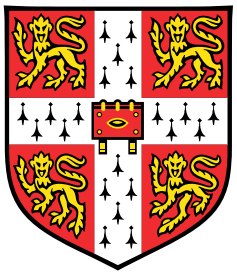
Energy-Based Diffusion Neural Sampler for Boltzmann Densities



RuiKang OuYang

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Wolfson College

October 2024

To stochasticity.

Declaration

I, RuiKang OuYang of Wolfson College, being a candidate for the Master of Philosophy in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

This report jointly uses codes from DEM, which can be found HERE, and our own implementation, implemented by my collaborator and myself. The code produced in this report can be found HERE. Access will be granted by request.

Word Count: 13089

RuiKang OuYang
October 2024

Acknowledgements

I would like to acknowledge my supervisor Prof. José Miguel Hernández-Lobato first and foremost. His passion, insights, and enthusiasm are extremely helpful throughout this project. And I would like to acknowledge my collaborator Bo Qiang, there're many great discussions and brainstorms in these two months and it's great to work with you. Also, thanks to Jiajun He and Dr. Javier Antoran, for the inspiring discussion with them.

Finally, I would love to thank my parents, grandparents, my girlfriend Lois and all my friends. Thank you all for your support and caring. Wish you all the best.

Abstract

Efficiency and sample quality are essential when drawing statistically independent samples from a Boltzmann-type distribution, which is desired in a wide range of scientific problems, such as generating equilibrium samples of many-body systems. Statistical methods like Monte Carlo and MCMC, or actual numerical methods like Molecular Dynamics, are promising but computationally expensive, emerging a trend that leverages the data compression capacity of neural networks for efficient sampling.

In this thesis, we propose ENERGY-BASED DENOISING ENERGY MATCHING (EnDEM) and BOOTSTRAP ENDEM (BEnDEM). The former one, EnDEM, is inspired by the current state-of-the-art Boltzmann neural sampler, DEM, by targeting a less noisy stochastic energy estimator which allows many potentials to further improve performance. While the latter one, BEnDEM, is built on top of EnDEM which improves its learning target by bootstrapping from the learned energy. Both EnDEM and BEnDEM are trained in a bi-level iterated scheme as iDEM, which includes a simulation-free inner loop training an energy-based diffusion sampler and an outer-loop that simulates the learned diffusion sampler to generate more informative samples to further improve the sampler, resulting in scalability to high dimensions. We evaluate EnDEM and BEnDEM on a suit of tasks ranging from synthetic energy functions to invariant n-body particle systems, demonstrating their stronger capacity compared with DEM. We also provide multiple possible ways for further improvement built on top of our models, demonstrating their potential to solve higher dimensional and more complex tasks in the future.

Table of contents

Nomenclature ix

1 Introduction 1
- 1.1 Contribution . . . 2
- 1.2 Outline . . . 3

2 Background and preliminaries 4
- 2.1 Boltzmann Densities . . . 4
- 2.2 Classical Sampling Methods with MCMC . . . 5
  - 2.2.1 Metropolis Hastings . . . 5
  - 2.2.2 Hamiltonian Monte Carlo . . . 6
  - 2.2.3 Metropolis-Adjusted Langevin Algorithm . . . 6
  - 2.2.4 Importance Sampling . . . 7
  - 2.2.5 Annealed Importance Sampling . . . 7
- 2.3 Deep Learning . . . 8
  - 2.3.1 Energy-based Models . . . 8
  - 2.3.2 Score-based Models . . . 9
  - 2.3.3 E(n) Equivariant Graph Neural Networks . . . 10
- 2.4 Diffusion Models . . . 11
  - 2.4.1 Variance Exploding SDEs . . . 13
  - 2.4.2 Train DMs . . . 13
  - 2.4.3 Sample from DMs . . . 14
- 2.5 Related Works . . . 14
  - 2.5.1 Denoising Energy Matching . . . 15
  - 2.5.2 Other neural samplers . . . 18
- 2.6 Summary . . . 18

3 EnDEM: Energy-based iDEM **19**
3.1 Probability Error Bound of MC Score Estimator . . . 20
3.2 MC Energy Estimator . . . 21
3.3 EnDEM . . . 21
3.4 Variance, Bias and Probability Error Bound of MC Energy Estimator . . . 22
3.4.1 Variance, Bias and Concentration Inequality . . . 23
3.4.2 Probability Error Bound . . . 23
3.5 Why energies can be better than scores? . . . 24
3.6 MC Estimators via Annealed Importance Sampling . . . 25
3.7 Incorporating Symmetries . . . 27
3.8 Summary . . . 27

4 BEnDEM: EnDEM with Bootstrap Energy Estimation **28**
4.1 BEnDEM . . . 29
4.1.1 Bootstrap Energy Estimation . . . 29
4.1.2 Geometrical Bootstrap Schedule . . . 30
4.2 Training BEnDEM . . . 31
4.3 Variance-Bias trade-off: a Sequential Estimator perspective . . . 33
4.3.1 Sequential MC Energy Estimator . . . 33
4.3.2 Bootstrap(1) Energy Estimator . . . 34
4.3.3 Bootstrap(n) Energy Estimator . . . 35
4.4 Summary . . . 37

5 Experiments and Results **39**
5.1 Datasets . . . 39
5.2 Evaluation Metrics . . . 40
5.3 Ablation Models . . . 42
5.3.1 DEM-EN: DEM with Energy Network . . . 42
5.3.2 BDEM-EN: DEM-EN with Bootstrap Energy Estimation . . . 42
5.4 Experiment Settings . . . 43
5.5 Variance and Bias of MC estimators: a 2D GMM case study . . . 44
5.6 Main results . . . 46
5.7 Summary . . . 48

6 Conclusions **49**
6.1 Limitation . . . 49
6.2 Future work . . . 50

References 52

Appendix A Supplementary maths 56
A.1 Optimization of neural network . . . 56
A.2 Basic Concentration Inequalities . . . 57
A.2.1 Sub-Gaussian and its concentration . . . 57
A.2.2 Bounded random variable and its concentration . . . 57

Appendix B Proofs in EnDEM and BEnDEM 58
B.1 Probability Error Bound for MC Score Estimator . . . 58
B.1.1 For MC Score Estimator . . . 58
B.2 Properties of MC Energy Estimator . . . 59
B.2.1 logarithm of sub-Gaussian r.v. and its concentration inequality . . . 59
B.2.2 Variance, Bias and Concentration Inequality for MC Energy Estimator 59
B.3 Proof of Proposition 3 . . . 60
B.4 Variance of MC Score Estimator . . . 61
B.4.1 For $\mathbb{R}^1$ data . . . 62
B.4.2 For $\mathbb{R}^d$ data . . . 63
B.4.3 Conclusion . . . 64
B.5 Proof of Proposition 4 . . . 64
B.6 Bias of Bootstrap Energy Estimator . . . 65
B.6.1 Bootstrap(1) estimator . . . 65
B.6.2 Bootstrap(n) estimator . . . 67

Appendix C AIS estimator for Energy and Score 70

Appendix D EnDEM for general SDEs 72

Appendix E Efficient Sampling for EnDEM 75

Appendix F Incorporating Metropolis-Hastings 77

Appendix G Supplementary Experiments 78

Nomenclature

Distributions and Random Variables

$p(x)$	The density of a random variable at x
$p_t(x)$	The density of a time-dependent random variable at x
x	A random variable
$x^{(k)}$	The k^{th} sample drawn from any distribution
x_t	A time-dependent random variable at t

Functions

$\mathcal{E}$	The Energy function
$\mathcal{E}_t$	The time-dependent Energy function at t, $\mathcal{E}_0 = \mathcal{E}$
S	The Score function
S_t	The time-dependent Score function at t, $S_0 = S$

Operators

∇f	The gradient w.r.t the first input of a function f, i.e., $\nabla f = \nabla_x f(x,t)$
$\nabla_\theta f$	The gradient w.r.t parameters θ, where f is parameterized by θ
$\nabla_x \cdot f$	The divergence of a function f w.r.t its first input x, i.e. $\nabla_x \cdot f = \sum_i \frac{\partial f(x,t)}{\partial x_i}$
$\mathbf{tr}$	The trace of a matrix, e.g. $\mathbf{tr}(M) = \sum_i M_{ii}$

Chapter 1

Introduction

Drawing independent samples from a target distribution is a fundamental task in probabilistic modeling. Various applications are built on top of it, such as liquids Allen and Tildesley (2017) and proteins Lazim et al. (2020). In this work, we focus on sampling from an unnormalized density, i.e. Boltzmann-type distribution $\mu_{\text{target}} \propto \exp(-\mathcal{E}(x))$, of many-body system, e.g. molecules, with goals of efficiency and mode-coverage.

In data-driven tasks, where sufficient amount of data is available, Diffusion Models (DMs; Karras et al. (2022); Song et al. (2020b)) achieves state-of-the-art performance in many tasks, such as Image Generation (Karras et al., 2022), Image Super-Resolution (Saharia et al., 2021), Text-to-Image Synthesis (Nichol et al., 2021; Ramesh et al., 2022), Audio Generation (Kong et al., 2020) and Molecular Design (Hoogeboom et al., 2022). And its efficiency, i.e. data generation speed, can be further improved by several techniques like distillation (Luhman and Luhman, 2021; Song et al., 2023).

While in many scientific settings, we are learning to sample from a target distribution μ_{target}, where the size of the dataset (or initial samples) is limited or 0. To draw high-quality samples from the target distribution, one can employ Monte Carlo (MC) methods and Markov Chain Monte Carlo (MCMC) techniques, such as Annealed Importance Sampling (AIS; Neal (2001)), Metropolis-Hastings (MH; Bishop (2006)), Harmonic Monte Carlo (HMC; Betancourt (2018)) and Metropolis-Adjusted Langevin Algorithm (MALA; Roberts and Tweedie (1996)). Alternatively, numerical methods like Molecule Dynamics (MD; Leimkuhler and Matthews (2012)) can be employed. Though these methods are promising to generate high-quality samples from the desired target distribution, they tend to be computationally expensive and inefficient.

On the other hand, the nature of the insufficient amount of data prevents us from training a deep generative model q_θ, e.g. DMs, to approximate the target distribution μ_{target}, which is typically maximizing log-likelihood (i.e. $\max_\theta \mathbb{E}_{\mu_{\text{target}}} \log q_\theta(x)$) or equivalently minimizing

the KL divergence (i.e. $\min_\theta \mathbb{D}_{\mathrm{KL}}(\mu_{\text{target}} \| q_\theta)$). A potential remedy involves learning pseudo data generated from an arbitrary prior, simulating a noising process as DMs and train a neural sampler to match the trajectories, such as Path Integral Sampler (PIS; Zhang and Chen (2022)), Time-reversed Diffusion Sampler (DIS; Berner et al. (2024)) and Denoising Diffusion Sampler (DDS; Vargas et al. (2023)). However, these pure learning methods require simulation during training, which poses challenges for scaling up to higher dimensional tasks.

To solve aforementioned issues of recent samplers for Boltzmann-type distribution, Akhound-Sadegh et al. (2024) proposes Iterated Denoising Energy Matching (iDEM), which is not only computationally tractable but also guarantees good coverage of all modes, leading to state-of-the-art generations outperform previous methods like Flow Annealed Bootstrap (FAB; Midgley et al. (2023)) and those as mentioned above. iDEM proposes a bi-level training scheme to target a novel noise-convolved score estimator which can be estimated by the system energy $\mathcal{E}$ using Monte Carlo methods: the inner-loop is simulation-free and trains a diffusion sampler targeting a novel score estimator with buffer samples; while the outer-loop simulates the trained diffusion sampler to generate more informative data in the buffer. Under this iterated bi-level training strategy, the diffusion sampler can keep drawing more samples covering the modes of target distribution as well as learning corresponding scores, which is similar to an off-policy reinforcement learning algorithm. While iDEM achieves state-of-the-art performance, one can be an issue: when the noise level increases, the variance of score estimators can be large enough to explode, resulting in an invalid training target.

In this thesis, we first propose ENERGY-BASED DENOISING ENERGY MATCHING (EnDEM), which targets the noise-convolved energies instead of scores. Sampling with EnDEM is more expensive than iDEM, while the energy estimator is less noisy than the score estimator and therefore can provide more useful training signals and achieve better performance. Leveraging the learned annealed energies, we propose an improvement of EnDEM via bootstrap energy estimation, BOOTSTRAP ENDEM (BEnDEM), which targets a less noisy estimator at large t estimating from the well-learned energies at smaller noise level. Experiments show that EnDEM can achieve better performance as well as faster convergence-rate compared with DEM, while BEnDEM can further improve the performance and can converge faster than DEM as well.

1.1 Contribution

This thesis is contributed as follows:

- A comprehensive review of methods for sampling from Boltzmann-type distribution, including classical ones like MCMC and most recent ones like DEM.
- We propose EnDEM, a novel energy-based model leveraging denoising diffusions for high-quality samples, demonstrating better performance compared with DEM.
- We propose BEnDEM, an improvement of EnDEM via bootstrapping energy estimation, achieving the best performance.
- Comprehensive theoretical analysis of DEM, EnDEM, and BEnDEM. We provide an error bound of training target of EnDEM and the variance of both energy and score based training targets. We also provide a theoretical understanding of BEnDEM from a Variance-Bias trade-off perspective
- We also provide several future directions for further improving our models, such as generalize EnDEM, efficient EnDEM sampling and EnDEM+MCMC.

1.2 Outline

The thesis is organized as follows:

- In Chapter 2, we introduce related background in methods for sampling from Boltzmann-type distribution, including classical ones and neural samplers, as well as reviews of related works.
- In Chapter 3, we introduce EnDEM, which is inspired by DEM but learning a less noisy target and allows potential ways for further improvement.
- In Chapter 4, we introduce BEnDEM, which is built on top of EnDEM and uses the idea of bootstrapping to reduce the variance of learning targets.
- In Chapter 5, we introduce the datasets, metrics, and settings for our experiments, followed by the results.
- In Chapter 6, we conclude our works as well as discuss its limitations and future directions.

Chapter 2

Background and preliminaries

In this chapter, we first introduce our target, the Boltzmann density, and classical methods for sampling from this distribution like IS, HMC, and MALA. Then we talk about utilizing deep learning techniques for sampling, including EBMs, SBMs, and Diffusion Models. While those topics are extensive, they are highly relevant to our method and could provide more useful insights. At the end of this chapter, we will review related works, especially the most relevant one, DEM.

