(x_t,t)]$ holds. □

C PROOF OF PROPOSITION 3

Proposition 3 *Given a bootstrap trajectory $\{s_i\}_{i=0}^n$ such that $\sigma_{s_i}^2 - \sigma_{s_{i-1}}^2 \leq \kappa$, where $s_0 = 0$, $s_n = 1$ and $\kappa > 0$ is a small constant. Suppose E_θ is incrementally optimized from $t \in [0,1]$ as follows: if $t \in [s_i, s_{i+1}]$, $E_\theta(x_t,t)$ targets an energy estimator bootstrapped from $\forall s \in [s_{i-1}, s_i]$ using Eq. 15. For $\forall 0 \leq i \leq n$ and $\forall s \in [s_{i-1}, s_i]$, the variance of the bootstrap energy estimator is given by*

$$\textit{Var}[E_K(x_t,t,s;\theta)] = \frac{v_{st}(x_t)}{v_{0t}(x_t)}\textit{Var}[E_K(x_t,t)] \tag{49}$$

and the bias of $E_K(x_t,t,s;\theta)$ is given by

$$\text{Bias}[E_K(x_t,t,s;\theta)] = \frac{v_{0t}(x_t)}{2m_t^2(x_t)K^{i+1}} + \sum_{j=1}^{i}\frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j}. \tag{50}$$

where $m_z(x_z) = \exp(-\mathcal{E}_z(x_z))$ and $v_{yz}(x_z) = \text{Var}_{\mathcal{N}(x;x_z,(\sigma_z^2-\sigma_y^2)I)}[\exp(-\mathcal{E}_y(x))]$ for $\forall 0 \leq y < z \leq 1$.

Proof. The variance of $E_K(x_t,t,s;\theta)$ can be simply derived by leveraging the variance of a sub-Gaussian random variable similar to Equation 25. While the entire proof for bias of $E_K(x_t,t,s;\theta)$ is organized as follows:

1. we first show the bias of Bootstrap(1) estimator, which is bootstrapped from the system energy
2. we then show the bias of Bootstrap(n) estimator, which is bootstrapped from a lower level noise convolved energy recursively, by induction.

C.1 BOOTSTRAP(1) ESTIMATOR

The Sequential estimator and Bootstrap(1) estimator are defined by:

$$E_K^{\text{Seq}}(x_t,t) := -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-E_K(x_{s|t}^{(i)},s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x;x_t,(\sigma_t^2-\sigma_s^2)I) \tag{51}$$

$$= -\log\frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(ij)})), \quad x_{0|t}^{(ij)} \sim \mathcal{N}(x;x_t,\sigma_t^2 I) \tag{52}$$

$$E_K^{B(1)}(x_t,t,s;\theta) := -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-E_\theta(x_{s|t}^{(i)},s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x;x_t,(\sigma_t^2-\sigma_s^2)I) \tag{53}$$

The mean and variance of a Sequential estimator can be derived by considering it as the MC estimator with K^2 samples:

$$\mathbb{E}[E_K^{\text{Seq}}(x_t,t)] = \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^2} \quad \text{and} \quad \text{Var}(E_K^{\text{Seq}}(x_t,t)) = \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K^2} \tag{54}$$

While an optimal network obtained by targeting the original MC energy estimator 9 at s is [2] :

$$E_{\theta^*}(x_s,s) = \mathbb{E}[E_K(x_s,s)] = -\log m_s(x_s) + \frac{v_{0s}(x_s)}{2m_s^2(x_s)K} \tag{55}$$

[2]We consider minimizing the L_2-norm, *i.e.* $\theta^* = \arg\min_\theta \mathbb{E}_{x_0,t}[\|E_\theta(x_t,t) - E_K(x_t,t)\|^2]$. Since the target, E_K, is noisy, the optimal outputs are given by the expectation, *i.e.* $E_\theta^* = \mathbb{E}[E_K]$.

Then the optimal Bootstrap(1) estimator can be expressed as:

$$E_K^{B(1)}(x_t, t, s; \theta^*) = -\log \frac{1}{K} \sum_{i=1}^{K} \exp\left(-\left(-\log m_s(x_{s|t}^{(i)}) + \frac{v_{0s}(x_{s|t}^{(i)})}{2m_s^2(x_{s|t}^{(i)})K}\right)\right) \tag{56}$$

Before linking the Bootstrap estimator and the Sequential one, we provide the following approximation which is useful. Let a, b two random variables and $\{a_i\}_{i=1}^K$, $\{b_i\}_{i=1}^K$ are corresponding samples. Assume that $\{b_i\}_{i=1}^K$ are close to 0 and concentrated at m_b, while $\{a_i\}_{i=1}^K$ are concentrated at m_a, then

$$\log \frac{1}{K} \sum_{i=1}^{K} \exp(-(a_i + b_i)) = \log \frac{1}{K} \left\{ \sum_{i=1}^{K} \exp(-a_i) \left[\frac{\sum_{i=1}^K \exp(-(a_i + b_i))}{\sum_{i=1}^K \exp(-a_i)} \right] \right\} \tag{57}$$

$$= \log \frac{1}{K} \sum_{i=1}^{K} \exp(-a_i) + \log \frac{\sum_{i=1}^K \exp(-(a_i + b_i))}{\sum_{i=1}^K \exp(-a_i)} \tag{58}$$

$$\approx \log \frac{1}{K} \sum_{i=1}^{K} \exp(-a_i) + \log \frac{\sum_{i=1}^K \exp(-a_i)(1 - b_i)}{\sum_{i=1}^K \exp(-a_i)} \tag{59}$$

$$= \log \frac{1}{K} \sum_{i=1}^{K} \exp(-a_i) + \log \left(1 - \frac{\sum_{i=1}^K \exp(-a_i) b_i}{\sum_{i=1}^K \exp(-a_i)}\right) \tag{60}$$

$$\approx \log \frac{1}{K} \sum_{i=1}^{K} \exp(-a_i) - \frac{\sum_{i=1}^K \exp(-a_i) b_i}{\sum_{i=1}^K \exp(-a_i)} \tag{61}$$

$$\approx \log \frac{1}{K} \sum_{i=1}^{K} \exp(-a_i) - m_b \tag{62}$$

where Approximation applies a first order Taylor expansion of $e^x \approx 1 + x$ around $x = 0$ since $\{b_i\}_{i=1}^K$ are close to 0; while Approximation uses $\log(1 + x) \approx x$ under the same assumption. Notice that when K is large and $\sigma_t^2 - \sigma_s^2 \leq \kappa$ is small , $\{\frac{v_{0s}(x_{s|t}^{(i)})}{2m_s^2(x_{s|t}^{(i)})K}\}_{i=1}^K$ are close to 0 and concentrated at $\frac{v_{0s}(x_t)}{2m_s^2(x_t)K}$. Therefore, by plugging them into Equation 62, Equation 56 can be approximated by

$$E_K^{B(1)}(x_t, t, s; \theta^*) \approx -\log \frac{1}{K} \sum_{i=1}^{K} m_s(x_{s|t}^{(i)}) + \frac{v_{0s}(x_t)}{2m_s^2(x_t)K} \tag{63}$$

When K is large and σ_s^2 is small, the bias and variance of $E_K(x_{s|t}^{(i)}, s)$ are small, then we have

$$-\log \frac{1}{K} \sum_{i=1}^{K} m_s(x_{s|t}^{(i)}) \approx -\log \frac{1}{K} \sum_{i=1}^{K} E_K(x_{s|t}^{(i)}, s) = E_K^{\text{Seq}}(x_t, t) \tag{64}$$

Therefore, the optimal Bootstrap estimator can be approximated as follows:

$$E_K^{B(1)}(x_t, t, s; \theta^*) \approx E_K^{\text{Seq}}(x_t, t) + \frac{v_{0s}(x_t)}{2m_s^2(x_t)K} \tag{65}$$

where its mean and variance depend on those of the Sequential estimator (54):

$$\mathbb{E}[E_K^{B(1)}(x_t, t, s; \theta^*)] = \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_t^2(x_t)K^2} + \frac{v_{0s}(x_t)}{2m_s^2(x_t)K} \tag{66}$$

$$\text{Var}[E_K^{B(1)}(x_t, t, s; \theta^*)] = \frac{v_{0t}(x_t)}{m_t^2(x_t)K^2} \tag{67}$$

C.2 BOOTSTRAP(n) ESTIMATOR

Given a bootstrap trajectory $\{s_i\}_{i=1}^n$ where $s_0 = 0$ and $s_n = s$, and E_θ is well learned at $[0, s]$. Let the energy network be optimal for $u \leq s_n$ by learning a sequence of Bootstrap(i) energy estimators

($i \leq n$). Then the optimal value of $E_\theta(x_s, s)$ is given by $\mathbb{E}[E_K^{B(n-1)}(x_s, s)]$. We are going to show the variance of a Bootstrap(n) estimator by induction. Suppose we have:

$$E_{\theta^*}(x_s, s) = \mathcal{E}_s(x_s) + \sum_{j=1}^{n} \frac{v_{0s_j}(x_s)}{2m_{s_j}^2(x_s)K^j} \tag{68}$$