Before diving into this chapter, we outline the following three questions:

1. *How to sample from a Boltzmann-type distribution?*
2. *How to speed up sampling by combining deep learning while preserving performance?*
3. *How to efficiently sample from a Boltzmann-type distribution with few to* 0 *data?*

2.1 Boltzmann Densities

In statistical mechanisms and mathematics, a Boltzmann density μ_{target} is defined by an unnormalized density function $\mathcal{E}$:

$$\mu_{\text{target}}(x) = \frac{\exp(-\mathcal{E}(x))}{Z} \propto \exp(-\mathcal{E}(x)) \tag{2.1}$$

where $\mathcal{E}(x)$ is also called **energy function**, $Z = \int_{\mathbb{R}^d} \exp(-\mathcal{E}(x))dx$ is the partition function and typically intractable.

Boltzmann density is popular in a lot of scientific problems like many-body systems, e.g. molecules, where its energy could be computed by the spatial relationship of particles and the chemical bonds inside.

In some cases, we also have access to "scores", which are defined as the derivative of log-likelihood w.r.t x:

$$S(x) := \nabla_x \log p(x) = -\nabla_x \mathcal{E}(x) \tag{2.2}$$

which is a vector field pointing to a direction corresponding to higher probability. For example, in a molecular system, score refers to force, determining how atoms will move in response to the energy landscape (Arts et al., 2023; Durumeric et al., 2024; Hsu et al., 2024).

Above all, an interesting and essential problem is:

How to sample from a Boltzmann-type distribution?

2.2 Classical Sampling Methods with MCMC

A very common technique for sampling from a Boltzmann density is using Markov Chain Monte Carlo (MCMC; Bishop 2006). MCMC starts from samples x_0 drawn from initial proposal p_0, and moves them around a sequence $x_0 \to x_1 \to \ldots \to x_n$ by a series of transition kernels $T_k(x,x'), k = 1,\ldots,n$ to reach the target distribution μ_{target}. In this section, we focus on several common MCMC-based sampling methods, which are theoretically guaranteed to converge to target distribution but are computationally expensive.

2.2.1 Metropolis Hastings

A way to construct such a transition kernel is followed by the Metropolis-Hastings (MH; Bishop 2006) algorithm. In particular, at the k^{th} step, in which the current state (sample) is $x^{(k)}$ and given a (conditional) proposal $q_k(\cdot|\cdot)$ at the, MH first draws samples $x' \sim q_k(x'|x^{(k)})$ then accepts it as new state $x^{(k+1)}$ with probability $A_k(x', x^{(k)})$, where:

$$A_k(x,x') = \min\left(1, \frac{\mu_{\text{target}}(x')q_k(x|x')}{\mu_{\text{target}}(x)q_k(x'|x)}\right) \tag{2.3}$$

$$= \min\left(1, \frac{\exp(-\mathcal{E}(x'))q_k(x|x')}{\exp(-\mathcal{E}(x))q_k(x'|x)}\right) \tag{2.4}$$

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

$$\mu_{\text{target}}(x^{(k+1)})q_k(x^{(k)}|x^{(k+1)}) = \mu_{\text{target}}(x^{(k)})q_k(x^{(k+1)}|x^{(k)}) \tag{2.5}$$

Its convergence rate can be very slow and requires a long time to mix with the target distribution.

2.2.2 Hamiltonian Monte Carlo

Though MH is theoretically guaranteed to converge to the target distribution, it suffers from issues like mode collapse in high-dimensional space. Hamiltonian Monte Carlo (HMC; Betancourt 2018), another type of MCMC, solves this issue by combining momentum, which is inspired by Hamiltonian dynamics:

$$H(x,p) = \mathcal{E}(x) + \frac{1}{2}p^T M^{-1} p \tag{2.6}$$

$$\frac{dx_i}{dt} = \frac{\partial H}{\partial p_i}, \frac{dp_i}{dt} = -\frac{\partial H}{\partial x_i} \tag{2.7}$$

where H is the Hamiltonian, and M is the mass matrix which is symmetric and positive definite.

Similar to MH, HMC starts from an initial proposal q_0 and iteratively samples new proposal x' by simulating the Hamiltonian dynamics (2.7) with an integrator for a fixed time T:

$$p(t+\frac{\Delta t}{2}) = p(t) - \frac{\Delta t}{2}\nabla_x \mathcal{E}(x(t)) \tag{2.8}$$

$$x(t+\frac{\Delta t}{2}) = x(t) + \Delta t\, p(t+\frac{\Delta t}{2}) \tag{2.9}$$

$$p(t+\Delta t) = p(t+\frac{\Delta t}{2}) - \frac{\Delta t}{2}\nabla_x \mathcal{E}(x(t+\frac{\Delta t}{2})) \tag{2.10}$$

where $p(0) \sim \mathcal{N}(0,M)$, $x(0) = x^{(k)}$ and $x' = x^{(k+1)}$. Then a new state $x^{(k+1)}$ is generated by accepting x' with probability computed by Equation 2.4.

Even though HMC could be more efficient, as it uses the score, i.e. gradient of energy, its performance is sensitive to the choice of step size T and mass m.

2.2.3 Metropolis-Adjusted Langevin Algorithm

To sample x from a Boltzmann-type target distribution μ_{target}, we could simulate the following Langevin Dynamics:

$$dx_t = -\nabla_x \mathcal{E}(x_t)dt + \sqrt{2}dB_t \tag{2.11}$$

$$= S(x_t)dt + \sqrt{2}dB_t \tag{2.12}$$

where B_t is a suitable Brownian Motion. The marginal distribution of x_t is guaranteed to converge to μ_{target} as $t \to \infty$ (Barrat and Hansen, 2003).

The simple Langevin Dynamics is then improved by the Metropolis-Adjusted Langevin Algorithm (MALA; Roberts and Tweedie 1996), which combines Langevin Dynamics and MCMC. Similar to HMC, MALA starts from initial samples from a proposal q. Then, it iteratively generates a new state $x^{(k+1)}$ by simulating Langevin Dynamics (2.11) for a fixed time T and accepts it with probability computed by Equation 2.4.

2.2.4 Importance Sampling

It is often the case we are interested in the expectation of some observable $f(x)$ with respect to a distribution $x \sim \mu_{\text{target}}$, i.e.

$$\mathbb{E}_{\mu_{\text{target}}}[f(x)] = \int \mu_{\text{target}}(x) f(x) dx \tag{2.13}$$

When μ_{target} is simple and could be sampled directly, Equation 2.13 can be estimated via Monte Carlo. However, when μ_{target} is complex or not easy to sample, we need alternative methods.

Importance Sampling (IS) chooses an easy-to-sample distribution q as proposal and use importance weights, $w(x_i) = \exp(-\mathcal{E}(x_i))/q(x_i)$, to reweigh those samples:

$$\text{IS} := \mathbb{E}_{\mu_{\text{target}}}[f(x)] \approx \frac{\sum_{i=1}^{n} w(x_i) f(x_i)}{\sum_{i=1}^{n} w(x_i)}, x_i \sim q(x) \tag{2.14}$$

$q(x)$ is often chosen to minimize the variance of the IS estimator and roughly proportional to $f(x)\mu_{\text{target}}(x)$ (Owen, 2013) while finding such a proposal is challenging in high dimension space or with a multimodal μ_{target}.

2.2.5 Annealed Importance Sampling

To alleviate the variance issue in IS, Annealed Importance Sampling (AIS; Neal (2001)) is proposed to target a sequence of distributions that are (logarithmly) interpolated between the initial proposal $\pi_0 := q$ and target $p := \mu_{\text{target}}$, where intermediate distribution π_k and π_{k+1} are slightly different. Let L be the length of AIS,

$$\pi_k(x) := q(x)^{\frac{L-k}{L}} p(x)^{\frac{k}{L}}, k = 0, ..., L \tag{2.15}$$

where $\pi_L = \mu_{\text{target}}$. AIS starts from samples drawn from the initial proposal $x^{(0)} \sim \pi_0(x)$, then sequentially targets the intermediate distribution π_k until target distribution $\pi_L = \mu_{\text{target}}$, where the intermediate samples $x^{(k)}$ is produced by running an MCMC (e.g. HMC). Then we could compute the annealed importance weight along the sequence with $x_i = x_i^{(L-1)}$:

$$w_{\text{ais}}(x_i) = \frac{\pi_1(x_i^{(0)})}{\pi_0(x_i^{(0)})} \frac{\pi_2(x_i^{(1)})}{\pi_1(x_i^{(1)})} \cdots \frac{\pi_L(x_i^{(L-1)})}{\pi_{L-1}(x_i^{(L-1)})} \tag{2.16}$$

Plugging $w_{\text{ais}}(x)$ into Equation 2.14 gives us an AIS estimator.

2.3 Deep Learning

Deep Learning (DL) has been popularly studied in the last decade, which solves many computational problems by utilizing the compression capability of neural networks and arising the next industrial evolution. Deep Neural Network (DNN), f_θ is a mapping parameterized by a set of parameters θ:

$$f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y} \tag{2.17}$$

is learning to approximate a "ground-truth" mapping f which can be computationally expensive.

In this section, we will shift to using DL and DNN for probabilistic modeling and one specific DNN architecture for many-body systems, before diving into DL-based sampling methods in the next section.

2.3.1 Energy-based Models

In probabilistic machine learning, we are interested in modeling the distribution of data, $p_\theta(x)$. A common way to optimize parameter θ is by maximizing log-likelihood:

$$\theta^* = \arg\max_\theta \mathbb{E}_{p_{\text{data}}}[\log p_\theta(x)] \tag{2.18}$$

which is equivalent to minimize the KL-divergence $\mathbb{D}_{\text{KL}}(p_{\text{data}} \| p_\theta)$.

One way to parameterize the model is by modeling the energy function $E_\theta(x)$, called Energy-based Models (EBM):

$$p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z_\theta} \tag{2.19}$$

$$Z_\theta = \int \exp(-E_\theta(x))dx \tag{2.20}$$

where Z_θ is an intractable constant w.r.t input x, called partition function. In this way, the gradient of parameter w.r.t the optimization problem (2.18) could be rewritten as:

$$\nabla_\theta \mathbb{E}_{p_{\text{data}}}[\log p_\theta(x)] = \mathbb{E}_{p_{\text{data}}}[-\nabla_\theta E_\theta(x)] - \mathbb{E}_{p_\theta}[-\nabla_\theta E_\theta(x)] \tag{2.21}$$

which resembles minimizing a Contrastive Divergence (Song and Kingma, 2021). And $\mathbb{E}_{p_\theta}$ could be estimated by Monte Carlo, where samples could be drawn by Langevin dynamics or MALA.

2.3.2 Score-based Models

In many cases, like generative models, we are interested in drawing samples from p_θ where we have access to samples from the true data distribution p_{data}. Langevin dynamics (2.11) is efficient to solve this problem. It's noticeable that Langevin dynamics only requires access to score (2.2), we could parameterize it instead of energy, leading to another family of models, Score-based Models (SBM), which is more efficient when simulating the Langevin dynamics as it avoids computing derivative of energy w.r.t x, i.e. $\nabla_x \mathcal{E}(x)$.

Explicit Score Matching (ESM; Hyvärinen 2007) optimizes the score model $s_\theta(x)$ towards the true score $S(x)$ by:

$$\min_\theta \mathcal{L}_{esm}(\theta) := \mathbb{E}_{p_{\text{data}}}[\|S(x) - s_\theta(x)\|^2] \tag{2.22}$$

while in most cases we don't have access to the true scores.

Implicit Score Matching (ISM; Hyvärinen 2007) alternatively minimizes an implicit score matching objective, which is proven to be equivalent to optimizing ESM (Song et al., 2020a):

$$\min_\theta \mathcal{L}_{ism}(\theta) := \mathbb{E}_{p_{\text{data}}}[\nabla_x \cdot s_\theta(x) + \frac{1}{2}\|s_\theta(x)\|^2] \tag{2.23}$$

$$= \mathbb{E}_{p_{\text{data}}}[\mathbf{tr}(\nabla_x s_\theta(x)) + \frac{1}{2}\|s_\theta(x)\|^2] \tag{2.24}$$

$$\mathcal{L}_{esm}(\theta) = \mathcal{L}_{ism}(\theta) + c \tag{2.25}$$

while ISM doesn't require access to true scores, computing $\nabla_x s_\theta(x)$ is computationally expensive.

Sliced Score Matching (SSM; Song et al. 2020a) reduces expensive computation of ISM by an idea of random projection on $1d$ space: If two d-dimensional vector fields (i.e. $s_\theta(x)$ and $S(x)$) are close to each other, then their projection w.r.t. any random projection vector v to a $1D$ space should be close as well. It leads to an SSM objective:

$$\min_\theta \mathcal{L}_{ssm}(\theta) := \min_\theta \frac{1}{2}\mathbb{E}_{p(v)}\mathbb{E}_{p_{\text{data}}(x)}[(v^T\nabla_x \log p(x) - v^T s_\theta(x))^2] \tag{2.26}$$

$$= \min_\theta \mathbb{E}_{p(v)}\mathbb{E}_{p_{\text{data}}(x)}[v^T\nabla_x(s_\theta(x)v) + \frac{1}{2}(v^T s_\theta(x))^2] \tag{2.27}$$

which doesn't require an additional backpropagation to compute $\nabla_x s_\theta(x)$.

Denoising Score Matching (DSM; Song and Ermon 2019) estimates scores of perturbed data $\tilde{x} \sim p_\sigma(\tilde{x})$ rather than estimating score of original data distribution like SSM, where

$$p_\sigma(\tilde{x}) = \int p_{\text{data}}(x) p_\sigma(\tilde{x}|x) dx \tag{2.28}$$

and $q_\sigma(\cdot|\cdot)$ a perturbation kernel with variance σ^2. Then a score network $s_\theta(x)$ is learned to target this noisy score, which is roughly approximating the original clean score when σ is small enough. Thus, we instead optimize the following DSM objective:

$$\min_\theta \mathcal{L}_{\text{dsm}}(\theta) := \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x})}[\|s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log p_\sigma(\tilde{x})\|^2] \tag{2.29}$$

$$= \frac{1}{2}\mathbb{E}_{p_\sigma(\tilde{x}|x)p_{\text{data}}(x)}[\|s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log p_\sigma(\tilde{x}|x)\|^2] \tag{2.30}$$

Equation 2.30 can be further simplified with a Gaussian perturbation kernel, $p_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}; x\sigma^2 I)$:

$$\mathcal{L}_{\text{dsm}}(\theta) = \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x}|x)p_{\text{data}}(x)}[\|s_\theta(\tilde{x}) + \frac{\tilde{x}-x}{\sigma^2}\|^2] \tag{2.31}$$

DSM provides an efficient way to train an SBM and is highly related to the state-of-the-art generative models - Diffusion Models (DMs). We'll further discuss DSM and DMs in the next section.

2.3.3 E(n) Equivariant Graph Neural Networks

Before diving into DMs, we briefly introduce a neural network architecture that is essential in modeling many-body system - $E(n)$ Equivariant Graph Neural Networks (EGNN; Satorras

et al. 2022). In general, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $v_i \in \mathcal{V}$ are nodes and $e_{ij} \in \mathcal{E}$ are edges, a GNN architecture can be stacked by multiple Graph Convolution Layer (GCL; Kipf and Welling (2016)), which injects node embeddings from last layer and outputs embeddings for next layer, i.e. $h^{l+1} = GCL(h^l; \mathcal{G})$. A GCL is defined as

$$m_{ij} = \phi_e(h_i^l, h_j^l, a_{ij}) \tag{2.32}$$

$$m_i = \sum_{j \in \mathcal{N}(i)} m_{ij} \tag{2.33}$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i) \tag{2.34}$$

where $h_i^l \in \mathbb{R}^d$ is the d-dimensional embedding of node v_i at layer l, a_{ij} are the edge attributes and $\mathcal{N}(i)$ represents the set of neighbors of node v_i. Typically, ϕ_e and ϕ_h are the edge and node operations respectively, and are commonly parameterized by an MLP. However, the GCL is not $E(n)$-equivariant, i.e. for any $g \in E(n)$, $GCL(g \circ h^l) \neq g \circ GCL(h^l)$, and therefore not suitable for many-body system.

Instead, Satorras et al. (2022) proposes a Equivariant Graph Convolution Layer (EGCL) with $h^{l+1}, x^{l+1} = EGCL(h^l, x^l)$, which is defined as

$$m_{ij} = \phi_e(h_i^l, h_j^l, \|x_i^l - x_j^l\|^2, a_{ij}) \tag{2.35}$$

$$x^{l+1} = x^l + \frac{1}{|\mathcal{V}| - 1} \sum_{i \neq j} (x_i^l - x_j^l) \phi_x(m_{ij}) \tag{2.36}$$

$$m_i = \sum_{j \neq i} m_{ij} \tag{2.37}$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i) \tag{2.38}$$

where x^l are the coordinate embeddings and $x^0 = x \in \mathbb{R}^n$ are the coordinates in n-dimensional space, and ϕ_x is also parameterized by a MLP. The EGCL is shown to be $E(n)$-equivariant, i.e. for any $g \in E(n)$

$$(EGCL \circ g)(h^l, x^l) = (g \circ EGCL)(h^l, x^l) \tag{2.39}$$

2.4 Diffusion Models

In the last two sections, we introduce classical methods for sampling from Boltzmann-type distribution, which is computationally expensive. Also, we introduce deep learning for probabilistic modeling - EBMs and SBMs. In this section, we are going to introduce

Diffusion Models (DMs; Song et al. 2020b), a family of state-of-the-art generative models, which combines deep learning to solve our second question:

How to speed up sampling by combining deep learning while preserving performance?

Diffusion Models (DMs) consists of two processes (Song et al., 2020b): 1. A forward SDE (diffusion process) noises a complex, high-dimensional and multimodal target distribution $\mu_{\text{target}} = p_0$ into an easy-to-sample unimodal distribution (or prior); 2. A tractable reverse SDE (denoising process) starts from that easy-to-sample prior and iteratively recovers a sample from the target distribution.
Forward SDE can be generally defined by an Ito-SDE, which is well known as diffusion process (Song et al., 2020b):

$$dx = f(x,t)dt + g(t)d\mathbf{w} \tag{2.40}$$

where $f(x,t)$ is called drift coefficient, $g(t)$ the diffusion coefficient and $\mathbf{w}$ a Wiener process (Brownian Motion). For simplification, we suppose $t \in [0,1]$.
Reverse SDE of the above diffusion process is also a diffusion process (Anderson, 1982):

$$dx = [f(x,t) - g^2(t)\nabla_x \log p_t(x)]dt + g(t)d\tilde{\mathbf{w}} \tag{2.41}$$

where $p_t(x)$ is the marginal distribution at t, dt an infinitesimal negative timestep and $\tilde{\mathbf{w}}$ a standard Wiener process when time flows backwards from $t = 1$ to 0.
Generally, DMs are parameterizing a time-dependent score network $s_\theta(x,t)$, to target the score of marginal distribution:

$$S_t(x) := \nabla_x \log p_t(x) \tag{2.42}$$

which could be plugged into Equation 2.41 to generate samples, i.e.:

$$dx = [f(x,t) - g^2(t)s_\theta(x_t,t)]dt + g(t)d\tilde{\mathbf{w}} \tag{2.43}$$

In the rest of this section, we will briefly introduce training and sampling of DMs under a variance-exploding SDE setting for simplification. Discussion about more general forward SDEs setting can be found in Appendix D.

2.4.1 Variance Exploding SDEs

Variance Exploding (VE) SDEs define an Ito-SDE with $f(x,t) \equiv 0$ and $g(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}$:

$$dx = \sqrt{\frac{d\sigma^2(t)}{dt}} d\mathbf{w} \tag{2.44}$$

where $\sigma_t := \sigma(t)$ defines a noise schedule.

Then the conditional distribution of $x_t|x_s$ is given by Song et al. (2020b):

$$p(x_t|x_s,s,t) = \mathcal{N}(x_t; x_s, (\sigma_t^2 - \sigma_s^2)I) \tag{2.45}$$

In this case, given a clean data at $t = 0$, i.e. x_0, x_t is perturbed by a Gaussian noise with 0 mean and $\sigma_t^2 I$ covariance, leading to following perturbation kernel:

$$p_{\sigma_t}(x_t|x_0) = \mathcal{N}(x_t; x_0, \sigma_t^2 I) \tag{2.46}$$

From henceforth, we denote p_{σ_t} as p_t for simplification. Then, given a noise schedule $\sigma(t)$ and data distribution p_{data}, the marginal distribution p_t can be expressed by substituting the perturbation kernel in Equation 2.28:

$$p_t(x_t) = \int p_{\text{data}}(x)\mathcal{N}(x_t; x, \sigma_t^2 I)dx \tag{2.47}$$

2.4.2 Train DMs

DMs are trained by optimising a DSM loss (2.31) that generalized to a set of time-dependent perturbation kernels q_{σ_t}, leading to the following Diffusion Models loss, $\mathcal{L}_{\text{DM}}$:

$$\mathcal{L}_{\text{DM}}(\theta) := \mathbb{E}_{t\in U[0,1]}\mathbb{E}_{p_{\text{daata}}(x_0)p_t(x_t|x_0)}\left[\left\| s_\theta(x_t,t) - \frac{x_0 - x_t}{\sigma_t^2}\right\|^2\right] \tag{2.48}$$

$$= \mathbb{E}_{t,x_0,x_t}\left[\frac{1}{\sigma_t^4}\left\|x_t + \sigma_t^2 s_\theta(x_t,t) - x_0\right\|^2\right] \tag{2.49}$$

By Tweedie's formula (Efron, 2011),

$$\nabla_x \log p_t(x) = \mathbb{E}_{p(x_0|x_t)}\left[\frac{x_0 - x_t}{\sigma_t^2}\right] \tag{2.50}$$

where $p(x_0|x_t) \propto p_{\text{data}}(x_0)p_t(x_t|x_0)$ by Bayes rule.

Therefore, the learning objective (2.49) resembles training a denoiser $D_\theta(x,t) := x + \sigma_t^2 s_\theta(x,t)$ to minimize the expected L_2 denoising error, where we can also design any positive weighting function $\lambda(t)$ to reweigh loss at t:

$$\mathcal{L}_{\mathrm{DM}}(\theta;\lambda) = \mathbb{E}_{t,x_0,x_t}\left[\lambda(t)\left\|D_\theta(x_t,t) - x_0\right\|^2\right] \tag{2.51}$$

The parameterization of denoiser D_θ is generalized by Karras et al. (2022), which achieves state-of-the-art performance in image generation.

2.4.3 Sample from DMs

Once the score network s_θ is well trained, we can plug it into Equation 2.41 and have:

$$dx = g^2(t)s_\theta(x,t)dt + g(t)d\tilde{\mathbf{w}} \tag{2.52}$$

which can be solved by any SDE solversSong et al. (2020b). For example, we can discretize Equation 2.52 as:

$$x_{t-1} = x_t + \sigma_t^2 s_\theta(x_t,t) + \sigma_t \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0,I) \tag{2.53}$$

which resembles 1-step Langevin dynamics from marginal p_t to p_{t-1}.

For deterministic sampling, Song et al. (2020b) shows that given a reverse SDE (2.41), we can find a probability flow ODE that shares the same marginal distribution $p_t(\mathbf{x})$:

$$dx = \{f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)\}dt \tag{2.54}$$

$$= \{f(x,t) - \frac{1}{2}g(t)^2 s_\theta(x,t)\}dt \tag{2.55}$$

We can solve this probability flow ODE by any ODE solver, like 1^{st} order Euler (Karras et al., 2022; Song et al., 2020b) or 2^{nd} order Heun (Karras et al., 2022).

2.5 Related Works

In previous sections, we briefly introduce several methods answering the first two questions: *How to sample from a Boltzmann-type distribution?* and *How to speed up sampling by combing deep learning while preserving performance?*

In this section, we focus on the last question:

How to efficiently sample from a Boltzmann-type distribution with few to 0 data?

We'll first detailly introduce Iterated Denoising Energy Matching (iDEM; Akhound-Sadegh et al. 2024), which is the foundation of our method. Then we'll briefly introduce several relevant methods like FAB, DDS, and DIS.

2.5.1 Denoising Energy Matching

We now present the foundation model of our methods - Iterated Denoising Energy Matching (iDEM; Akhound-Sadegh et al. 2024). From henceforth, we interchangeably use μ_{target} and p_0 to refer to the target density at time $t = 0$ and set p_1 to denote a tractable prior to time $t = 1$.

For a diffusion process defined by a VE-SDE with known noise schedule $\sigma(t)$, we first review its marginal distribution at time t, i.e. Equation 2.47. We are going to introduce usages of Equation 2.47 in two different settings:

Learning from a large dataset & Learning to sample

Learning from a large dataset. When we have access to a large scale dataset, p_{data} can be represented by a mixture of Dirac measures over the training dataset, denoted as p_0. Then the marginal distribution $p_t(x_t)$ can be estimated by an MC estimator, while the corresponding score can be estimated by an MC estimator as well (by Tweedie's formula Efron (2011)):

$$p_t(x_t) = \mathbb{E}_{p_0}[\mathcal{N}(x_t; x, \sigma_t^2 I)] \tag{2.56}$$

$$S_t(x_t) = \nabla \log \mathbb{E}_{p_0}[\mathcal{N}(x_t; x, \sigma_t^2 I)] = \mathbb{E}_{p_0}[-(x_t - x)/\sigma_t^2] \tag{2.57}$$

resulting to a simple regression loss described as 2.51, by taking $\mathbb{E}_{p_0}$ out in $\mathbb{E}_{p_t}\|s_\theta(x_t, t) - S_t(x_t)\|^2$ when targeting scores at t. Therefore, the DSM loss is simply computed by an MC estimator over samples from the dataset.

Learning to sample. Now only an unnormalized density, i.e. $p_0(x) = \frac{\exp(-\mathcal{E}(x))}{Z_0}$, is available where the partition function Z_0 is a constant w.r.t input x, with few to 0 known samples. Equation 2.47 can be rewritten as follow:

$$p_t(x_t) = \int \frac{\exp(-\mathcal{E}(x))}{Z_0} \mathcal{N}(x_t; x, \sigma_t^2 I) dx \tag{2.58}$$

It's noticeable that $\mathcal{N}(x_t; x, \sigma_t^2 I)$ can be rewritten as $\mathcal{N}(x; x_t, \sigma_t^2 I)$ according to symmetry of Gaussian. Therefore, the marginal distribution p_t and corresponding score S_t can be

expressed as follow:

$$p_t(x_t) = \frac{1}{Z_0}\mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[\exp(-\mathcal{E}(x))] \tag{2.59}$$

$$S_t(x_t) = \nabla \log p_t(x_t) = \nabla_{x_t} \log \mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[\exp(-\mathcal{E}(x))] \tag{2.60}$$

MC score estimator. The above expressions suggest we can use an MC estimator to estimate $S_t(x_t)$, by first drawing samples around x_t (i.e. $x_{0|t}^{(i)} \sim \mathcal{N}(x;x_t,\sigma_t^2 I)$) then evaluating their energies and finally operated with $\nabla \log \sum$:

$$S_t(x_t) \approx S_K(x_t,t) := \nabla_{x_t} \log \sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)})) \tag{2.61}$$

To improve numerical stability, the score MC estimator $S_K(x_t,t)$ is implemented by the gradient of a LogSumExp operator over the negative energies.

Learning objective of DEM is therefore a simple regression loss defined as:

$$\mathcal{L}_{\text{DEM}}(x_t,t) := \|s_\theta(x_t,t) - S_K(x_t,t)\|^2 \tag{2.62}$$

Notice that we could evaluate the loss given any pair of (x_t,t), while the DSM loss (2.51) requires an expectation over the dataset to recover the marginal score given by Tweedie's formula (2.50). This flexibility allows direct score matching without any knowledge of true data distribution p_0, leading to the following DEM loss:

$$\mathcal{L}_{\text{DEM}}(\theta;\lambda) := \mathbb{E}_{t,x_0,x_t}[\lambda(t)\|s_\theta(x_t,t) - S_K(x_t,t)\|^2] \tag{2.63}$$

where x_0 can be sampled from any distribution q rather than from the true data distribution.

Train DEM. Ideally, if x_0 is sampled from a distribution q whose support covers the one of true data distribution, we can keep sampling data from q and evaluate Equation 2.63 to optimize the model; then eventually, the model learns scores for any data point in the sampling space. However, it's obviously far from efficient. Akhound-Sadegh et al. (2024) proposes a bi-level iterative scheme for training DEM similar to Midgley et al. (2023), where the score network is updated by Equation 2.63 with a buffer in an inner-loop; while the buffer is updated in an outer-loop .

> **Inner Loop.** The neural sampler (score network) s_θ is trained to approximate the score at different noise levels, i.e. $s_\theta(\cdot,t)$ for different t, for each data point in the buffer. In particular, s_θ is trained by DEM loss (2.63), which is efficient as its simulation-free nature.