Then for any $t \in (s, 1]$, the learning target of $E_\theta(x_t, t)$ is bootstrapped from $s_n = s$,

$$E_K^{B(n)}(x_t, t) = -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-E_{\theta^*}(x_{s|t}^{(i)}, s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x; x_t, (\sigma_t^2 - \sigma_s^2)I) \tag{69}$$

$$= -\log \frac{1}{K} \sum_{i=1}^{K} \exp\left(-\mathcal{E}_s(x_{s|t}^{(i)}) - \sum_{j=1}^{n} \frac{v_{0s_j}(x_{s|t}^{(i)})}{2m_{s_j}^2(x_{s|t}^{(i)})K^j}\right) \tag{70}$$

Assume that $\sigma_t^2 - \sigma_s^2$ is small and K is large, then we can apply Approximation 62 and have

$$E_K^{B(n)}(x_t, t) \approx -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}_s(x_{s|t}^{(i)})) + \sum_{j=1}^{n} \frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j} \tag{71}$$

In Bootstrap(n) setting, σ_s^2 is not small and we can't approximate $\mathcal{E}_s(x_{s|t}^{(i)})$ simply by a MC estimator E_K. However, we can sequentially estimate such energy by bootstrapping through the trajectory $\{s_i\}_{i=1}^n$, resembling a Sequential(n) estimator which is equivalent to $E_{K^{n+1}}$,

$$E_K^{B(n)}(x_t, t) \approx E_{K^{n+1}}(x_t, t) + \sum_{j=1}^{n} \frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j} \tag{72}$$

therefore, the optimal output of the energy network at (x_t, t) by learning this Bootstrap(n) estimator is

$$\mathbb{E}[E_K^{B(n)}(x_t, t)] \approx \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_t^2(x_t)K^{n+1}} + \sum_{j=1}^{n} \frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j} \tag{73}$$

which suggests that the accumulated bias of a Bootstrap(n) estimator is given by

$$\frac{v_{0t}(x_t)}{2m_t^2(x_t)K^{n+1}} + \sum_{j=1}^{n} \frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j} \tag{74}$$

□

D INCORPORATING SYMMETRY USING NEM

We consider applying NEM and BNEM in physical systems with symmetry constraints like n-body system. We prove that our MC energy estimator E_K is G-invariant under certain conditions, given in the following Proposition.

Proposition 4 *Let G be the product group $SE(3) \times \mathbb{S}_n \hookrightarrow O(3n)$ and p_0 be a G-invariant density in $\mathbb{R}^d$. Then the Monte Carlo energy estimator of $E_K(x_t, t)$ is G-invariant if the sampling distribution $x_{0|t} \sim \mathcal{N}(x_{0|t}; x_t, \sigma_t^2)$ is G-invariant, i.e.,*

$$\mathcal{N}(x_{0|t}; g \circ x_t, \sigma_t^2) = \mathcal{N}(g^{-1}x_{0|t}; x_t, \sigma_t^2).$$

Proof. Since p_0 is G-invariant, then $\mathcal{E}$ is G-invariant as well. Let $g \in G$ acts on $x \in \mathbb{R}^d$ where $g \circ x = gx$. Since $x_{0|t}^{(i)} \sim \overline{\mathcal{N}}(x_{0|t}; x_t, \sigma_t^2)$ is equivalent to $g \circ x_{0|t}^{(i)} \sim \overline{\mathcal{N}}(x_{0|t}; g \circ x_t, \sigma_t^2)$. Then we have

$$E_K(g \circ x_t, t) = -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(g \circ x_{0|t}^{(i)})) \tag{75}$$

$$= -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)})) = E_K(x_t, t) \tag{76}$$

$$x_{(0|t)}^{(i)} \sim \overline{\mathcal{N}}(x_{0|t}; x_t, \sigma_t^2) \tag{77}$$

Therefore, E_K is invariant to $G = \text{SE}(3) \times \mathbb{S}_n$. □

Furthermore, $E_K(x_t, t, s; \phi)$ is obtained by applying a learned energy network, which is G-invariant, to the analogous process and therefore is G-invariant as well.

E GENERALIZING NEM

This section generalizes NEM in two ways: we first generalize the SDE setting, by considering a broader family of SDEs applied to sampling from Boltzmann distribution; then we generalize the MC energy estimator by viewing it as an importance-weighted estimator.

E.1 NEM FOR GENERAL SDES

Diffusion models can be generalized to any SDEs as $dx_t = f(x_t, t)dt + g(t)dw_t$, where $t \geq 0$ and w_t is a Brownian motion. Particularly, we consider $f(x, t) := -\alpha(t)x$, *i.e.*

$$dx_t = -\alpha(t)x_t dt + g(t)dw_t \tag{78}$$

Then the marginal of the above SDE can be analytically derived as:

$$x_t = \beta(t)x_0 + \beta(t)\sqrt{\int_0^t (g(s)\beta(s))^2 ds}\epsilon, \quad \beta(t) := e^{-\int_0^t \alpha(s)ds} \tag{79}$$

where $\epsilon \sim \mathcal{N}(0, I)$. For example, when $g(t) = \sqrt{\bar{\beta}(t)}$ and $\alpha(t) = \frac{1}{2}\bar{\beta}(t)$, where $\bar{\beta}(t)$ is a monotonic function (*e.g.* linear) increasing from $\bar{\beta}_{\min}$ to $\bar{\beta}_{\max}$, the above SDE resembles a Variance Preserving (VP) process (Song et al., 2020). In DMs, VP can be a favor since it constrains the magnitude of noisy data across t; while a VE process doesn't, and the magnitude of data can explode as the noise explodes. Therefore, we aim to discover whether any SDEs rather than VE can be better by generalizing NEM and DEM to general SDEs.

In this work, we provide a solution for general SDEs (78) rather than a VE SDE. For simplification, we exchangeably use $\beta(t)$ and β_t. Given a SDE as Equation 78 for any integrable functions α and g, we can first derive its marginal as Equation 79, which can be expressed as:

$$\beta_t^{-1}x_t = x_0 + \sqrt{\int_0^t (g(s)\beta(s))^2 ds}\epsilon \tag{80}$$

Therefore, by defining $y_t = \beta_t^{-1}x_t$ we have $y_0 = x_0$ and therefore:

$$y_t = y_0 + \sqrt{\int_0^t (g(s)\beta(s))^2 ds}\epsilon \tag{81}$$

which resembles a VE SDE with noise schedule $\tilde{\sigma}^2(t) = \int_0^t (g(s)\beta(s))^2 ds$. We can also derive this by changing variables:

$$\begin{aligned} dy_t &= (\beta^{-1}(t))' x_t dt + \beta^{-1}(t)dx_t && (82)\\ &= \beta^{-1}(t)\alpha(t)x_t dt + \beta^{-1}(t)(-\alpha(t)x_t dt + g(t)dw_t) && (83)\\ &= \beta^{-1}(t)g(t)dw_t && (84) \end{aligned}$$

which also leads to Equation 81. Let $\tilde{p}_t$ be the marginal distribution of y_t and p_t the marginal distribution of x_t, with $y_{0|t}^{(i)} \sim \mathcal{N}(y; y_t, \tilde{\sigma}_t^2 I)$ we have

$$\tilde{p}_t(y_t) \propto \int \exp(-\mathcal{E}(y))\mathcal{N}(y_t; y, \tilde{\sigma}_t^2 I)dy \tag{85}$$

$$\tilde{S}_t(y_t) = \nabla_{y_t} \log \tilde{p}_t(y_t) \approx \nabla_{y_t} \log \sum_{i=1}^K \exp(-\mathcal{E}(y_{0|t}^{(i)})) \tag{86}$$

$$\tilde{\mathcal{E}}_t(y_t) \approx -\log \frac{1}{K}\sum_{i=1}^K \exp(-\mathcal{E}(y_{0|t}^{(i)})) \tag{87}$$

Therefore, we can learn scores and energies of y_t simply by following DEM and NEM for VE SDEs. Then for sampling, we can simulate the reverse SDE of y_t and eventually, we have $x_0 = y_0$.