Outer Loop. To train the neural sampler s_θ better, it is crucial to find informative points, i.e. those with high probabilities (or low energies, equivalently). It is possible to use any off-policy methods and MCMC or even MD to generate informative points. But those methods are typically inefficient. Instead, Akhound-Sadegh et al. (2024) uses the neural sampler s_θ itself to generate informative samples and use a buffer to utilize previously generated samples. The proposal of s_θ can help fast exploration of the data region, while the inner-loop provides informative direction for exploration in the next iteration by optimizing Equation 2.63. Iteratively, s_θ can generate more and more informative points and gradually approximate $S_K(\cdot,t)$ perfectly.

A complete description of iDEM is given in Algorithm 1.

Algorithm 1 Iterated Denoising Energy Matching

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

while Outer-Loop do **do**
  $\{x_1\}_{i=1}^b \sim p_1(x_1)$
  $\{x_0\}_{i=1}^b \leftarrow \text{sde.int}(\{x_1\}_{i=1}^b, s_\theta, L)$ ▷ Sample
  $\mathcal{B} = (\mathcal{B} \cup \{x_0\}_{i=1}^b)$ ▷ Update Buffer $\mathcal{B}$
  while Inner-Loop do **do**
    $x_0 \leftarrow \mathcal{B}.\text{sample}()$ ▷ Uniform sampling from $\mathcal{B}$
    $t \sim \mathcal{U}(0,1), x_t \sim \mathcal{N}(x_0, \sigma_t^2)$
    $\mathcal{L}_{\text{DEM}}(x_t,t) = \|S_K(x_t,t) - s_\theta(x_t,t)\|^2$
    $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{DEM}})$
  end while
end while

Ensure: s_θ

Incoorperating Symmetries In many physical problems, the Boltzmann-type distribution is preserving symmetries of the physical system. For example, if we consider a $n-$body system in $\mathbb{R}^d$, where $d = 3n$, the symmetries correspond to the rotation, translation, and permutation of the particles, resulting in the target density μ_{target} (or the energy function $\mathcal{E}$) is invariant to the product group $G = \text{SE}(3) \times \mathbb{S}_n$. Given an G-invariant energy function $\mathcal{E}$, its derivative $\nabla\mathcal{E}$ is G-equivariant and Akhound-Sadegh et al. (2024) proves that the MC score estimator S_K is G-equivariant under certain condition:

Proposition 1 *(Akhound-Sadegh et al., 2024) 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 score estimator of $S_K(x_t,t)$ is G-equivariant if the sampling distribution $x_{0|t} \sim \mathcal{N}(x_{0|t}; x_t, \sigma_t^2)$ is G-invariant, i.e.,*

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

In practice, the S_K can be equivariant by replacing the standard normal distribution with a normal distribution that has zero center of mass.

2.5.2 Other neural samplers

Flow Annealed Importance Sampling Boostrap (FAB, Midgley et al. 2023) uses samples from AIS to train a discrete normalizing flow using a $\alpha = 2$ divergence, which is MCMC-based. In other words, FAB is learning from multiple MCMC trajectories, given by AIS. Also, FAB is trained by a bi-level iterative scheme as well, where an outer-loop collects samples and trajectories of AIS; while an inner-loop uses those samples, trajectories, and their AIS weights to train a model p_θ and updates the buffer as well.
Path Integral Sampler (PIS; Zhang and Chen 2022), **Denoising Diffusion Sampler** (DDS; Vargas et al. 2023) and **time-reversed Diffusion Sampler** (DIS; Berner et al. 2024) are using SDE trajectories to train neural SDE (Tzen and Raginsky, 2019) to sample from the target distribution. These samplers share a similar idea during training: minimize the KL-divergence over forward (noising process) trajectories and the backward (denoising process) ones. The forward trajectories are given by simulating an SDE from the target distribution; while we can start from a tractable prior and simulate a trainable SDE to generate backward trajectories. As their nature of learning from trajectories, these methods are simulation-based and not "off-policy", resulting in expensive training.

2.6 Summary

In this chapter, we sequentially solved three questions around "sampling from Boltzmann-type distribution": *How to sample from a Boltzmann-type distribution?*, *How to speed up sampling by combing deep learning while preserving performance?* and *How to efficiently sample from a Boltzmann-type distribution with few to 0 data?* by reviewing classical sampling methods, e.g. MCMC (Bishop, 2006), and current neural sampler, e.g. FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024). It's noticeable that iDEM (Akhound-Sadegh et al., 2024) achieves state-of-the-art performance. However, it still has several issues behind it, resulting in difficulty when scaling up to more complex problems like high-dimensional many-body systems. In the next chapter, we'll introduce our model, which shares a similar spirit as iDEM (Akhound-Sadegh et al., 2024) while achieving better performance.

Chapter 3

EnDEM: Energy-based iDEM

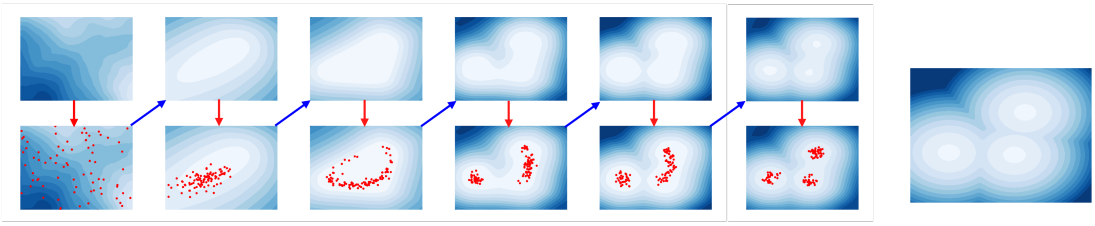
In the last chapter, we introduce the goal of this thesis: *Efficient sampling from Boltzmann-type distribution with few to 0 samples*, as well as several representative methods. We also introduce state-of-the-art method, iDEM (Akhound-Sadegh et al., 2024), which outperforms previous methods like FAB (Midgley et al., 2023), PIS (Zhang and Chen, 2022), DIS (Betancourt, 2018) and DDS (Vargas et al., 2023).

iDEM (Akhound-Sadegh et al., 2024) uses a novel score estimator as a target, to train a score-based diffusion models. However, it suffers from two main issues: high bias and variance of training targets.

First notice that, given by Equation 2.61, the MC score estimator is biased, where $\log \sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))$ is noisy and biased while the gradient operator ∇ can amplify those noisiness and bias. The noisiness of the training target results in less reliable training signals and difficulty in optimization; while bias can be misleading, and makes the model learn wrong scores.

To solve these issues, we propose a variant of iDEM based on energy diffusion called Energy-based iDEM (EnDEM), which targets the noise-convolved energies, $\mathcal{E}_t$, instead of scores S_t. An illustration of Iterated training of EnDEM is provided in Figure 3.1: similar to iDEM, EnDEM starts from random initialized time-convolved energies and simulates the amortised diffusion sampler to generate samples; it iteratievly uses those samples to match the time-convolved energies and generates new samples for training in next iteration.

In this chapter, we will first introduce the issues of iDEM and review the theoretical results provided by Akhound-Sadegh et al. (2024). Then, we'll introduce our methods, by going through theoretical results first, including error bounds and variance of our target estimator and comparison with iDEM. It is proven that our method is training with a more reliable target in terms of its bias and variance. After that, we'll present experimental results



(a) Iterated training (b) Ground truth

Fig. 3.1 Illustration of sampling from Boltzmann-type distribution using EnDEM. (a) Contour lines for predicted target energy at $t = 0$, i.e. $E_\theta(x,0)$; Scatters for buffer samples generated from a diffusion sampler $s_\theta(x_t,t) = \nabla_{x_t} E_\theta(x_t,t)$; $\downarrow$ for sampling from the diffusion sampler and $\nearrow$ for updating E_θ using buffer samples. (b) Contour lines for ground truth energy $\mathcal{E}(x)$.

on several datasets, which show that our method outperforms the current state-of-the-art model iDEM.

3.1 Probability Error Bound of MC Score Estimator

The MC score estimator is computed by Equation 2.61. We review it here:

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

Notice that the part inside the logarithm, $Z_K(x_t,t) := \frac{1}{K}\sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)}))$, is a MC estimator which is unbiased; while logarithm of an unbiased estimator results a biased estimator. Therefore, $\log Z_K(x_t,t)$ and $\nabla \log Z_K(x_t,t)$ are biased. Akhound-Sadegh et al. (2024) analyzes the bias of MC score estimator (2.61) by assuming sub-Gaussianess of $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ and $\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))$.

Proposition 2 *(Akhound-Sadegh et al., 2024) If* $\exp(-\mathcal{E}(x_0^{(i)}|t))$ *and* $\|\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))\|$ *are sub-Gaussian, then, there exists a constant* $c(x_t)$ *such that with probability* $1-\delta$ *(over* $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2)$*) we have*

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

Akhound-Sadegh et al. (2024) finds that even if the MC score estimator S_K (2.61) is biased, the neural sampler is still able to generate samples with high mode-coverage; while Proposition

2 shows a $O(1/\sqrt{K})$ decay rate of the bias of S_K. However, when the target distribution is high-dimensional and complex, the constant $c(x_t)$ in Proposition 2 can be large and such that S_K can be problematic.

3.2 MC Energy Estimator

Instead of learning scores, we propose an MC energy estimator and target this estimator with an energy network $E_\theta(x_t,t)$, resulting in learning a sequence of annealed energy landscapes with an energy diffusion.

Recap that the marginal distribution (2.59) can be expressed by:

$$p_t(x_t) = \frac{1}{Z_0}\exp(-\log \mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[\exp(-\mathcal{E}(x))]) \tag{3.1}$$

Let's defined an annealed energy at time t, i.e. $\mathcal{E}_t$:

$$\mathcal{E}_t(x) := -\log \mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[\exp(-\mathcal{E}(x))] \propto p_t(x_t) \tag{3.2}$$

where $\int \mathcal{E}_t(x)dx = Z_0$. In this setting, the annealed (or noise-convolved) energies share the same partition function Z_0.

Equation 3.2 suggests a Monte Carlo estimator similar to score's to approximate the annealed energy function:

$$E_K(x_t,t) := -\log \frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)})), \quad x_{0|t}^{(i)} \sim \mathcal{N}(x;x_t,\sigma_t^2 I) \tag{3.3}$$

$$= -\log \sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)})) + \log K \tag{3.4}$$

where we can similarly use the LogSumExp trick in the implementation and the MC energy estimator E_K and MC score estimator S_K follow the definition between energies and scores:

$$S_K(x_t,t) = -\nabla E_K(x_t,t) \tag{3.5}$$

3.3 EnDEM

To target the MC energy estimator $E_K(x,t)$, we propose the Energy-based iDEM (EnDEM), which models the annealed energy functions $E_\theta(x,t)$ and simulates an energy diffusion for sampling.

Training with a regression loss. In this case, we optimize $E_\theta(x_t,t)$ by minimizing a regression loss w.r.t the MC energy estimator:

$$\mathcal{L}_{\text{EnDEM}}(x_t,t) := \|E_\theta(x_t,t) - E_K(x_t,t)\|^2 \tag{3.6}$$

$$\mathcal{L}_{\text{EnDEM}}(\theta;\lambda) := \mathbb{E}_{t,x_0,x_t}[\lambda(t)\|E_\theta(x_t,t) - E_K(x_t,t)\|^2] \tag{3.7}$$

Sampling with an energy diffusion. When generating samples from the learned energies with a diffusion sampler, we can substitute s_θ by $-\nabla E_\theta(x_t,t)$ in Equation 2.43:

$$dx = [f(x,t) + g^2(t)\nabla E_\theta(x,t)]dt + g(t)d\tilde{\mathbf{w}} \tag{3.8}$$

where $f(x,t) = 0$ and $g(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}$ in our VE-SDE setting.

A complete description of EnDEM is provided in Algorithm 2.

Algorithm 2 Energy-based Iterated Denoising Energy Matching

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

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

Ensure: s_θ

3.4 Variance, Bias and Probability Error Bound of MC Energy Estimator

To analyze the variance and bias of our MC energy estimator, we follow the same assumption of sub-Gaussianess of $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$. Let $Z_K(x_t,t) := \frac{1}{K}\sum_{i=1}^K z_{0t}^{(i)}(x_t)$, where $z_{0t}(x_t) = \exp(-\mathcal{E}(x))$ and x is a random variable with density $N(x;x_t,\sigma_t^2 I)$. Then $z_{0t}(x_t)$

is sub-Gaussian under our assumption. let

$$m_{0t}(x_t) := \mathbb{E}[z_{0t}(x_t)] = \mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[\exp(-\mathcal{E}(x))] \tag{3.9}$$

$$v_{0t}(x_t) := \mathrm{Var}(z_{0t}(x_t)) = \mathrm{Var}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}(\exp(-\mathcal{E}(x))) \tag{3.10}$$

Then the unbiased estimator $Z_K(x_t,t)$ is also sub-Gaussian with mean m and variance $v_{0t}(x_t)/K$. Under this setting, $m_{0t}(x_t) = \exp(-\mathcal{E}_t(x_t))$. By combining a concentration inequality of $Z_K(x_t,t)$ and adapting our notations, the concentration inequality proposed in 2 can be further expressed by:

$$\|\mathcal{S}_K(x_t,t) - S_t(x_t)\| \leq \frac{2\sqrt{2v_{0t}(x_t)\log(\frac{2}{\delta})}(1+\|\nabla\mathcal{E}_t(x_t)\|)}{m_{0t}(x_t)\sqrt{K}} \tag{3.11}$$

with probability $1-\delta$ (Appendix A.2.1).

3.4.1 Variance, Bias and Concentration Inequality

Now, the MC energy estimator E_K is given by the logarithm of an unbiased estimator Z_K, i.e. $E_K := -\log Z_K$. Applying a 2^{nd} order Taylor expansion and using the sub-Gaussianess of Z_K, we show that (see Appendix B.2):

$$\mathbb{E}[E_K(x_t,t)] = \mathbb{E}\log Z_K(x_t,t) \approx -\log m_{0t}(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K} \tag{3.12}$$

$$\mathrm{Var}(E_K(x_t,t)) = \mathrm{Var}(\log Z_K(x_t,t)) \approx \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K} \tag{3.13}$$

and with probability $1-\delta$:

$$|\log Z_K - \mathbb{E}\log Z_K(x_t,t)| \leq \frac{\sqrt{2v_{0t}(x_t)\log\frac{2}{\delta}}}{m_{0t}(x_t)\sqrt{K}} \tag{3.14}$$

Also notice that, Equation 3.12 shows that the bias of E_K w.r.t its expectation is roughly $\frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K}$. While its variance is roughly twice its bias and decays at a rate of $O(1/K)$.

3.4.2 Probability Error Bound

We derive the probability error bound for our energy estimator, i.e. $|E_K(x_t,t) - \mathcal{E}_t(x_t)|$, by leveraging the triangle inequality and the mean and variance of $\log Z_K$, in the following proposition:

Proposition 3 *(ours) If* $\exp(-\mathcal{E}(x_0^{(i)}|t))$ *is sub-Gaussian, then with probability* $1-\delta$ *(over* $x_{0|t}^{(i)} \sim \mathcal{N}(x_t, \sigma_t^2)$*) we have*

$$\|E_K(x_t,t) - \mathcal{E}_t(x_t)\| \leq \frac{\sqrt{2v_{0t}(x_t)\log\frac{2}{\delta}}}{m_{0t}(x_t)\sqrt{K}} \tag{3.15}$$

A complete proof is given in Appendix B.3.

Equation 3.15 and Equation 3.11 show that the error bound of E_K has a smaller constant term than the one of S_K, meaning that our energy target can be less biased, especially on regions with steep gradient (i.e. large $\|\nabla\exp(-\mathcal{E}(x))\|$).

3.5 Why energies can be better than scores?

A question can potentially be asked: *Even if you are targeting a less biased estimator, you need to differentiate it when calculating the scores, and the bias can be amplified again. Then why don't you target scores directly?* We are now going to ask this question from two perspectives: variance and bias.

Variance. To compare the variance of the MC energy estimator (3.3) and the score one (2.61), we derive the (approximated) variance of the MC score estimator in Appendix B.4. We show that the sum of variances of the elements of S_K is larger than the variance of E_K. It means that the MC energy estimator E_K can provide more useful and less noisy training signal than S_K, and also can achieve smaller loss during training [1]. Especially, in low-energy regions we derive that

$$\sum_{j=1}^{d} \mathrm{Var}(S_K(x_t,t)[j]) \approx \frac{4v_{0t}(x_t)(1+\|\nabla\mathcal{E}_t(x_t)\|)^2}{m_{0t}^2(x_t)K} \tag{3.16}$$

Therefore, the variance of E_K (3.13) can be much smaller in high-energy regions. This is desirable as most regions correspond to high energy in complex and high-dimensional tasks, showing the fundamental advantage of targeting E_K instead of S_K.

Bias. On the other hand, the small bias of E_K results in it being a more reliable target compared with S_K. Even if we need to differentiate E_K when sampling, which can amplify its bias back to the same as S_K's, a reliable E_K can provide us with more information about the

[1] Suppose you are training a neural network $f_\theta(x)$ to target noisy observations $y(x) = f(x) + \varepsilon(x)$ using a l_2 loss, where $x \in \mathbb{R}^d$ and $\varepsilon(x)$ is a random function with 0 mean. Then $f_{\theta^*} = f$ is a minimizer; while the minimum of loss equals to $\sum_{i=1}^d \sigma_i^2 := \sum_{i=1}^d \mathbb{E}_x[\mathrm{Var}(\varepsilon_i(x))]$. See proof in Appendix A.1.

target distribution and its energy landscape. The well-learned annealed energies also allow us to combine statistical techniques like importance sampling to improve sample quality. Also in the next chapter, we'll introduce an improvement of EnDEM based on the learned anneal energies.

3.6 MC Estimators via Annealed Importance Sampling

A simple way to improve the MC estimators, both Energy and Score ones, is using annealed importance sampling (AIS; Section 2.2.5).

MC Energy Estimator as an importance-weighted estimate. First notice that the energy function $\mathcal{E}_t(x_t)$ is actually $Z_0 \log p_t(x_t)$, where $p_t(x_t)$ is derived by Equation 2.58. Then $\mathcal{E}_t(x_t)$ can be expressed as an expectation over a proposal $q(x|x_t)$:

$$\mathcal{E}_t(x_t) = -\log \int \exp(-\mathcal{E}(x))\mathcal{N}(x_t; x, \sigma_t^2 I)dx \tag{3.17}$$

$$= -\log \int q(x|x_t)\frac{\exp(-\mathcal{E}(x))\mathcal{N}(x_t; x, \sigma_t^2 I)}{q(x|x_t)}dx \tag{3.18}$$

$$= -\log \mathbb{E}_{q(x|x_t)}\left[\frac{\exp(-\mathcal{E}(x))\mathcal{N}(x_t; x, \sigma_t^2 I)}{q(x|x_t)}\right] \tag{3.19}$$

When using a proposal symmetric to the perturbation kernel $\mathcal{N}(x_t; x, \sigma_t^2 I)$, i.e. $q(x|x_t) = \mathcal{N}(x; x_t, \sigma_t^2 I)$, then Equation 3.19 can be simplified as:

$$\mathcal{E}_t(x_t) = -\log \mathbb{E}_{\mathcal{N}(x; x_t, \sigma_t^2 I)}[\exp(-\mathcal{E}(x))] \tag{3.20}$$

which suggests the MC energy estimator we proposed in Equation 3.3, and its derivative recovers the MC score estimator (2.61) in Akhound-Sadegh et al. (2024).

With a general proposal $q(x|x_t)$ and samples $x_0^{(i)}$ drawn from this proposal, we can establish an importance-weighted estimation for $\mathcal{E}_t(x_t)$:

$$E_K^{\mathrm{IS}}(x_t, t) := -\log \frac{\sum_{i=1}^K w(x_0^{(i)}) \exp(-\mathcal{E}(x_0^{(i)}))}{\sum_{i=1}^K w(x_0^{(i)})} \tag{3.21}$$

$$w(x) = \frac{\mathcal{N}(x; x_t, \sigma_t^2 I)}{q(x|x_t)} \tag{3.22}$$

According to Section 2.2.4, when the energy landscape is complex and high-dimensional, we require an optimal proposal roughly proportional to $\exp(-\mathcal{E}(x))\mathcal{N}(x_t; x, \sigma_t^2 I)$ to minimize the variance of IS estimator. Therefore, the original MC estimators can lead to high variance

estimation in complex tasks, while sampling from the optimal proposal requires techniques like MCMC which is expensive.

Improvement with AIS. A possible way to improve the performance of estimators in high dimensional space is via AIS. Following the previous discussion, we aim to draw samples by simulating an MCMC over a sequence of intermediate distributions logarithmically interpolated between the target and the initial proposal. Then, the AIS energy estimator can be estimated by LogSum of weights of these samples, i.e. $E_K^{\text{AIS}} = -\log \frac{1}{K}\sum_i w_i$, while the score one is computed by differentiation, i.e. $S_K^{\text{AIS}} = \nabla \log \sum_i w_i$.

We use an initial proposal then the same as before, i.e. $\pi_0(x) = q(x) = \mathcal{N}(x; x_t, \sigma_t^2 I)$. Given the steps of AIS, L, and according to Section 2.2.5, the intermediate distributions π_k can be expressed as follows by plugging q and $p(x) \propto \exp(-\mathcal{E}(x))\mathcal{N}(x_t; x, \sigma_t^2 I)$ into Equation 2.15:

$$\pi_k(x) \propto q(x)^{1-\frac{k}{L}} p(x)^{\frac{k}{L}} \tag{3.23}$$

$$\propto \mathcal{N}(x; x_t, \sigma_t^2) \exp\left(-\frac{k}{L}\mathcal{E}(x)\right) := \tilde{\pi}_k(x) \tag{3.24}$$

With K MC samples, we can compute the annealed importance weight of the i^{th} sample as follows:

$$w_i = \frac{\tilde{\pi}_1(x_0^{(i)})}{\tilde{\pi}_0(x_0^{(i)})} \frac{\tilde{\pi}_2(x_1^{(i)})}{\tilde{\pi}_1(x_1^{(i)})} \cdots \frac{\tilde{\pi}_L(x_{L-1}^{(i)})}{\pi_{L-1}(x_{L-1}^{(i)})} = \prod_{k=1}^{L} \exp\left(-\frac{1}{L}\mathcal{E}(x_{k-1}^{(i)})\right) \tag{3.25}$$

where $x_0^{(i)}$ is drawn from $\pi_0 = N(x; x_t, \sigma_t^2)$ and $x_k^{(i)}$ is by simulating a MCMC from $x_{k-1}^{(i)}$ to target π_k for $k = 1, ..., L-1$. In our setting, we stick the MCMC to a HMC. Therefore, we can establish AIS estimators for both energy and score as follows:

$$E_K^{\text{AIS}(L)}(x_t, t) := -\log \frac{1}{K}\sum_{i=1}^{K} w_i \tag{3.26}$$

$$= -\log \sum_{i=1}^{K} \exp\left(-\frac{1}{L}\sum_{k=1}^{L} \mathcal{E}(x_{k-1}^{(i)})\right) + \log K \tag{3.27}$$

$$= -\text{LogSumExp}\left(-\frac{1}{L}\sum_{k=1}^{L} \mathcal{E}(x_{k-1})\right) + \log K \tag{3.28}$$

$$S_K^{\text{AIS}(L)}(x_t, t) := -\nabla_{x_t} E_K^{\text{AIS}(L)}(x_t, t) \tag{3.29}$$

$$= \nabla_{x_t} \text{LogSumExp}\left(-\frac{1}{L}\sum_{k=1}^{L} \mathcal{E}(x_{k-1})\right) \tag{3.30}$$

Notice that by setting $L = 1$, then the AIS estimator recovers the original ones (2.61 and 3.3). A complete description of computing the AIS estimators is provided in Appendix C.

3.7 Incorporating Symmetries

Similar to iDEM, we consider applying EnDEM in physical systems with symmetry constraints like n-body system. Analogous to Proposition 1 proposed by Akhound-Sadegh et al. (2024), we prove that our MC energy estimator E_K is G-invariant under the same condition.

Proposition 4 *(ours) 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.,*

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

Proposition 4 shows that, when S_K is G-equivariant, E_K is G-invariant.

3.8 Summary

In this chapter, we introduce our method, EnDEM: Energy-based iDEM, by first providing the motivations and methodology. Then we theoretically discuss the advantage of targeting the annealed (or time-convolved) energies instead of scores, showing that the energy targets provide us more useful and less noisy training signal as well as have the potential to leverage statistical techniques for higher-quality samples.

On the other hand, we show that the original MC energy estimators (3.3) can be expressed as logarithm of an IS estimator (3.19), leading to an improvement (3.27 and 3.29) for both the energy and score estimators via AIS. Besides, E_K can be G-invariant when S_K is G-equivariant, suggesting that we can simply replace the training target for many-body systems.

Before presenting the experimental results in Chapter 5, we'll introduce an improvement of EnDEM via Bootstrap Energy Estimation in the next chapter.

Chapter 4

BEnDEM: EnDEM with Bootstrap Energy Estimation

In the last chapter, we propose EnDEM, which trains an energy network E_θ to target an estimator with less bias and variance. Even though EnDEM can be more expensive than DEM when simulating the reverse SDE, as a consequence of differentiating x over E_θ, it allows us more ways to improve performance based on the learned anneal energies. In this chapter, we seek to alleviate the high variance of targets when t is large in a bootstrapping way.

Notice that we use a variance exploding noising process in our setting, where x_0 is noised by adding a 0 mean noise with variance σ_t^2 at t, i.e. $x_t = x_0 + \sigma_t \varepsilon, \varepsilon \sim \mathcal{N}(0, I)$. Since the noise schedule $\sigma(t)$ is not bounded, the variance of x_t can be large when t increases. Therefore, the MC estimators can be very noisy at large t, resulting in useless training signals. In common generative tasks like image generation, the value of each pixel is known to range in $[0, 255]$. Then we can normalize them to $[-1, 1]$, and apply a noising process with continuous time $t \in [0, 1]$ to train a DM. The performance of DM is promising in this setting (Karras et al., 2022). While in some physical problems, it is often the case that the data region is unknown and we can't normalize them (Gee et al., 1997; Sarmadi et al., 2023). In this circumstance, we need to noise the data with larger $\sigma(t)$ (or equivalently, the same $\sigma(t)$ but with a larger time range $[0, T]$) to ensure the target distribution is mixed to the tractable prior, resulting to the problem of high variance of training target in EnDEM and DEM we mentioned previously.

An intuitive idea is using bootstrap energy estimation to reduce the variance of the estimator. Suppose the energy model E_θ fits well at time s, where σ_s^2 is not large and $E_K(x_s, s)$ can provide useful training signals. Then the energy estimator at (x_t, t) can be bootstrapped from the learned energy at s. Analogous to Equation 3.3, we sample $x_{s|t}^{(i)}$ around

x_t with variance of scale $\sigma_t^2 - \sigma_s^2$. Therefore, the variance of the target at t can be reduced by σ_s^2. Therefore, we can use such bootstrapped energy, instead of E_K, as the target of (x_t, t). Sequentially, we can use a set of time splits $0 = t_0 < t_1 < ... < t_N = T$, where we train the energy network as useful for $t \in [t_0, t_1]$ while targeting the modified energy estimator bootstrapped from last time split (i.e. t_{n-1}) for $t \in [t_{n-1}, t_n]$ for $n = 1, ..., N$.

While there is obviously no free lunch since the neural network can not perfectly learn (even though your network is powerful enough, its training targets are still noisy), and will produce errors in prediction. These inductive biases can be accumulated through the chain of bootstrapping, resulting in high bias of the training targets. Therefore, the design of time splits can be shown as a Variance-Bias trade-off, where a frequent bootstrapping strategy can reduce variance but introduce bias.

In this chapter, we will first propose and introduce our method, Bootstrap EnDEM (BEnDEM), providing a conceptual overview. Then we'll discuss the Variance-Bias trade-off for the bootstrapping method, from the perspective of a sequential energy estimator. At the end, we will discuss the ablation models for both EnDEM and BEnDEM which target the score estimators the same as iDEM but use a score network and result in learning the smoothed annealed energies.