Instead, we can also learn energies and scores of x_t. By changing the variable, we can have

$$p_t(x_t) = \beta_t^{-1}\tilde{p}_t(\beta_t^{-1}x_t) = \beta_t^{-1}\tilde{p}_t(y_t) \tag{88}$$

$$S_t(x_t) = \beta_t^{-1}\tilde{S}_t(\beta_t^{-1}x_t) = \beta_t^{-1}\tilde{S}_t(y_t) \tag{89}$$

which provides us the energy and score estimator for x_t:

$$\mathcal{E}_t(x_t) \approx -\log \beta_t^{-1}\frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)})) \tag{90}$$

$$S_t(x_t) \approx \beta_t^{-1}\nabla_{x_t}\log\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)})) \tag{91}$$

$$x_{0|t}^{(i)} \sim \mathcal{N}(x;\beta_t^{-1}x_t,\tilde{\sigma}^2(t)I) \tag{92}$$

Typically, α is a non-negative function, resulting in $\beta(t)$ decreasing from 1 and can be close to 0 when t is large. Therefore, the above equations realize that even though both the energies and scores for a general SDE can be estimated, the estimators are not reliable at large t since β_t^{-1} can be extremely large; while the SDE of y_t (84) indicates that this equivalent VE SDE is scaled by β_t^{-1}, resulting that the variance of y_t at large t can be extremely large and requires much more MC samples for a reliable estimator. This issue can be a bottleneck of generalizing DEM, NEM, and BNEM to other SDE settings, therefore developing more reliable estimators for both scores and energies is of interest in future work.

E.2 MC Energy Estimator as an Importance-Weighted Estimator

As for any SDEs, we can convert the modeling task to a VE process by changing variables, we stick to considering NEM with a VE process. Remember that the MC energy estimator aims to approximate the noised energy given by Eq. 6, which can be rewritten as:

$$\mathcal{E}_t(x_t) = -\log\int\exp(-\mathcal{E}(x))\mathcal{N}(x_t;x,\sigma_t^2 I)dx \tag{93}$$

$$= -\log\int\exp(-\mathcal{E}(x))\frac{\mathcal{N}(x_t;x,\sigma_t^2 I)}{q_{0|t}(x|x_t)}q_{0|t}(x|x_t)dx \tag{94}$$

$$= -\log\mathbb{E}_{q_{0|t}(x|x_t)}\left[\exp(-\mathcal{E}(x))\frac{\mathcal{N}(x_t;x,\sigma_t^2 I)}{q_{0|t}(x|x_t)}\right] \tag{95}$$

The part inside the logarithm of Eq. 95 suggests an Importance Sampling technique for approximation, by using a proposal $q_{0|t}(x|x_t)$. Notice that when choosing a proposal symmetric to the perturbation kernal, *i.e.* $q_{0|t}(x|x_t) = \mathcal{N}(x;x_t,\sigma_t^2)$, Eq. 95 resembles the MC energy estimator we discussed in Section 3.3. Therefore, this formulation allows us to develop a better estimator by carefully selecting the proposal $q_{0|t}(x|x_t)$.

However, Owen (2013) shows that to minimize the variance of the IS estimator, the proposal $q_{0|t}(x|x_t)$ should be chosen roughly proportional to $f(x)\mu_{\text{target}}(x)$, where $f(x) = \exp(-\mathcal{E}(x))$ in our case. Finding such a proposal is challenging in high-dimension space or with a multimodal μ_{target}. A potential remedy can be leveraging Annealed Importance Sampling (AIS; Neal (2001)).

F Memory-Efficient NEM

Differentiating the energy network to get denoising scores in (B)NEM raises additional computations compared with iDEM, which usually twices the computation of forwarding a neural network and can introduce memory overhead due to saving the computational graph. The former issue can be simply solved by reducing the number of integration steps to a half. To solve the latter memory issue, we propose a Memory-efficient NEM by revisiting the Tweedie's formula (Efron, 2011). Given a

VE noising process, $dx_t = g(t)dw_t$, where w_t is Brownian motion and $\sigma_t^2 := \int_{s=0}^{t} g(s)^2 ds$, the Tweedie's formula can be written as:

$$\nabla \log p_t(x_t) = \frac{\mathbb{E}[x_0|x_t] - x_t}{\sigma_t^2} \tag{96}$$

$$\mathbb{E}[x_0|x_t] = \int x_0 p(x_0|x_t) dx_0 \tag{97}$$

$$= \int x_0 \frac{p(x_t|x_0)p_0(x_0)}{p_t(x_t)} dx_0 \tag{98}$$

By revisiting Eq. 5, it's noticable that the noised energy, $\mathcal{E}_t$ (Eq. 6), shares the same partition function as $\mathcal{E}$, *i.e.* $\int \exp(-\mathcal{E}_t(x))dx = \int \exp(-\mathcal{E}(x))dx, \forall t \in [0, 1]$. Hence, Eq. 98 can be simplified as follows, which further suggests a MC estimator for the denoising score with no require of differentiation

$$\mathbb{E}[x_0|x_t] = \int x_0 \frac{\mathcal{N}(x_t; x_0, \sigma_t^2 I) \exp(-\mathcal{E}(x_0))}{\exp(-\mathcal{E}_t(x_t))} dx_0 \tag{99}$$

$$\approx \exp\left(-\left(\mathcal{E}(x_{0|t}) - \mathcal{E}_t(x_t)\right)\right) x_{0|t} \tag{100}$$

where $x_{0|t} \sim \mathcal{N}(x; x_t, \sigma_t^2 I)$. Given learned noised energy, we can approximate this denoiser estimator as follows:

$$D_\theta(x_t, t) := \exp\left(-\left(\mathcal{E}(x_{0|t}) - E_\theta(x_t, t)\right)\right) x_{0|t} \tag{101}$$

$$\tilde{D}_\theta(x_t, t) := \exp\left(-\left(E_\theta(x_{0|t}, 0) - E_\theta(x_t, t)\right)\right) x_{0|t} \tag{102}$$

where we can alternatively use D_θ or $\tilde{D}_\theta$ according to the accessability of clean energy $\mathcal{E}$, the relative computation between $\mathcal{E}(x)$ and $E_\theta(x, t)$, and the accuracy of $E_\theta(x, 0)$.

G TweeDEM: A middle point between Denoising Energy Matching and Energy Flow Matching through Tweedie's formula

In this supplementary work, we propose TWEEDIE DEM (TweeDEM), by leveraging the Tweedie's formula (Efron, 2011) into DEM, *i.e.* $\nabla_x \log p_t(x) = \mathbb{E}_{p(x_0|x_t)}\left[\frac{x_0 - x_t}{\sigma_t^2}\right]$. Surprisingly, TweeDEM can be equivalent to the iEFM-VE proposed by Woo & Ahn (2024), which is a variant of iDEM corresponding to another family of generative model, flow matching.

We first derive an MC denoiser estimator, *i.e.* the expected clean data given a noised data x_t at t

$$\mathbb{E}[x_0|x_t] = \int x_0 p(x_0|x_t) dx_0 \tag{103}$$

$$= \int x_0 \frac{q_t(x_t|x_0)p_0(x_0)}{p_t(x_t)} dx_0 \tag{104}$$

$$= \int x_0 \frac{\mathcal{N}(x_t; x_0, \sigma_t^2 I) \exp(-\mathcal{E}(x_0))}{\exp(-\mathcal{E}_t(x_t))} dx_0 \tag{105}$$

where the numerator can be estimated by an MC estimator $\mathbb{E}_{\mathcal{N}(x_t, \sigma_t^2 I)}[x \exp(-\mathcal{E}(x))]$ and the denominator can be estimated by another similar MC estimator $\mathbb{E}_{\mathcal{N}(x_t, \sigma_t^2 I)}[\exp(-\mathcal{E}(x))]$, suggesting we can approximate this denoiser through self-normalized importance sampling as follows

$$D_K(x_t, t) := \sum_{i=1}^{K} \frac{\exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\sum_{j=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(j)}))} x_{0|t}^{(i)} \tag{106}$$

$$= \sum_{i=1}^{K} w_i x_{0|t}^{(i)} \tag{107}$$

where $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2 I)$, w_i are the importance weights and $D_K(x_t, t) \approx \mathbb{E}[x_0|x_t]$. Then a new MC score estimator can be constructed by plugging the denoiser estimator D_K into Tweedie's formula

$$\tilde{S}_K(x_t, t) := \sum_{i=1}^{K} w_i \frac{x_{0|t}^{(i)} - x_t}{\sigma_t^2} \tag{108}$$

Table 2: Comparison between DEM, DDM, and TweeDEM.

Score estimator: $\sum_i w_i s_i$, with $w_i = Softmax(\tilde{w})[i]$				
Sampler↓ Components→	Weight Type	$\tilde{w}_i$	Score Type	s_i
DEM(Akhound-Sadegh et al., 2024)	System Energy	$\exp(-\mathcal{E}(x_{0\|t}^{(i)}))$	System Score	$-\nabla\mathcal{E}(x_{0\|t}^{(i)}))$
DDM(Karras et al., 2022)	Gaussian Density	$\mathcal{N}(x_{0\|t}^{(i)}; x_t, \sigma_t^2 I)$	Gaussian Score	$\nabla \log \mathcal{N}(x_{0\|t}^{(i)}; x_t, \sigma_t^2 I)$
TweeDEM	System Energy	$\exp(-\mathcal{E}(x_{0\|t}^{(i)}))$	Gaussian Score	$\nabla \log \mathcal{N}(x_{0\|t}^{(i)}; x_t, \sigma_t^2 I)$

where $\frac{x_{0|t}^{(i)} - x_t}{\sigma_t^2}$ resembles the vector fields $v_t(x_t)$ in Flow Matching. In another perspective, these vector fields can be seen as scores of Gaussian, *i.e.* $\nabla \log \mathcal{N}(x; x_t, \sigma_t^2 I)$, and therefore $\tilde{S}_K$ is an importance-weighted sum of Gaussian scores while S_K can be expressed as an importance-weighted sum of system scores $-\nabla\mathcal{E}$. In addition, Karras et al. (2022) demonstrates that in Denoising Diffusion Models, the optimal scores are an importance-weighted sum of Gaussian scores, while these importance weights are given by the corresponding Gaussian density, *i.e.* $S_{\text{DM}}(x_t, t) = \sum_i \tilde{w}_i (x_{0|t}^{(i)} - x_t)/\sigma_t^2$ and $\tilde{w}_i \propto \mathcal{N}(x_{0|t}^{(i)}; x_t, \sigma_t^2 I)$. We summarize these three different score estimators in Table 2.