4.1 BEnDEM

4.1.1 Bootstrap Energy Estimation

Suppose $0 \leq s < t \leq T$, given a noise schedule $\sigma(t)$ defined by a VE SDE, we have $\sigma_s < \sigma_t$ and $x_t | x_s \sim \mathcal{N}(x; x_s, (\sigma_t^2 - \sigma_s^2) I)$. The marginal distribution p_t can be obtained by integrating out x_s over the joint distribution $p(x_s, x_t)$

$$p_t(x_t) = \int p_s(x_s) \mathcal{N}\left(x_t; x_s, (\sigma_t^2 - \sigma_s^2) I\right) dx_s \tag{4.1}$$

$$= \int \frac{\exp(-\mathcal{E}_s(x_s))}{Z_s} \mathcal{N}(x_t; x_s, (\sigma_t^2 - \sigma_s^2) I) dx_s \tag{4.2}$$

where $\mathcal{E}_s$ is defined by Equation 3.2 and $Z_s = \int \exp(-\mathcal{E}_s(x_s))ds = Z_0$. Therefore, analogous to Equation 2.58 and Equation 3.2 we have

$$p_t(x_t) = \int \frac{\exp(-\mathcal{E}_s(x_s))}{Z_0} \mathcal{N}(x_t; x_s, (\sigma_t^2 - \sigma_s^2)I) dx_s \tag{4.3}$$

$$\mathcal{E}_t(x_t) = -\log \mathbb{E}_{\mathcal{N}(x;x_s,(\sigma_t^2-\sigma_s^2)I)} [\exp(-\mathcal{E}_s(x))] \tag{4.4}$$

$$\approx -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}_s(x_{s|t}^{(i)})), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x; x_s, (\sigma_t^2 - \sigma_s^2)I) \tag{4.5}$$

Equation 4.5 suggests an energy estimator bootstrapped from time s for smaller variance. Though we don't directly have access to $\mathcal{E}_s(x)$, we can use our energy network $E_\theta(x,s)$ as an approximation if it fits well at s. Therefore, we propose the Bootstrap Energy Estimator $E_K(x_t,t,s;\phi)$ and corresponding bootstrapping loss $\mathcal{L}_{\text{BEnDEM}}(x_t,t)$ as follows:

$$E_K(x_t,t,s;\phi) := -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-E_\phi(x_{s|t}^{(i)}, s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x; x_s, (\sigma_t^2 - \sigma_s^2)I) \tag{4.6}$$

$$= -\log \sum_{i=1}^{K} \exp(-E_\phi(x_{s|t}^{(i)}, s)) + \log K \tag{4.7}$$

$$\mathcal{L}_{\text{BEnDEM}}(x_t,t) = \|E_\theta(x_t,t) - E_K(x_t,t,s;\phi)\|^2 \tag{4.8}$$

where ϕ is the parameter with stopping-gradient; and E_ϕ refers to a teacher network in terms of distillation and can be either another pre-trained energy network or itself denoted by θ^-. For simplification, we define $E_K(x_t,t,0;\phi)$ as the MC energy estimator, i.e.

$$E_K(x_t,t,0;\phi) := E_K(x_t,t), \forall \phi \tag{4.9}$$

4.1.2 Geometrical Bootstrap Schedule

The VE SDE with a geometrical noise schedule is given by Karras et al. (2022) as:

$$\sigma(t) = \sigma_{\min} \left((\frac{\sigma_{\max}}{\sigma_{\min}})^{2t} - 1 \right)^{0.5} \tag{4.10}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are predefined and typically $\sigma_{\max}/\sigma_{\min}$ is very large, e.g. $1.0/0.0001$. This noise schedule results in the variance of added noise being geometrically increased across time t. Notice that the success of bootstrap energy estimation is based on variance control, i.e. we should always target a low variance target then the learned energy will be bootstrapped as another low-variance training target for a longer time. Therefore, we

propose a Geometrical Bootstrap schedule, which controls the variance of perturbation kernel $p(x_t|x_s)$ used for bootstrapping under a predefined level. Given a variance-control level β, the Geometrical Bootstrap schedule returns $N+1$ time splits $t_0,...,t_N$ of time range $[0,1]$ with

$$N = \left\lceil \frac{\sigma_{\max}^2 - 2\sigma_{\min}^2}{\beta} \right\rceil \tag{4.11}$$

$$t_n = \frac{1}{2\log(\sigma_{\max}/\sigma_{\min})} \log\left(\frac{n\beta}{\sigma_{\min}^2} + 2\right) \tag{4.12}$$

by solving $\sigma_{t_n}^2 - \sigma_{t_{n-1}}^2 = \beta$, $\sigma_{t_0}^2 = \sigma_{\min}^2$, $t_N = 1$ and $\sigma_{t_N}^2 = \sigma_{\max}^2 - \sigma_{\min}^2$. In practice, we treat $E_K(x_t,t,t_0;\phi)$ as $E_K(x_t,t,0;\phi)$ since we t_0 is very close to 0, and we set $t_N = \min(1,t_N)$ to preserve the time range boundary.

4.2 Training BEnDEM

The bootstrap energy estimation at t can be accurate only when the annealed energy at s is well-learned. Therefore, we need to sequentially learn energy at small t using the original MC energy estimator, then refine the estimator using bootstrap for large t. In terms of time splits $0 = t_0 < ... < t_N = 1$, we need to sequentially learn the annealed energies in the time range $[t_{n-1},t_n]$ for $n = 1,...,N$.

Given a time split t_n, and suppose the annealed energy for time in $[0,t_n]$ is well learnt, we train the energy network E_θ for $t \in [t_n,t_{n+1}]$ by targeting the bootstrap energy estimator from $s \in [t_{n-1},t_n]$, resulting to following loss:

$$\mathcal{L}_{\text{BEnDEM}}(n,\theta;\lambda) := \mathbb{E}_{t,s,x_0,x_t}\left[\lambda(t)\left\|E_\theta(x_t,t) - E_K(x_t,t,s;\phi)\right\|^2\right] \tag{4.13}$$

where $t \sim U[t_n,t_{n+1}]$, $s \sim U[t_{n-1},t_n]$ and $n = 0,...,N-1$. And we set $U[t_{-1},t_0]$ to be $\delta(0)$, i.e. starting from the MC energy estimator (3.3). However, such a training scheme is inefficient as it requires sequentially appending the time range from $[t_0,t_1]$ until $[t_0,t_N] = [0,1]$, and in the n^{th} iteration we need to compute a loss $\sum_{i=1}^n \mathcal{L}_{\text{BEnDEM}}(i,\theta;\lambda)$. To improve efficiency, we return to $t \sim U[0,1]$ as DMs, by leveraging a weighting function $w(n)$ related to the loss in the last time interval $[t_{n-1},t_n]$. Ideally, when E_θ in $[t_{n-1},t_n]$ is not well trained, i.e. a high loss in this range, the weighting function returns a low value that ignores the training signal provided by the Bootstrap energy estimator. Therefore we have

$$\mathcal{L}_{\text{BEnDEM}}(\theta;\lambda) := \mathbb{E}_{t,s,x_0,x_t}\left[w(n)\lambda(t)\left\|E_\theta(x_t,t) - E_K(x_t,t,s;\phi)\right\|^2\right] \tag{4.14}$$

where $t \in U[0,1]$ and $n = n(t) := \arg\{i : t_{i-1} \leq t \leq t_i\}$; and given a t, we first determine its interval $[t_n, t_{n+1}]$ and corresponding index n, then s is uniformly sampled from $[t_{n-1}, t_n]$ and $w(n)$ can be predefined or determined by loss at s.

Notice that, losses at s can describe the quality of training and therefore we can utilize it as signals for bootstrapping. In practice, we first evaluate the previous model E_θ at s and t respectively, and without gradients. We compute the losses l_s and l_t w.r.t targeting $E_K(x,s)$ and $E_K(x,t)$, and their ratio $r_{st} := l_s/l_t$. Ideally, the target at s has a smaller variance than t and therefore $r_{st} > 1$ if the model is well trained at s. Thus, we have: when $E_\theta(x,s)$ is well trained, r_{st} will be small and we should use bootstrapping; otherwise, r_{st} would be ranged from 0 to 1, and we can use a probability of r_{st} to accept "using Bootstrap estimator". Also, we stick to using the last step model with a stopping gradient, denoted as θ^-, to compute the bootstrap estimator. Algorithm 3 describes a complete inner-loop of BEnDEM training, while its outer-loop follows Algorithm 2.

Algorithm 3 Bootstrap Energy-based iDEM 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 **do**
  $x_0 \leftarrow \mathcal{B}$.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
  $s \sim \mathcal{U}(t_{n-1}, t_n), x_s \sim \mathcal{N}(x_0, \sigma_s^2)$
  $l_s \leftarrow \|E_K(x_s,s) - E_\theta(x_s,s)\|^2$
  $l_t \leftarrow \|E_K(x_t,t) - E_\theta(x_t,t)\|^2$
  $\alpha \leftarrow l_s/l_t$
  with probability α,
    $\mathcal{L}_{\text{BEnDEM}}(x_t,t) = \|E_K(x_t,t,s;\theta^-) - E_\theta(x_t,t)\|^2$
  Otherwise, ▷ Use MC estimator if the model is not well trained
    $\mathcal{L}_{\text{BEnDEM}}(x_t,t) = \|E_K(x_t,t) - E_\theta(x_t,t)\|^2$
  $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{BEnDEM}})$
end while

Ensure: E_θ

4.3 Variance-Bias trade-off: a Sequential Estimator perspective

BEnDEM can reduce the variance of the target while introducing more bias because of error accumulation, by intuition. In this section, we theoretically discuss this variance-bias trade-off, by leveraging a sequential MC energy estimator.

4.3.1 Sequential MC Energy Estimator

The MC energy estimator is given in Equation 3.3, while its mean, variance and bias are given in Equation 3.12 and 3.13, by assuming sub-Gaussianness of $\exp(-\mathcal{E}(x))$ with mean and variance defined by Equation 3.9 and 3.10. We first review these equations as follows:

$$
\begin{aligned}
m_{0t}(x_t) &= \mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[\exp(-\mathcal{E}(x))] \\
v_{0t}(x_t) &= \mathrm{Var}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}(\exp(-\mathcal{E}(x))) \\
E_K(x_t,t) &= -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)})), \quad x_{0|t}^{(i)} \sim \mathcal{N}(x;x_t,\sigma_t^2) \\
\mathbb{E}[E_K(x_t,t)] &= -\log m_{0t}(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K} \\
&= \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K} \\
\mathrm{Var}(E_K(x_t,t)) &= \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K}
\end{aligned}
$$

While Equation 4.4 suggests an MC energy estimator at t bootstrapped from energy $\mathcal{E}_s$ from s, where $\mathcal{E}_s(x)$ can be estimated by the original MC energy estimator, resulting to the following Sequential MC energy estimator:

$$
E_K^{\mathrm{Seq}}(x_t,t) := -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-E_K(x_{s|t}^{(i)},s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x;x_t,(\sigma_t^2-\sigma_s^2)I) \tag{4.15}
$$

$$
= -\log\frac{1}{K}\sum_{i=1}^{K}\frac{1}{K}\sum_{j=1}^{K}\exp(-\mathcal{E}(x_{0|s}^{(ij)})) \tag{4.16}
$$

where $x_{0|s}^{(ij)} \sim \mathcal{N}(x; x_{s|t}^{(i)}, \sigma_s^2 I)$ and $x_{s|t}^{(i)} \sim \mathcal{N}(x; x_t, (\sigma_t^2 - \sigma_s^2) I)$. By the property of Gaussian, the Sequential MC energy estimator is equivalent to

$$E_K^{\text{Seq}}(x_t, t) = -\log \frac{1}{K^2} \sum_{i=1}^{K} \sum_{j=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(ij)})), \quad x_{0|t}^{(ij)} \sim \mathcal{N}(x; x_t, \sigma_t^2 I) \tag{4.17}$$

which is the original MC energy estimator with K^2 MC samples, i.e. $E_K^{\text{Seq}}(x_t, t) = E_{K^2}(x_t, t)$. And we can obtain its mean and variance by plugging K^2 into Equation 3.12 and 3.13.

4.3.2 Bootstrap(1) Energy Estimator

A sequential MC energy estimator can be viewed as squaring the MC samples of the original estimator, resulting in a quadratic decay rate of variance and bias w.r.t the Num. of MC sample K. However, it is obviously trading computation for performance which is inefficient. Instead, if we can compress the information at s, i.e. learn a powerful neural network can output the $\mathcal{E}_s$ end-to-end without K evaluations of the energy function , then we can evaluate K times of this neural network instead of evaluating K^2 times of $\mathcal{E}$, resembling a Bootstrap MC Energy Estimator (4.6). We now consider the one bootstrapped once from s to estimate $\mathcal{E}_t$, where $\mathcal{E}_s(x)$ is predicted by an energy network $E_\theta(x, s)$ by learning from the original estimator $E_K(x, s)$, i.e.

$$E_K^{B(1)}(x_t, t, s; \phi) := -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-E_\phi(x_{s|t}^{(i)}, s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x; x_t, (\sigma_t^2 - \sigma_s^2) I) \tag{4.18}$$

Ideally, the Bootstrapping energy estimator is equivalent to the Sequential estimator, resulting in $1/K$ smaller variance and bias compared with E_K. However, the inductive bias of neural networks can introduce extra bias.

Since the training target of $E_\theta(x, s)$ is noisy, whose mean and variance are given by Equation 3.12 and 3.13. Suppose we are using l_2 loss and have a perfect optimizer. Then the optimal energy network returns $E_{\phi^*}(x_s, s) = \mathbb{E}[E_K(x_s, s)]$, with an expected loss equals to $\text{Var}(E_K(x_s, s))$, therefore

$$E_K^{B(1)}(x_t, t, s; \phi^*) = -\log \frac{1}{K} \sum_{i=1}^{K} \exp\left(-\left(-\log m_{0s}(x_{s|t}^{(i)}) + \frac{v_{0s}(x_{s|t}^{(i)})}{2 m_{0s}^2(x_{s|t}^{(i)}) K} \right) \right) \tag{4.19}$$

$$= -\log \frac{1}{K} \sum_{i=1}^{K} m_{0s}(x_{s|t}^{(i)}) - \log \frac{1}{K} \sum_{i=1}^{K} \exp\left(-\frac{v_{0s}(x_{s|t}^{(i)})}{2 m_{0s}^2(x_{s|t}^{(i)}) K} \right) \tag{4.20}$$

where $x_{s|t}^{(i)} \sim \mathcal{N}(x; x_t, (\sigma_t^2 - \sigma_s^2)I)$. We further link the Bootstrap energy estimator and the Sequential one by assuming a large K and $\sigma_t^2 - \sigma_s^2$ is small:

$$E_K^{B(1)}(x_t, t, s; \phi^*) \approx E_K^{\text{Seq}}(x_t, t) - \log \frac{1}{K} \sum_{i=1}^{K} \exp\left(-\frac{v_{0s}(x_{s|t}^{(i)})}{2m_{0s}^2(x_{s|t}^{(i)})K} \right) \tag{4.21}$$

$$\mathbb{E}[E_K^{B(1)}(x_t, t, s; \phi^*)] \approx \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^2} + \frac{v_{0s}(x_t)}{2m_{0s}^2(x_t)K} \tag{4.22}$$

the first assumption is trivial, and the second one is reasonable as well since we always have a variance-control level β in our bootstrap setting. Therefore, the optimal energy network obtained by targeting the Bootstrap estimator (4.6) is $E_{\theta^*}(x_t, t) = \mathbb{E}[E_K^{B(1)}(x_t, t, s; \phi^*)]$, which includes two terms of bias:

1. $\frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^2}$: the bias of a Sequential energy estimator, with a quadratic decay rate.

2. $\frac{v_{0s}(x_t)}{2m_{0s}^2(x_t)K}$: the bias introduced by the inductive bias of neural network, which depends on the bootstrapped time s and the corresponding statistics v_{0s} and m_{0s}.

A complete proof of Equation 4.21 and 4.22 are given in Appendix B.6.1.

Therefore, a Bootstrap(1) estimator can outperform the MC energy estimator in two perspectives:

Variance. It's trivial that the variance of the Bootstrap energy estimator is smaller than that of the original estimator, especially when t is large. Therefore the Bootstrap energy estimator can provide more useful and less noisy training signals.

Bias. Equation 4.22 shows the bias of Bootstrap energy estimator. In practice, we're not using the same K for different estimators, since evaluating the neural network can be as expensive as evaluating $\mathcal{E}$. Suppose we use K_1 MC samples for the MC energy estimator E_{K_1}, then we can choose K_2 MC samples for the bootstrap one $E_{K_2}^{B(1)}$ to ensure no amplified bias, where

$$K_2 > \frac{1}{1 - (v_{0s}/m_{0s}^2)/(v_{0t}/m_{0t}^2)} \tag{4.23}$$

4.3.3 Bootstrap(n) Energy Estimator

In general, we care about the performance of an estimator bootstrapped $n \in \{0, 1, 2, ..., N-1\}$ times given a bootstrap schedule $0 = t_0 < ... < t_N = 1$. We define a Bootstrap(n) estimator as follow:

Definition 1 *(Bootstrap(n) Energy Estimator) Given time splits* $0 = t_0 < ...t_N < 1$, $E_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}$ *and* $n \in \{0,...,N-1\}$, *for* $\forall (x_t,t,s) \in \mathbb{R} \times [t_n, t_{n+1}] \times [t_{n-1}, t_n]$, *a Bootstrap(n) estimator is defined as:*

$$E_K^{B(n)}(x_t,t,s;\theta) := -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-E_\theta(x_{s|t}^{(i)}, s)) \tag{4.24}$$

where $x_{s|t}^{(i)} \sim \mathcal{N}(x; x_t, (\sigma_t^2 - \sigma_s^2)I)$ *and with initial condition*

$$E_K^{B(0)}(x_t,t,s;\theta) := E_K(x_t,t), \forall \theta \in \Theta, \forall s \in \mathbb{R} \tag{4.25}$$

Then the optimal value of the energy network trained by Bootstrap energy estimation is given by the following Proposition 5:

Proposition 5 *Given time splits* $0 = t_0 < t_1 < ... < t_N = 1$, $n \in \{0,...,N\}$ *and a neural network* E_θ. *For any fixed trajectory* $\{s_i\}_{i=0}^n$ *with* $s_i \in [t_{i-1}, t_i]$ *and* $s_0 = 0$, *if* E_θ *is optimal for any* $u \le t_n$ *by sequentially optimising*

$$\theta^{(i)} = \arg\min_\theta \| E_\theta(x_u, u) - E_K^{B(i)}(x_u, u, s_i; \theta^{(i-1)}) \|^2 \tag{4.26}$$

for $\forall i \in \{0,...,n-1\}$ *and* $\forall u \in [t_i, t_{i+1}]$, *then for* $\forall (x_t,t) \in \mathbb{R}^d \times [t_n, t_{n+1}]$, *the neural network optimized by targeting* $E_K^{B(n)}(x_t,t,s_n;\theta^{(n-1)})$ *has the optimal value:*

$$E_{\theta^{(n)}}(x_t,t) = \mathcal{E}_t(x_t) + \sum_{j=1}^{n+1} \frac{v_{0,s_j}(x_t)}{2m_{0,s_j}^2(x_t) K^j} \tag{4.27}$$

where $s_{n+1} = t$. *Especially, when* $n = 0$ *it resembles the optimal values by targeting the MC energy estimator (3.3), i.e.*

$$E_{\theta^{(0)}}(x_t,t) = \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t) K} \tag{4.28}$$

A complete proof can be found in Appendix B.6.2. Proposition 5 realizes that the bias of the optimal network trained by Bootstrap Energy Estimator can be composed into two terms:

1. $\frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^{n+1}}$: the bias of a Sequential(n) energy estimator, with a geometric decay rate.
2. $\sum_{j=1}^n \frac{v_{0,s_j}(x_t)}{2m_{0,s_j}^2(x_t)K^j}$: the accumulated bias introduced by bootstrapping

while Proposition 5 also suggests that the variance of our training target can be roughly expressed as:

$$\mathrm{Var}(E_K^{B(n)})(x_t,t,s_n;\theta^{(n-1)}) \approx \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K^{n+1}} \tag{4.29}$$

which is geometrically reduced compared with the MC Energy estimator (3.3). Therefore, BEnDEM can be realized as a variance-bias trade-off of its training target.

Also, since we have the bias of a Bootstrap(n) energy estimator, we can choose a K_{n+1} such that:

$$\sum_{i=1}^{n+1} \frac{v_{0,s_i}(x_t)}{2m_{0,s_i}^2(x_t)\prod_{j=1}^{i}K_j} < \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K_1} \tag{4.30}$$

$$\Leftrightarrow K_{n+1} > \frac{1}{\prod_{i=2}^{n}K_i - \frac{m_{0t}^2(x_t)}{v_{0t}(x_t)}\sum_{i=1}^{n}\frac{v_{0,s_i}(x_t)}{m_{0,s_i}^2(x_t)}\prod_{j=i+1}^{n}K_j}, \quad n \geq 1 \tag{4.31}$$

to ensure no extra bias, suppose $\{K_i\}_{i=1}^n$ are given and we define $\prod_{j=m}^{m-1}K_j = 1, \forall m \in \mathbb{Z}$.

4.4 Summary

In this chapter, we introduce BEnDEM, which improves EnDEM by using self-bootstrapping energy estimation to trade accumulated bias and variance of the target. We first provide a way for training such as a Bootstrapping-based model, which avoids inefficient sequential training. In the end, we theoretically discuss the variance-bias trade-off of BEnDEM, showing that a Bootstrap(n) estimator can reduce the variance of learning target geometrically while introducing accumulated bias. Figure 4.1 illustrates the difference between DEM, EnDEM, and Bootstrap EnDEM.

In the next chapter, we're going to present the experimental results of both EnDEM and BEnDEM on various datasets, showing the capacity of our methods.

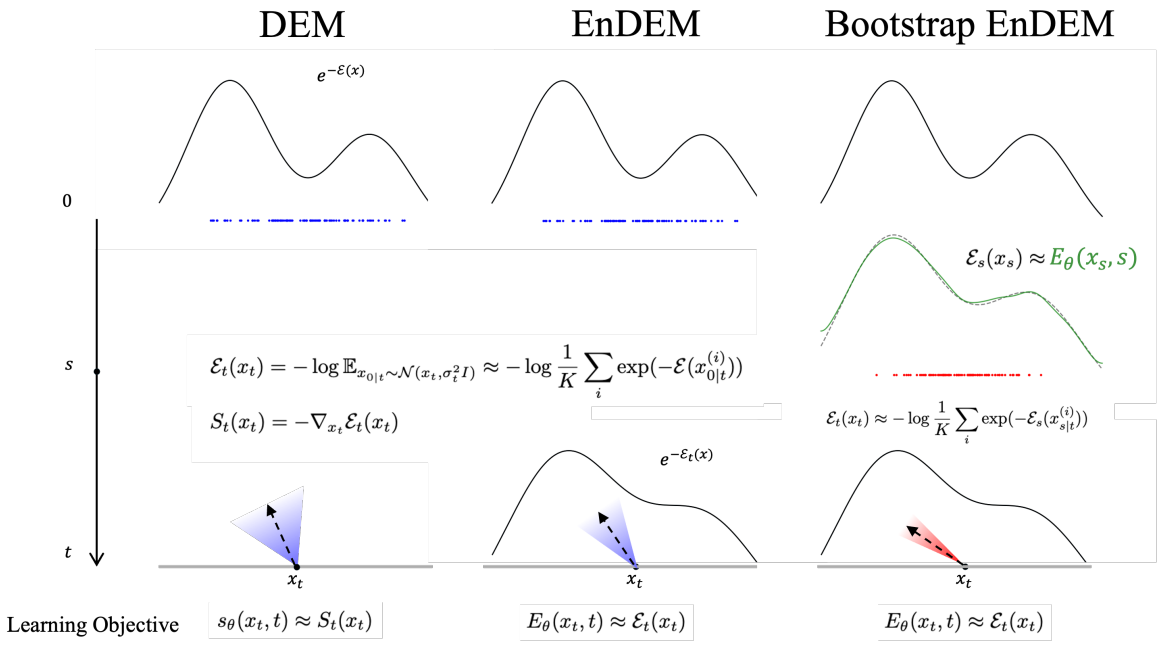


Fig. 4.1 Three methods for sampling from Boltzmann-type distribution leveraging diffusion models. ● and ● represent samples for estimators; ↖ for estimated scores; light cones, i.e. ▲ and ▲, for variance of scores; and the green curve represents trained energy model outputs. **Lelf:** DEM regresses to a consistent MC score estimator and the trained neural network can be injected into a reverse SDE integration for sampling directly. **Middle:** EnDEM regresses to a consistent MC energy estimator which is less noisy and allows evaluating the convolved energy, while the trained energy network requires additional differentiation to compute scores. **Right:** Bootstrap EnDEM proposes an energy estimator bootstrapped from time s instead of 0, resulting in less noisy targets when t is large.

Chapter 5

Experiments and Results

In this chapter, we describe how to evaluate our methods, EnDEM and BEnDEM, on several datasets and compare them with the baseline model iDEM (Akhound-Sadegh et al., 2024). We'll describe the dataset and evaluation metrics before presenting the results in the next chapter. Also for the ablation study, we conduct experiments on training a (bootstrap or not) energy network in an iDEM style, which will be described in Section 5.3. Then we will describe the details of the model setting in each dataset. At the end, we will present the experimental results as well as comparison and discussion about our models.

5.1 Datasets

In this section, we'll describe 2 different datasets used for our experiments: GMM-40 and DW-4. While two other datasets, LJ-13 and LJ-55 are provided in Appendix G.

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,

$$\Sigma = \begin{pmatrix} 40 & 0 \\ 0 & 40 \end{pmatrix} \tag{5.1}$$

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

$$p_{gmm}(x) = \frac{1}{40} \sum_{i=1}^{40} \mathcal{N}(x; \mu_i, \Sigma) \tag{5.2}$$

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

$$\mathcal{E}^{GMM}(x) = -\log p_{gmm}(x) \tag{5.3}$$

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

$$\mathcal{E}^{DW}(x) = \frac{1}{2\tau}\sum_{ij} a(d_{ij} - d_0) + b(d_{ij} - d_0)^2 + c(d_{ij} - d_0)^4 \tag{5.4}$$

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. (2023) as the "Ground truth" samples for evaluating.

5.2 Evaluation Metrics

Negative Log Likelihood (NLL). The negative log-likelihood is a classical metric measuring how likely a test dataset is under a model, i.e. $-\log p_{model}(D_{test})$. For score-based diffusion models, we can numerically compute the NLL by an ODE, while an energy-based diffusion model allows us to direct unnormalized density computation and approximate the log partition function by Monte Carlo. While in our experiments, we follow the setting used by Akhound-Sadegh et al. (2024), which computes the exact likelihood from a continuous normalizing flow (CNF) model and is shown to be a relatively good estimator. The CNF is trained by an optimal transport flow matching on samples from each model (OT-CFM) (Tong et al., 2024). Under this setting, we can compare the sample quality of various model architectures fairly using the same model architecture, training regime, and numerical likelihood approximation independent of the sampler form. Then given a test sample x, its likelihood can be estimated as

$$\log p_{model}(x) = \log p_{prior}(x) + \int_1^0 -\mathrm{Tr}(\frac{df}{dx_t})dt \tag{5.5}$$

where $x(t) = x_1 - \int_1^0 f(t,x)dt$ and $f : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ the flow function.

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:

$$\mathcal{W}_2(\mu,\nu) = (\inf_\pi \int \pi(x,y)d^2(x,y)dxdy)^{\frac{1}{2}} \tag{5.6}$$

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 $d(x,y) = \|x-y\|_2$. For low-dimensional data, we compute the data $\mathcal{W}_2$ by simply binning the data; while for high-dimensional ones, we instead compute the $\mathcal{W}_2$ based on interatomic distances, i.e. distances between the nuclei of atoms in a molecule or crystal.

Effective Sample Size (ESS). This metric measures the quality and efficiency of a set of weighted samples. It quantifies the number of independent and identically distributed (i.i.d.) samples that would be needed to achieve the same level of estimation accuracy as the weighted samples. ESS is defined as

$$ESS = \frac{1}{n\sum_{i=1}^n \tilde{w}_i^2} = \frac{(\sum_{i=1}^n w_i)^2}{n\sum_{i=1}^n w_i^2} \in [\frac{1}{n}, 1] \tag{5.7}$$

where $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^n w_j}$ and $w_i = \exp(-\mathcal{E}(x_i))/p_{model}(x_i)$.

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

$$TV(P,Q) = \frac{1}{2}\int_\Omega |p(x) - q(x)|dx \tag{5.8}$$

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

Log Partition Function ($\log Z$). To estimate $\log Z$, Akhound-Sadegh et al. (2024) uses an importance sampling where the proposal is given by the OT-CMF model, i.e. $q(x) = p_{model}(x)$.

Then

$$\log Z = \log \mathbb{E}_{q(x)}\left[\frac{\exp(-\mathcal{E}(x))}{q(x)}\right] \tag{5.9}$$

$$\geq \mathbb{E}_{q(x)} \log\left[\frac{\exp(-\mathcal{E}(x))}{q(x)}\right] = \mathbb{E}_{q(x)}[-\mathcal{E}(x) - \log p_{model}(x)] \tag{5.10}$$

which yields a lower bound of $\log Z$. Therefore, we favor the sampler with the largest $\log Z$ estimation.

5.3 Ablation Models

Notice that our methods, EnDEM and BEnDEM, are different from iDEM since they model an energy network to target an energy estimator. For the ablation study, one can use the score estimator to train an energy network, similar to the ones using denoising score matching target to train an EBM (Salimans and Ho, 2021).