H Experimental Details

H.1 Energy functions

GMM. A Gaussian Mixture density in 2-dimensional space with 40 modes, which is proposed by Midgley et al. (2023). Each mode in this density is evenly weighted, with identical covariances,

$$\Sigma = \begin{pmatrix} 40 & 0 \\ 0 & 40 \end{pmatrix} \tag{109}$$

and the means $\{\mu_i\}_{i=1}^{40}$ are uniformly sampled from $[-40, 40]^2$, i.e.

$$p_{gmm}(x) = \frac{1}{40} \sum_{i=1}^{40} \mathcal{N}(x; \mu_i, \Sigma) \tag{110}$$

Then its energy is defined by the negative-log-likelihood, i.e.

$$\mathcal{E}^{GMM}(x) = -\log p_{gmm}(x) \tag{111}$$

For evaluation, we sample 1000 data from this GMM with *TORCH.RANDOM.SEED(0)* following Midgley et al. (2023); Akhound-Sadegh et al. (2024) as a test set.

DW-4. First introduced by Köhler et al. (2020), the DW-4 dataset describes a system with 4 particles in 2-dimensional space, resulting in a task with dimensionality $d = 8$. The energy of the system is given by the double-well potential based on pairwise Euclidean distances of the particles,

$$\mathcal{E}^{DW}(x) = \frac{1}{2\tau} \sum_{ij} a(d_{ij} - d_0) + b(d_{ij} - d_0)^2 + c(d_{ij} - d_0)^4 \tag{112}$$

where a, b, c and d_0 are chosen design parameters of the system, τ the dimensionless temperature and $d_{ij} = \|x_i - x_j\|_2$ are Euclidean distance between two particles. Following Akhound-Sadegh et al. (2024), we set $a = 0$, $b = -4$, $c = 0.9$ $d_0 = 4$ and $\tau = 1$, and we use validation and test set from the MCMC samples in Klein et al. (2023a) as the "Ground truth" samples for evaluating.

LJ-n. This dataset describes a system consisting of n particles in 3-dimensional space, resulting in a task with dimensionality $d = 3n$. Following Akhound-Sadegh et al. (2024), the energy of the system is given by $\mathcal{E}^{Tot}(x) = \mathcal{E}^{LJ}(x) + c\mathcal{E}^{osc}(x)$ with the Lennard-Jones potential

$$\mathcal{E}^{\text{LJ}}(x) = \frac{\epsilon}{2\tau} \sum_{ij} \left(\left(\frac{r_m}{d_{ij}}\right)^6 - \left(\frac{r_m}{d_{ij}}\right)^{12} \right) \tag{113}$$

and the harmonic potential

$$\mathcal{E}^{osc}(x) = \frac{1}{2}\sum_i \|x_i - x_{COM}\|^2 \tag{114}$$

where $d_{ij} = \|x_i - x_j\|_2$ are Euclidean distance between two particles, r_m, τ and ϵ are physical constants, x_{COM} refers to the center of mass of the system and c the oscillator scale. We use $r_m = 1$, $\tau = 1$, $\epsilon = 1$ and $c = 0.5$ the same as Akhound-Sadegh et al. (2024). We test our models in LJ-13 and LJ-55, which correspond to $d = 65$ and $d = 165$ respectively. And we use the MCMC samples given by Klein et al. (2023a) as a test set.

H.2 Evaluation Metrics

2-Wasserstein distance $\mathcal{W}_2$. Given empirical samples μ from the sampler and ground truth samples ν, the 2-Wasserstein distance is defined as:

$$\mathcal{W}_2(\mu, \nu) = (\inf_\pi \int \pi(x, y) d^2(x, y) dx dy)^{\frac{1}{2}} \tag{115}$$

where π is the transport plan with marginals constrained to μ and ν respectively. Following Akhound-Sadegh et al. (2024), we use the Hungarian algorithm as implemented in the Python optimal transport package (POT) (Flamary et al., 2021) to solve this optimization for discrete samples with the Euclidean distance $d(x, y) = \|x - y\|_2$. $x - \mathcal{W}_2$ is based on the data and $\mathcal{E} - \mathcal{W}_2$ is based on the corresponding energy.

Total Variation (TV). The total variation measures the dissimilarity between two probability distributions. It quantifies the maximum difference between the probabilities assigned to the same event by two distributions, thereby providing a sense of how distinguishable the distributions are. Given two distribution P and Q, with densities p and q, over the same sample space Ω, the TV distance is defined as

$$TV(P, Q) = \frac{1}{2}\int_\Omega |p(x) - q(x)| dx \tag{116}$$

Following Akhound-Sadegh et al. (2024), for low-dimentional datasets like GMM, we use 200 bins in each dimension. For larger equivariant datasets, the total variation distance is computed over the distribution of the interatomic distances of the particles.

H.3 Experiment Settings

We pin the number of reverse SDE integration steps for iDEM, NEM, BNEM and TweeDEM (see Appendix G) as 1000 and the number of MC samples as 1000 in most experiments, except for the ablation studies.

GMM-40. For the basic model f_θ, we use an MLP with sinusoidal and positional embeddings which has 3 layers of size 128 as well as positional embeddings of size 128. The replay buffer is set to a maximum length of 10000.

During training, the generated data was in the range $[-1, 1]$ so to calculate the energy it was scaled appropriately by unnormalizing by a factor of 50. Baseline models are trained with a geometric noise schedule with $\sigma_{\min} = 1e-5$, $\sigma_{\max} = 1$; NEM and BNEM are trained with a cosine noise schedule with $\sigma_{\min} = 0.001$ and $\sigma_{\max} = 1$. We use $K = 500$ samples for computing the Bootstrap energy estimator E_K^B. We clip the norm of S_K, s_θ and ∇E_θ to 70 during training and sampling. The variance controller for BNEM is set to be $\beta = 0.2$. All models are trained with a learning rate of $5e-4$.

DW-4. All models use an EGNN with 3 message-passing layers and a 2-hidden layer MLP of size 128. All models are trained with a geometric noise schedule with $\sigma_{\min} = 1e-5$, $\sigma_{\max} = 3$ and a learning rate of $1e-3$ for computing S_K and E_K. We use $K = 500$ samples for computing the Bootstrap energy estimator E_K^B. We clip the norm of S_K, s_θ, and ∇E_θ to 20 during training and sampling. The variance controller for BNEM is set to be $\beta = 0.2$.

LJ-13. All models use an EGNN with 5 hidden layers and hidden layer size 128. Baseline models are trained with a geometric noise schedule with $\sigma_{\min} = 0.01$ and $\sigma_{\max} = 2$; NEM and BNEM are

Table 3: Neural sampler performance comparison for 4 different energy functions. We measured the performance using data Wasserstein-2 distance (x-$\mathcal{W}_2$), Energy Wasserstein-2 distance ($\mathcal{E}$-$\mathcal{W}_2$), and Total Variation (TV). * indicates divergent training. Each sampler is evaluated with 3 random seeds and we report the mean $\pm$ standard-deviation for each metric.

Energy →	**GMM-40** ($d = 2$)			**DW-4** ($d = 8$)			**LJ-13** ($d = 39$)			**LJ-55** ($d = 165$)		
Sampler ↓	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**
DDS	$15.04_{\pm 2.97}$	$305.13_{\pm 186.06}$	$0.96_{\pm 0.01}$	$0.82_{\pm 0.21}$	$558.79_{\pm 787924.86}$	$0.38_{\pm 0.14}$	*	*			*	
PIS	$6.58_{\pm 1.68}$	$79.86_{\pm 7.79}$	$0.95_{\pm 0.01}$	*	*	*	*		*	*	*	*
FAB	$9.08_{\pm 1.41}$	$47.60_{\pm 7.24}$	$0.79_{\pm 0.07}$	$0.62_{\pm 0.02}$	$112.70_{\pm 20.33}$	$0.38_{\pm 0.02}$	*	*	*	*	*	*
iDEM	$8.21_{\pm 5.43}$	$60.49_{\pm 70.12}$	$0.82_{\pm 0.03}$	$0.50_{\pm 0.03}$	$2.80_{\pm 1.72}$	$0.16_{\pm 0.01}$	$0.87_{\pm 0.00}$	$77515.90_{\pm 115028.07}$	$0.06_{\pm 0.01}$	$2.06_{\pm 0.04}$	$169347.94_{\pm 260160.98}$	$0.16_{\pm 0.02}$
NEM (ours)	$5.28_{\pm 0.89}$	$44.56_{\pm 39.56}$	$0.91_{\pm 0.02}$	$\mathbf{0.48}_{\pm 0.02}$	$0.85_{\pm 0.52}$	$\mathbf{0.14}_{\pm 0.01}$	$0.87_{\pm 0.01}$	$5.01_{\pm 3.14}$	$\mathbf{0.03}_{\pm 0.00}$	$1.90_{\pm 0.01}$	$\mathbf{118.58}_{\pm 106.63}$	$\mathbf{0.10}_{\pm 0.02}$
BNEM (ours)	$\mathbf{3.66}_{\pm 0.30}$	$\mathbf{1.87}_{\pm 1.00}$	$\mathbf{0.79}_{\pm 0.04}$	$0.49_{\pm 0.02}$	$\mathbf{0.38}_{\pm 0.09}$	$\mathbf{0.14}_{\pm 0.01}$	$\mathbf{0.86}_{\pm 0.00}$	$\mathbf{1.02}_{\pm 0.69}$	$\mathbf{0.03}_{\pm 0.00}$	$\mathbf{1.87}_{\pm 0.05}$	$821060.12_{\pm 1397674.81}$	$0.15_{\pm 0.07}$

Table 4: Ablation Study on applying energy smoothing based on Cubic Spline for iDEM

Energy →	**LJ-55** ($d = 165$)		
Sampler ↓	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**
iDEM	2.077 ± 0.0238	169347 ± 2601160	0.165 ± 0.0146
iDEM (cubic spline smoothed)	2.086 ± 0.0703	12472 ± 8520	0.142 ± 0.0095
NEM (ours)	$\mathbf{1.898} \pm 0.0097$	$\mathbf{118.57} \pm 106.62$	$\mathbf{0.0991} \pm 0.0194$

trained with a geometric noise schedule with $\sigma_{\min} = 0.001$ and $\sigma_{\max} = 6.0$ to ensure the data well mixed to Gaussian. We use a learning rate of $1e-3$, $K = 500$ samples for E_K^B, and we clipped S_K, s_θ and ∇E_θ to a max norm of 20 during training and sampling. The variance controller for BNEM is set to be $\beta = 0.5$.

LJ-55. All models use an EGNN with 5 hidden layers and hidden layer size 128. All models are trained with a geometric noise schedule with $\sigma_{\min} = 0.5$ and $\sigma_{\max} = 4$. We use a learning rate of $1e-3$, $K = 500$ samples for E_K^B. We clipped S_K and s_θ to a max norm of 20 during training and sampling. And we clipped ∇E_θ to a max norm of 1000 during sampling, as our model can capture better scores and therefore a small clipping norm can be harmful for sampling. The variance controller for BNEM is set to be $\beta = 0.4$.

For all datasets. We use clipped scores as targets for iDEM and TweeDEM training for all tasks. Meanwhile, we also clip scores during sampling in outer-loop of training, when calculating the reverse SDE integral. These settings are shown to be crucial especially when the energy landscape is non-smooth and exists extremely large energies or scores, like LJ-13 and LJ-55. In fact, targeting the clipped scores refers to learning scores of smoothed energies. While we're learning unadjusted energy for NEM and BNEM, the training can be unstable, and therefore we often tend to use a slightly larger $\sigma_{\min}$. Also, we smooth the Lennard-Jones potential through the cubic spline interpolation, according to Moore et al. (2024). Besides, we predict per-particle energies for DW-4 and LJ-n datasets, which can provide more information on the energy system. It shows that this setting can significantly stabilize training and boost performance.

I Supplementary Experiments

I.1 Main Results

We report a detailed version of the main table 1 in Table 3, which includes the mean and standard deviation of metrics.

I.2 Comparing the Robustness of Energy-Matching and Score-Matching

In this section, we discussed the robustness of the energy-matching model(NEM) with the score-matching model(DEM) by analyzing the influence of the numbers of MC samples used for estimators and choice of noise schedule on the sampler's performance.

Robustness with limited compute budget. We first complete the robustness discussed in Section 5, by conducting experiments on a more complex benchmark - LJ-13. reports different metrics of each

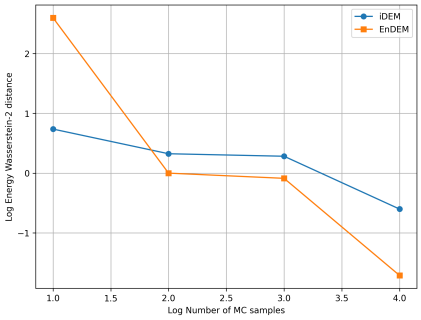


Figure 6: Comparison of the Energy Wasserstein-2 distance in DW4 benchmark between DEM and NEM across varying numbers of MC samples.

Table 5: Neural sampler performance comparison for 3 different energy functions. The number after the sampler, *e.g.* NEM-100, represents the number of integration steps and MC samples is 100. We measured the performance using data Wasserstein-2 distance(x-$\mathcal{W}_2$), Energy Wasserstein-2 distance($\mathcal{E}$-$\mathcal{W}_2$) and Total Variation(TV).

Energy →	**GMM-40** ($d = 2$)			**DW-4** ($d = 8$)			**LJ-13** ($d = 39$)		
Sampler ↓	**x-$\mathcal{W}_2$**↓	**$\mathcal{E}$-$\mathcal{W}_2$**↓	**TV**↓	**x-$\mathcal{W}_2$**↓	**$\mathcal{E}$-$\mathcal{W}_2$**↓	**TV**↓	**x-$\mathcal{W}_2$**↓	**$\mathcal{E}$-$\mathcal{W}_2$**↓	**TV**↓
iDEM-1000	$4.21_{\pm 0.86}$	$1.63_{\pm 0.61}$	$0.81_{\pm 0.03}$	$\mathbf{0.42}_{\pm 0.02}$	$1.89_{\pm 0.56}$	$\mathbf{0.13}_{\pm 0.01}$	$0.87_{\pm 0.00}$	$77515.90_{\pm 115028.07}$	$0.06_{\pm 0.01}$
iDEM-100	$8.21_{\pm 5.43}$	$60.49_{\pm 70.12}$	$0.82_{\pm 0.03}$	$0.50_{\pm 0.03}$	$2.80_{\pm 1.72}$	$0.16_{\pm 0.01}$	$0.88_{\pm 0.00}$	$1190.59_{\pm 590,290}$	$0.07_{\pm 0.00}$
NEM-1000	$2.73_{\pm 0.55}$	$1.68_{\pm 0.98}$	$0.81_{\pm 0.00}$	$0.46_{\pm 0.02}$	$\mathbf{0.28}_{\pm 0.08}$	$0.28_{\pm 0.13}$	$0.02_{\pm 0.01}$	$5.01_{\pm 2.56}$	$\mathbf{0.03}_{\pm 0.00}$
NEM-100	$5.28_{\pm 0.89}$	$44.56_{\pm 39.56}$	$0.91_{\pm 0.02}$	$0.48_{\pm 0.02}$	$0.85_{\pm 0.52}$	$0.14_{\pm 0.01}$	$0.88_{\pm 0.00}$	$13.14_{\pm 225.45}$	$0.04_{\pm 0.00}$
BNEM-1000	$\mathbf{2.55}_{\pm 0.47}$	$\mathbf{0.36}_{\pm 0.12}$	$\mathbf{0.66}_{\pm 0.08}$	$0.49_{\pm 0.01}$	$\mathbf{0.29}_{\pm 0.05}$	$0.15_{\pm 0.01}$	$\mathbf{0.86}_{\pm 0.00}$	$\mathbf{0.62}_{\pm 0.01}$	$\mathbf{0.03}_{\pm 0.00}$
BNEM-100	$3.66_{\pm 0.30}$	$1.87_{\pm 1.00}$	$0.79_{\pm 0.04}$	$0.49_{\pm 0.02}$	$0.38_{\pm 0.09}$	$0.14_{\pm 0.01}$	$0.87_{\pm 0.00}$	$5.93_{\pm 3.01}$	$\mathbf{0.03}_{\pm 0.00}$

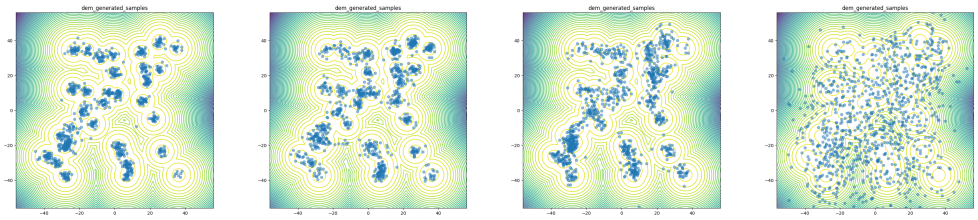
sampler in different settings, *i.e.* 1000 integration steps and MC samples v.s. 100 integration steps and MC samples, and different tasks.

Robustness v.s. Number of MC samples. As in Figure 6, NEM consistently outperforms iDEM when more than 100 MC samples are used for the estimator. Besides, NEM shows a faster decline when the number of MC samples increases. Therefore, we can conclude that the low variance of Energy-matching makes it more beneficial when we boost with more MC samples.

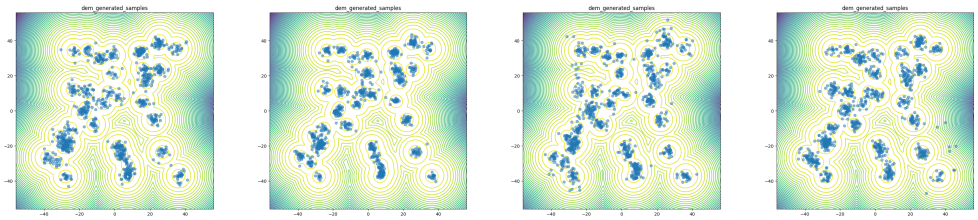
Robustness v.s. Different noise schedules. Then, we evaluate the performance differences when applying various noise schedules. The following four schedules were tested in the experiment:

- **Geometric noise schedule**: The noise level decreases geometrically in this schedule. The noise at step t is given by: $\sigma_t = \sigma_0^{1-t} \cdot \sigma_1^t$ where $\sigma_0 = 0.0001$ is the initial noise level, $\sigma_1 = 1$ is the maximum noise level, and t is the time step.
- **Cosine noise schedule**: The noise level follows a cosine function over time, represented by: $\sigma_t = \sigma_1 \cdot \cos(\pi/2 \frac{1+\delta-t}{1+\delta})^2$, where $\delta = 0.008$ is a hyper-parameter that controls the decay rate.
- **Quadratic noise schedule**: The noise level follows a quadratic decay:$\sigma_t = \sigma_0 t^2$ where σ_0 is the initial noise level. This schedule applies a slow decay initially, followed by a more rapid reduction.
- **Linear noise schedule**: In this case, the noise decreases linearly over time, represented as: $\sigma_t = \sigma_1 t$

The experimental results are depicted in Figure 7. It is pretty obvious that for iDEM the performance varied for different noise schedules. iDEM favors noise schedules that decay more rapidly to 0 when t approaches 0. When applying the linear noise schedules, the samples are a lot more noisy than other schedules. This also proves our theoretical analysis that the variance would make the score network hard to train. On the contrary, all 4 schedules are able to perform well on NEM. This illustrates that the reduced variance makes NEM more robust and requires less hyperparameter tuning.



(a) Geometric: 4.16 (b) Cosine: 6.32 (c) Quadratic: 3.95 (d) Linear: 9.87



(a) Geometric: 2.64 (b) Cosine: 2.29 (c) Quadratic: 2.53 (d) Linear: 3.13

Figure 7: Comparison of different sampler (Above: iDEM; Below: NEM) when employing different noise schedules. The performances of x-$\mathcal{W}_2$ are listed.

Robustness in terms of Outliers. Based on Figure 3, we set the maximum energy as (GMM-40: 100; DW-4: 0; LJ-13: 0; LJ-55: -150). We remove outliers based on these thresholds and recomputed the $\mathcal{E}$-$\mathcal{W}_2$. We report the new values as well as percentage of outliers in Table 6, which shows that the order of performance (BNEM>NEM>iDEM) still holds in terms of better $\mathcal{E}$-$\mathcal{W}_2$ value and less percentage of outliers.

Table 6: $\mathcal{E}$-$\mathcal{W}_2$ w/o outliers (outlier%) for different models and datasets. **Bold** indicates the best value and underline indicates the second one.

Sampler↓ Energy→	**GMM-40** ($d = 2$)	**DW-4** ($d = 8$)	**LJ-13** ($d = 39$)	**LJ-55** ($d = 165$)
iDEM	0.138 (0.0%)	8.658 (0.02%)	88.794 (4.353%)	21255 (0.29%)
NEM (ours)	0.069 (0.0%)	4.715 (**0.0%**)	5.278 (0.119%)	98.206 (0.020%)
BNEM (ours)	**0.032** (0.0%)	**1.050** (**0.0%**)	**1.241** (**0.025%**)	**11.401** (**0.0%**)

I.3 Empirical Analysis of the Variance of E_K and S_K

To justify the theoretical results for the variance of the MC energy estimator (9) and MC score estimator (7), we first empirically explore a 2D GMM. For better visualization, the GMM is set to be evenly weighted by 10 modes located in $[-1, 1]^2$ with identical variance $1/40$ for each component, resulting in the following density

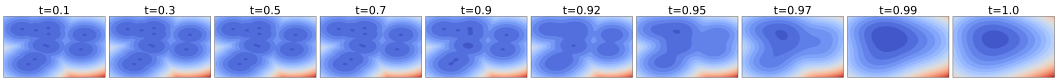
$$p'_{GMM}(x) = \frac{1}{10}\sum_{i=1}^{10}\mathcal{N}\left(x; \mu_i, \frac{1}{40}I\right) \tag{117}$$

while the marginal perturbed distribution at t can be analytically derived from Gaussian's property:

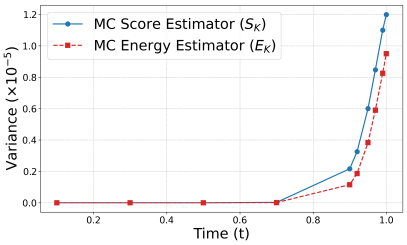
$$p_t(x) = (p'_{GMM} * \mathcal{N}(0, \sigma_t^2))(x_t) = \frac{1}{10}\sum_{i=1}^{10}\mathcal{N}\left(x; \mu_i, \left(\frac{1}{40} + \sigma_t^2\right)I\right) \tag{118}$$

given a VE noising process.

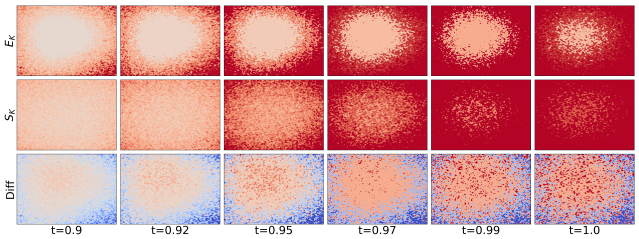
We empirically estimate the variance for each pair of (x_t, t) by simulating 10 times the MC estimators. Besides, we estimate the expected variance over x for each time t, i.e. $\mathbb{E}_{p_t(x_t)}[\text{Var}(E_K(x_t, t))]$ and $\mathbb{E}_{p_t(x_t)}[\text{Var}(S_K(x_t, t))]$.



(a) Ground truth energy across $t \in [0, 1]$



(b) Expected variance of estimators



(c) Point-wise variance for $t \in [0.9, 1]$

Figure 8: (a) the ground truth energy of the target GMM from $t = 0$ to $t = 1$; (b) the estimation of expected variance of x from $t = 0$ to $t = 1$, computed by a weighted sum over the variance of estimator at each location with weights equal to the marginal density p_t; (c) the variance of MC score estimator and MC energy estimator, and their difference (Var[score]-Var[energy]) for t from 0.9 to 1, we ignore the plots from $t \in [0, 0.9]$ since the variance of both estimators are small. The colormap ranges from blue (low) to red (high), where blues are negative and reds are positive.

Figure 8a shows that, the variance of both MC energy estimator and MC score estimator increase as time increases. In contrast, the variance of E_K can be smaller than that of S_K in most areas, especially when the energies are low (see Figure 8c), aligning our Proposition 2. Figure 8b shows that in expectation over true data distribution, the variance of E_K is always smaller than that of S_K across $t \in [0, 1]$.

I.4 Empirical Analysis of the Bias of Bootstrapping

To show the improvement gained by bootstrapping, we deliver an empirical study on the GMM-40 energy in this section. As illustrated in Section H.3, the modes of GMM-40 are located between $[-40, 40]^2$ with small variance. Therefore, the sub-Gaussianess assumption is natural. According to Proposition A and Proposition B, the analytical bias of the MC energy estimator (Eq. 9) and Bootstrapped energy estimator (Eq. 49) can be computed by Eq. 24 and Eq. **??**, respectively. We provide these two bias terms here for reference,

$$\text{Bias}(E_K(x_t, t)) = \frac{v_{0t}(x_t)}{2m_t(x_t)^2 K} \tag{119}$$

$$\text{Bias}(E_K(x_t, t, s; \theta)) = \frac{v_{0t}(x_t)}{2m_t^2(x_t)K^{n+1}} + \sum_{j=1}^{n} \frac{v_{0s_j}(x_t)}{2m_{s_j}^2(x_t)K^j} \tag{120}$$

Given a Mixture of Gaussian with K components, $p_0(x) = \sum_k \pi_k \mathcal{N}(x; \mu_k, \Sigma_k)$ and $\mathcal{E}(x) = -\log p_0(x)$, $m_t(x)$ and v_{0t} can be calculated as follows:

$$m_t(x_t) = \exp(-\mathcal{E}_t(x_t)) \tag{121}$$

$$= \int \mathcal{N}(x_t; x, \sigma_t^2 I) \exp(-\mathcal{E}(x)) dx \tag{122}$$

$$= \int \mathcal{N}(x_t; x, \sigma_t^2 I) p_0(x) dx \tag{123}$$

$$= \sum_k \pi_k \int \mathcal{N}(x_t; x, \sigma_t^2 I) \mathcal{N}(x; \mu_k, \Sigma_k) dx \tag{124}$$

$$= \sum_k \pi_k \mathcal{N}(x_t; \mu_k, \sigma_t^2 I + \Sigma_k) \tag{125}$$

$$v_{0t}(x_t) = \mathrm{Var}_{\mathcal{N}(x; x_t, \sigma_t^2 I)}(\exp(-\mathcal{E}(X))) \tag{126}$$

$$= \int \mathcal{N}(x_t; x, \sigma_t^2 I) p_0(x)^2 dx - m_t^2(x_t) \tag{127}$$

$$= \sum_{j,k} \pi_j \pi_k \int \mathcal{N}(x_t; x, \sigma_t^2 I) \mathcal{N}(x; \mu_k, \Sigma_k) \mathcal{N}(x; \mu_j, \Sigma_j) dx - m_t^2(x_t) \tag{128}$$

$$= \sum_{j,k} \pi_j \pi_k \int \mathcal{N}(x_t; x, \sigma_t^2 I) \mathcal{N}(x; \mu_{jk}, \Sigma_{jk}) C_{jk} \frac{|\Sigma_{jk}|^{1/2}}{(2\pi)^{d/2} |\Sigma_j|^{1/2} |\Sigma_k|^{1/2}} dx - m_t^2(x_t) \tag{129}$$

$$= \sum_{j,k} \pi_j \pi_k C_{jk} \frac{|\Sigma_{jk}|^{1/2}}{(2\pi)^{d/2} |\Sigma_j|^{1/2} |\Sigma_k|^{1/2}} \mathcal{N}(x_t; \mu_{jk}, \sigma_t^2 I + \Sigma_{jk}) - m_t^2(x_t) \tag{130}$$

where

$$\Sigma_{jk} = (\Sigma_j^{-1} + \Sigma_k^{-1})^{-1} \tag{131}$$

$$\mu_{jk} = \Sigma_{jk}(\Sigma_j^{-1}\mu_j + \Sigma_k^{-1}\mu_k) \tag{132}$$

$$C_{jk} = \exp\left(-\frac{1}{2}\left[\mu_j^\top \Sigma_j^{-1} \mu_j + \mu_k^\top \Sigma_k^{-1} \mu_k - (\Sigma_j^{-1}\mu_j + \Sigma_k^{-1}\mu_k)^\top \Sigma_{jk} (\Sigma_j^{-1}\mu_j + \Sigma_k^{-1}\mu_k)\right]\right) \tag{133}$$

In our GMM-40 case, the covariance for each component are identical and diagonal, *i.e.* $\Sigma_k \equiv \Sigma = vI$. By plugging it into the equations, we can simplify the $m_t(x_t)$ and $v_{0t}(x_t)$ terms as follows

$$m_t(x_t) = \sum_k \frac{1}{K} \mathcal{N}(x_t; \mu_k, (\sigma_t^2 + v)I) \tag{134}$$

$$v_{0t}(x_t) = \sum_{j,k} \frac{1}{K^2} \frac{\exp\left(-\frac{1}{4v}(\mu_j - \mu_k)^\top (\mu_j - \mu_k)\right)}{\sqrt{2}(2\pi v)} \mathcal{N}\left(x_t; \frac{1}{2}(\mu_j + \mu_k), (\sigma_t^2 + v/2)I\right) - m_t^2(x_t) \tag{135}$$

We computed the analytical bias terms and visualize in Figure 9. Figure 9a visualizes the bias of the both NEM and BNEM over the entire space. It shows that (1) bootstrapped energy estimator can have less bias (contributed by bias of EK and variance of training target); (2) If E_K is already bias, i.e. the "red" regions in the first row of Figure 9a when $t = 0.1$, bootstrapping can not gain any improvement, which is reasonable; (3) However, if E_K has low bias, i.e. the "blue" regions when $t = 0.1$, the Bootstrapped energy estimator can result in lower bias estimation, superioring MC energy estimator; (4) In low energy region, both MC energy estimator and Bootstrapped one result in accurate estimation. However, in a bi-level iterated training fashion, we always probably explore high energy at the beginning. Therefore, due to the less biasedness of Bootstrapped estimator at high energy regions, we're more likely to have more informative pseudo data which can further improve the model iteratively.

On the other hand, we ablate different settings of Num. of MC samples and the variance-control (VC) parameter. We visualize the results in Figure 9b. The results show that, with proper VC, bootstrapping

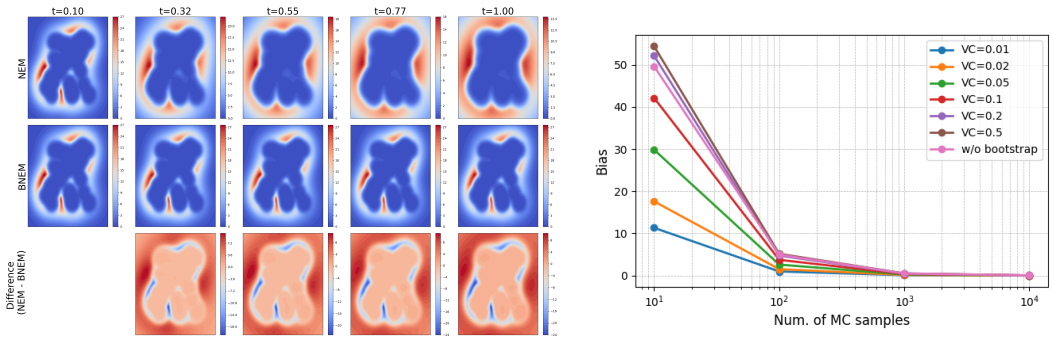
Table 7: Time complexity of different neural samplers.

Sampler↓ Phase→	Inner-loop	Outer-loop (or Sampling)
iDEM	$\mathcal{O}\left(L(2K\Gamma_{\mathcal{E}}(B) + 2\Gamma_{\text{NN}}(B))\right)$	$\mathcal{O}\left(T\Gamma_{\text{NN}}(B)\right)$
NEM (ours)	$\mathcal{O}\left(L(K\Gamma_{\mathcal{E}}(B) + 2\Gamma_{\text{NN}}(B))\right)$	$\mathcal{O}\left(2T\Gamma_{\text{NN}}(B)\right)$
BNEM (ours)	$\mathcal{O}\left(L(K\Gamma_{\text{NN}}(B) + 2\Gamma_{\text{NN}}(B))\right)$	$\mathcal{O}\left(2T\Gamma_{\text{NN}}(B)\right)$

Table 8: Time comparison (in seconds) of different samplers for both inner-loop and outer-loop across different energies.

Energy →	**GMM-40** ($d = 2$)		**DW-4** ($d = 8$)		**LJ-13** ($d = 39$)		**LJ-55** ($d = 165$)	
Sampler ↓	**Inner-loop**	**Outer-loop**	**Inner-loop**	**Outer-loop**	**Inner-loop**	**Outer-loop**	**Inner-loop**	**Outer-loop**
iDEM	1.657	1.159	6.783	2.421	21.857	21.994	36.158	47.477
NEM (ours)	1.658	2.252	5.217	8.517	14.646	52.563	17.171	114.601
BNEM (ours)	1.141	2.304	25.552	7.547	68.217	52.396	113.641	115.640

allows us to reduce the bias with less MC samples, which is desirable in high-dimensional and more complex problems.



(a) Bias with VC=0.1

(b) Bias with different VC and Num. MC samples.

Figure 9: Empirical analysis on bias with bootstrapping, on GMM-40.

I.5 Complexity Analysis

To compare the time complexity between iDEM, NEM and BNEM, we let: (1) In the outer-loop of training, we have T integration steps and batch size B; (2) In the inner-loop, we have L epochs and batch size B. Let $\Gamma_{\text{NN}}(B)$ be the time complexity of evaluating a neural network w.r.t. B data points, $\Gamma_{\mathcal{E}}(B)$ be the time complexity of evaluating the clean energy w.r.t. B data points, and K be the number of MC samples used. Since differentiating a function f using the chain rule requires approximately twice the computation as evaluating f, we summurise the time complexity of iDEM, NEM, and BNEM in Table 7. It shows that in principle, in the inner-loop, NEM can be slightly faster than iDEM, while BNEM depends on the relativity between complexity of evaluating the neural network and evaluating the clean energy function.

Table 8 reports the time usage per inner-loop and outer-loop. It shows that due to the need for differentiation, the sampling time, *i.e.* outer-loop, of BNEM/NEM is approximately twice that of iDEM. In contrast, the inner-loop time of NEM is slightly faster than that of iDEM, matching the theoretical time complexity, and the difference becomes more pronounced for more complex systems such as LJ-13 and LJ-55. For BNEM, the sampling time is comparable to NEM, but the inner-loop time depends on the relative complexity of evaluating the clean energy function versus the neural network, which can be relatively higher.

Table 9: Comparison between iDEM, NEM, and BNEM, with similar computational budget.

Energy →	**LJ-13** ($d = 39$)			**LJ-55** ($d = 165$)		
Sampler ↓	**x-**$\mathcal{W}_2$↓	$\mathcal{E}$**-**$\mathcal{W}_2$↓	**TV**↓	**x-**$\mathcal{W}_2$↓	$\mathcal{E}$**-**$\mathcal{W}_2$↓	**TV**↓
iDEM	0.870	6670	0.0600	2.060	17651	0.160
NEM-500 (ours)	0.870	31.877	0.0377	1.896	11018	0.0955
BNEM-500 (ours)	0.866	2.242	0.0329	1.890	25277	0.113

Table 10: Performance comparison between NEM and ME-NEM.

Energy →	**GMM-40** ($d = 2$)			**DW-4** ($d = 8$)			**LJ-13** ($d = 39$)		
Sampler ↓	**x-**$\mathcal{W}_2$↓	$\mathcal{E}$**-**$\mathcal{W}_2$↓	**TV**↓	**x-**$\mathcal{W}_2$↓	$\mathcal{E}$**-**$\mathcal{W}_2$↓	**TV**↓	**x-**$\mathcal{W}_2$↓	$\mathcal{E}$**-**$\mathcal{W}_2$↓	**TV**↓
NEM (ours)	1.808	0.846	0.838	0.479	2.956	0.14	0.866	27.362	0.0369
ME-NEM (ours)	2.431	0.107	0.813	0.514	1.649	0.164	0.88	20.161	0.0338

I.6 Performance Gain Under the Same Computational Budget

It's noticable that computing the scores by differentiating the energy network outputs, *i.e.* $\nabla_{x_t} E_\theta(x_t, t)$, requires twice of computation compared with iDEM which computes $s_\theta(x_t, t)$ by one neural network evaluation. In this section, we limit the computational budget during sampling of both NEM and BNEM by reducing their integration steps to half. We conduct experiment on LJ-13 and LJ-55, where we reduce the reverse SDE integration steps in both NEM and BNEM from 1000 to 500. Metrics are reported in Table 9. It shows that with similar computation, NEM and BNEM can still outperform iDEM.

I.7 Experiments for Memory-Efficient NEM

In this section, we conduct experiments on the proposed Memory-Efficient NEM (ME-NEM). The number of integration steps and MC samples are all set to 1000. ME-NEM is proposed to reduce the memory overhead caused by differentiating the energy network in NEM, which leverages the Tweedie's formula to establish a 1-sample MC estimator for the denoising score. In principle, ME-NEM doesn't require neural network differentiation, avoiding saving the computational graph. Though it still requires evaluating the neural network twice (see Eq. 102), this only requires double memory usage of iDEM and can be computed parallelly, resulting in a similar speed of sampling with iDEM. In this section, we simply show a proof-of-concept experiment on ME-NEM, while leaving more detailed experiments as our future work.

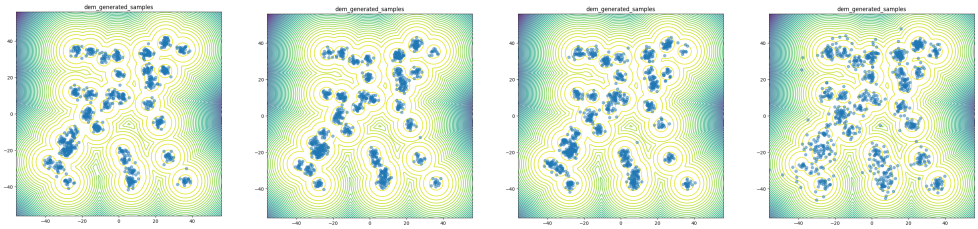
Table 10 reports the performance of NEM and ME-NEM, showcasing that ME-NEM can achieve similar results even though it leverages another MC estimator during sampling.

I.8 Experiments for TweeDEM

In Appendix G, we propose TweeDEM, a variant of DEM by leveraging Tweedie's formula (Efron, 2011), which theoretically links iDEM and iEFM-VE and suggests that we can simply replace the score estimator S_K (7) with $\tilde{S}_K$ (108) to reconstruct a iEFM-VE. We conduct experiments for this variant with the aforementioned GMM and DW-4 potential functions.

Setting. We follow the ones aforementioned, but setting the steps for reverse SDE integration 1000, the number of MC samples 500 for GMM and 1000 for DW-4. We set a quadratic noise schedule ranging from 0 to 3 for TweeDEM in DW-4.

To compare the two score estimators S_K and $\tilde{S}_K$ fundamentally, we first conduct experiments using these ground truth estimations for reverse SDE integration, *i.e.* samplers without learning. In addition, we consider using a neural network to approximate these estimators, *i.e.* iDEM and TweeDEM.



(a) $\tilde{S}_K$ (ours) (b) S_K (c) TweeDEM (ours) (d) DEM

Figure 10: Sampled points from samplers applied to GMM-40 potentials, with the ground truth represented by contour lines. $\tilde{S}_K$ and S_K represent using these ground truth estimators for reverse SDE integration.

Table 11: Sampler performance comparison for GMM-40 and DW-4 energy function. we measured the performance using data Wasserstein-2 distance(x-$\mathcal{W}_2$), Energy Wasserstein-2 distance($\mathcal{E}$-$\mathcal{W}_2$), and Total Variation(TV). † We compare the optimal number reported by Woo & Ahn (2024) and Akhound-Sadegh et al. (2024). . - indicates metric non-reported.

Energy →	**GMM-40** ($d = 2$)			**DW-4** ($d = 8$)		
Sampler ↓	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**	**x-$\mathcal{W}_2$↓**	**$\mathcal{E}$-$\mathcal{W}_2$↓**	**TV↓**
S_K	2.864	**0.010**	**0.812**	1.841	**0.040**	0.092
$\tilde{S}_K$ (ours)	**2.506**	0.124	0.826	**1.835**	0.145	**0.087**
iDEM†	3.98	-	0.81	2.09	-	0.09
iDEM (rerun)	6.406	46.90	0.859	**1.862**	**0.030**	**0.093**
iEFM-VE†	4.31	-	-	2.21	-	-
iEFM-OT†	4.21	-	-	2.07	-	-
TweeDEM (ours)	**3.182**	**1.753**	**0.815**	1.910	0.217	0.120

Table 11 reports x-$\mathcal{W}_2$, $\mathcal{E}-\mathcal{W}_2$, and TV for GMM and DW-4 potentials. Table 11 shows that when using the ground truth estimators for sampling, there's no significant evidence demonstrating the privilege between S_K and $\tilde{S}_K$. However, when training a neural sampler, TweeDEM can significantly outperform iDEM (rerun), iEFM-VE, and iEFM-OT for GMM potential. While for DW4, TweeDEM outperforms iEFM-OT and iEFM-VE in terms of $x - \mathcal{W}_2$ but are not as good as our rerun iDEM.

Figure 10 visualizes the generated samples from ground truth samplers, *i.e.* S_K and $\tilde{S}_K$, and neural samplers, *i.e.* TweeDEM and iDEM. It shows that the ground truth samplers can generate well mode-concentrated samples, as well as TweeDEM, while samples generated by iDEM are not concentrated on the modes and therefore result in the high value of $\mathcal{W}_2$ based metrics. Also, this phenomenon aligns with the one reported by Woo & Ahn (2024), where the iEFM-OT and iEFM-VE can generate samples more concentrated on the modes than iDEM.

Above all, simply replacing the score estimator S_K with $\tilde{S}_K$ can improve generated data quality and outperform iEFM in GMM and DW-4 potentials. Though TweeDEM can outperform the previous state-of-the-art sampler iDEM on GMM, it is still not as capable as iDEM on DW-4. Except scaling up and conducting experiments on larger datasets like LJ-13, combing S_K and $\tilde{S}_K$ is of interest in the future, which balances the system scores and Gaussian ones and can possibly provide more useful and less noisy training signals. In addition, we are considering implementing a denoiser network for TweeDEM as our future work, which might stabilize the training process.