5.3.1 DEM-EN: DEM with Energy Network

Using the score estimator to train an energy network E_θ is simply changing the model s_θ in Equation 2.63 into $-\nabla_x E_\theta(x,t)$. To preserve conservative, we also targets the system energy $\mathcal{E}$ for $t = 0$ as a regularizor, i.e.

$$\mathcal{L}_{\text{DEM-EN}}(\theta;\lambda) := \mathbb{E}_{t,x_0,x_t}[\lambda(t)\| - \nabla_x E_\theta(x_t,t) - S_K(x_t,t)\|^2 + \alpha\|E_\theta(x_0,0) - \mathcal{E}(x_0)\|^2] \tag{5.11}$$

while its first term can be further expressed as follows by plugging Equation 2.61:

$$\mathbb{E}_{t,x_0,x_t}\left[\lambda(t)\left\|\nabla_x\left(E_\theta(x_t,t) + \log\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)}))\right)\right\|^2\right] \tag{5.12}$$

$$= \mathbb{E}_{t,x_0,x_t}\left[\lambda(t)\left\|\nabla_x\left(E_\theta(x_t,t) - E_K(x_t,t)\right)\right\|^2\right] \tag{5.13}$$

where $x_{0|t}^{(i)} \sim \mathcal{N}(x; x_t, \sigma_t^2 I)$.

5.3.2 BDEM-EN: DEM-EN with Bootstrap Energy Estimation

Analogous to EnDEM to BEnDEM, we hereby propose another ablation study that improves the DEM-EN via Bootstrap Energy Estimation. Similarly, we rewrite the loss of BEnDEM

(4.14) to target Bootstrapped scores, i.e.

$$\mathcal{L}_{\text{BDEM-EN}}(\theta;\lambda) := \mathbb{E}_{t,s,x_0,x_t}\left[w(n)\lambda(t)\left\|\nabla E_\theta(x_t,t) - \nabla E_K(x_t,t,s;\phi)\right\|^2\right] \tag{5.14}$$

$$= \mathbb{E}_{t,s,x_0,x_t}\left[w(n)\lambda(t)\left\|\nabla\left(E_\theta(x_t,t) - E_K(x_t,t,s;\phi)\right)\right\|^2\right] \tag{5.15}$$

However, it can be computationally expensive since we target a bootstrapped score computed by differentiating the output of a neural network, though we can compute as Equation 5.15 for acceleration.

5.4 Experiment Settings

Setup. In this section, we describe the settings and hyper-parameters used for different tasks, i.e. GMM, DW-4, LJ-13, and LJ-15. These settings and hyper-parameters mainly follow the ones proposed by Akhound-Sadegh et al. (2024). We combine our implementation of EnDEM, BEnDEM, DEM-EN, and BDEM-EN with the DEM repo released by Akhound-Sadegh et al. (2024). Our codes are implemented in PyTorch; all neural networks are optimized using Adam; all experiments are run on NVIDIA A100 GPUs with 40GB of VRAM. Besides, all noising schedules are variance exploding and based on Equation 4.10, therefore we tune σ_{min} and σ_{max} for different tasks.

Basic Neural Network: Score-based v.s. Energy-based. For a fair comparison, we should use similar model sizes for both score-based model (iDEM) and energy-based ones (EnDEM, BEnDEM, DEM-EN, and BDEM-EN). Also, for equivariant systems, the model output of a score network is equivariant, while that of an energy network should be invariant. Therefore, suppose we define a score network using an architecture f_θ, i.e. $s_\theta(x,t) = f_\theta(x,t)$, we define our energy network as $E_\theta(x,t) = f_\theta(x,t).sum() + c$, where c is a learnable scalar. Therefore, the energy network is only 1 parameter more than the score network and is also invariant of inputs. We then name f_θ to be the basic model in the rest of this section. Without statements, the score network and energy network are parameterized as above.

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 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. All models are trained with a geometric noise schedule with $\sigma_{\text{min}} = 1e-5$, $\sigma_{\text{max}} = 1$, $K = 500$ samples for computing both

the MC score estimator S_K and MC energy estimator E_K, $K = 400$ samples for computing the Bootstrap energy estimator E_K^B and Bootstrap score estimator $S_K^B = \nabla E_K^B$ and we clipped the norm of S_K and S_K^B to 70. All models are trained with a learning rate of $5e-4$. To stabilize EnDEM and BEnDEM training, we clip the generated samples into $[-2,2]$ for each dimension to avoid learning extreme values of energy. For fairness, we employ this clipping strategy for all models.

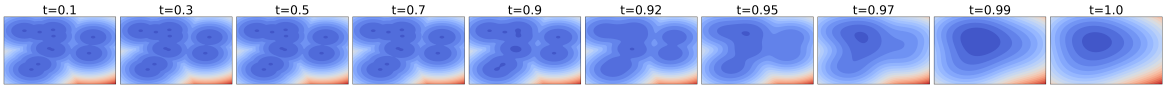
DW-4. All models use an EGNN with 3 message-passing layers and a 2-hidden layer MLP of size 128. All models are trained with a geometric noise schedule with $\sigma_{\text{min}} = 1e-5$, $\sigma_{\text{max}} = 3$, a learning rate of $1e-3$, $K = 1000$ samples for computing S_K and E_K, $K = 400$ samples for computing S_K^B and E_K^B, and we clipped S_K and S_K^B to a max norm of 20. To stabilize training as we mentioned in the GMM-40 setting, we first identify the smallest and largest values of the validation set data for each dimension and clip all samples into this range.

For all datasets. We use clipped scores as targets for DEM, DEM-EN, and BDEM-EN training for all tasks. Meanwhile, we also clip scores during sampling, when calculating the reverse SDE integral. These settings are shown to be crucial (Akhound-Sadegh et al., 2024) especially when the energy landscape is non-smooth and exists extremely large energies or scores, like LJ-13 and LJ-55. While we're learning unadjusted energy for EnDEM and BEnDEM, the training can be unstable and therefore we employ sample clipping on top of the setting of DEM. Due to the limited schedule of this thesis, we provide results of GMM-40 and DW-4 in the main context while leaving the description and early-stage results of LJ-13 and LJ-55 in Appendix G. Also, we're exploring a more stable training strategy for EnDEM and BEnDEM, without the non-trivial sample clipping trick, and will leave the results in our paper in the future.

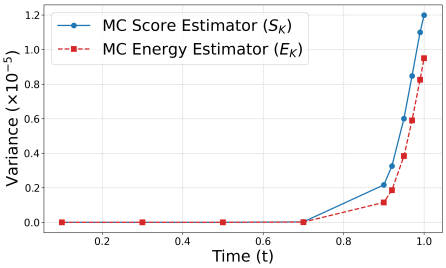
5.5 Variance and Bias of MC estimators: a 2D GMM case study

To justify the theoretical results for the variance of the MC energy estimator (3.3) and MC score estimator (2.61), we first empirically explore a 2D GMM, which has 10 modes located in $[-1,1]^2$ and with the following density:

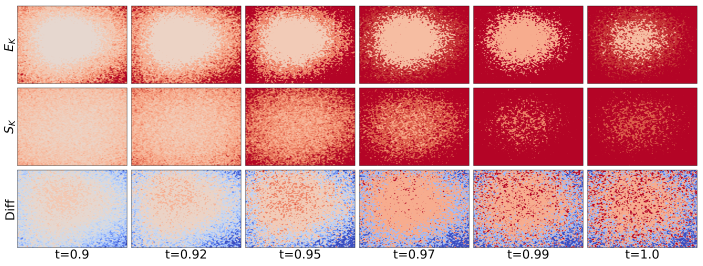
$$p'_{GMM}(x) = \frac{1}{10}\sum_{i=1}^{10}\mathcal{N}\left(x;\mu_i,\frac{1}{40}I\right) \tag{5.16}$$



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



(b) Expected variance of estimators



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

Fig. 5.1 (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 $t = 0$ to $t < 0.9$ since the variance of both estimators are small. The colormap ranges from blue (low) to red (high), where blues are negative while reds are positive.

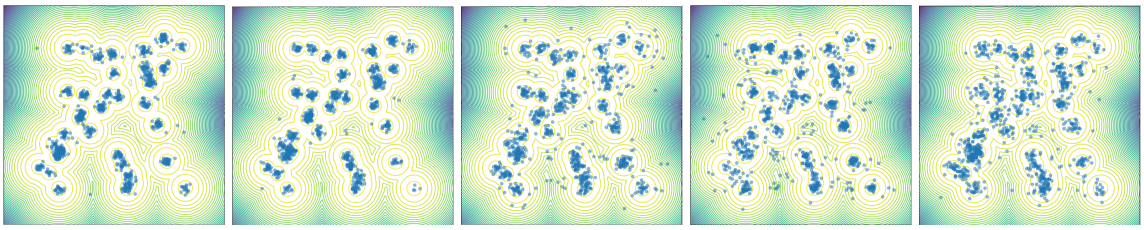
while the marginal perturbed distribution at t can be analytically derived following Gaussian's property:

$$p_t(x) = (p'_{GMM} * \mathcal{N}(0, \sigma_t^2)(x_t) = \frac{1}{10}\sum_{i=1}^{10} \mathcal{N}\left(x; \mu_i, \left(\frac{1}{40} + \sigma_t^2\right) I\right) \tag{5.17}$$

given a VE noising process.

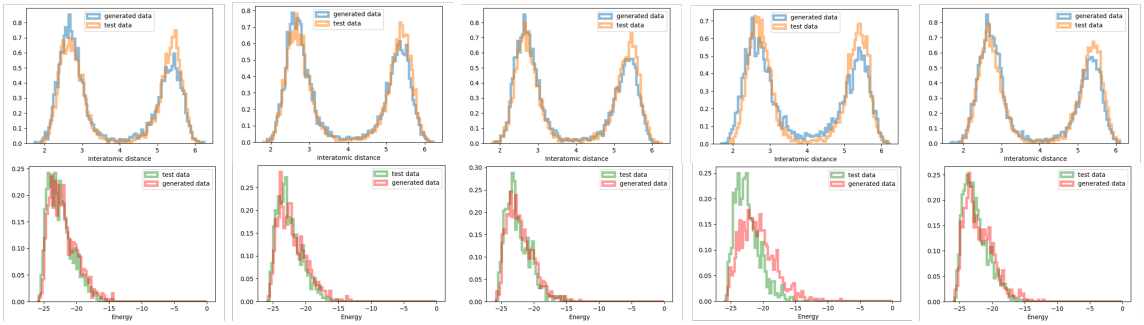
We empirically estimate the variance for each pair of (x_t, t) by simulating 10 times the MC estimators. Besides, we estimate the expected variance over x for each time t, i.e. $\mathbb{E}_{p_t(x_t)}[\mathrm{Var}(E_K(x_t,t))]$ and $\mathbb{E}_{p_t(x_t)}[\mathrm{Var}(S_K(x_t,t))]$.

Figure 5.1a shows that, the variance of both MC energy estimator and MC score estimator increase as time increases. While the variance of E_K can be smaller than that of S_K in most areas, especially when the energies are low (see Figure 5.1c). And Figure 5.1b shows that in expectation over true data distribution, the variance of E_K is always smaller than that of S_K across $t \in [0,1]$.



(a) EnDEM (b) BEnDEM (c) DEM-En (d) BDEM-En (e) iDEM

Fig. 5.2 Contour lines for the ground truth density of GMM-40 (5.4). Colored points, i.e. •, represent samples from each model.



(a) EnDEM (b) BEnDEM (c) DEM-En (d) BDEM-En (e) iDEM

Fig. 5.3 **Top:** histograms of the interatomic distance of drawn samples and test data (i.e. ground-truth samples). **Bottom:** histograms of energy of drawn samples and test data.

5.6 Main results

Figure 5.2 shows the contour of ground-truth GMM-40 density as well as samples drawn by each sampler. It shows that samples generated from iDEM can be deviated into gaps between modes, while the ones generated from EnDEM can be much more concentrated on all modes with few samples located in relatively high-energy regions. Furthermore, BEnDEM can slightly improve the performance by reducing the number of samples corresponding to high energy. However, the ablation models, DEM-En and BEnDEM-En, are difficult to train and result in more high-energy samples.

To demonstrate sample quality for higher dimensional data, e.g. DW-4, we visualize the histograms of interatomic distance and energy for samples drawn from each model and the ground-truth ones respectively, which are shown in Figure 5.3. These histograms demonstrate that DEM, EnDEM, and BEnDEM can generate samples that match the ground truth histograms well, while BDEM-En can be as difficult to train as GMM-40.

To quantitatively compare our models with the ablation ones and baseline one, we compare a suit of metrics provided in Table 5.1, including negative log-likelihood (NLL),

Table 5.1 Sampler performance for negative log-likelihood (NLL ↓), Effective Sample Size (ESS ↑), 2-Wasserstein metrics ($\mathcal{W}_2$ ↓), Toval Variation (TV ↓), and log partition function ($\log Z$ ↑).

Energy →	GMM-40 ($d=2$)					DW-4 ($d=8$)				
Algorithm ↓	NLL	ESS	$\mathcal{W}_2$	TV	$\log Z$	NLL	ESS	$\mathcal{W}_2$	TV	$\log Z$
iDEM†	6.96	0.73*	7.42	0.82	-0.34	7.17	0.825	2.13	0.10	29.567
iDEM (rerun)	6.50	0.27	5.28	0.87	-0.36	18.82	0.368	**1.88**	**0.09**	41.235
EnDEM	6.87	0.33	4.08	0.82	-0.02	18.68	0.379	1.92	0.10	41.448
BEnDEM	6.94	**0.36**	**3.84**	**0.80**	**0.16**	**18.61**	**0.392**	**1.88**	0.10	41.027
DEM-En	**6.13**	0.29	7.62	0.82	-0.69	19.09	0.376	**1.88**	**0.09**	**41.598**
BDEM-En	6.22	0.23	6.47	0.83	-0.63	19.09	0.376	2.04	0.20	41.598

Effective Sample Size (ESS), 2-Wasserstein metrics ($\mathcal{W}_2$), Toval Variation (TV), and log partition function ($\log Z$). However, even though we used exactly the same codes provided by Akhound-Sadegh et al. (2024), we were not able to reproduce similar results they provided. Therefore, we first report the mean of metrics given by Akhound-Sadegh et al. (2024), denoted as iDEM†. Then we rerun their repository[1] and report the metrics for iDEM, denoted as iDEM (rerun). We notice that all metrics are slightly different from 10% or even more, especially ESS and $\mathcal{W}_2$. For fairness, we present our comparison using the rerun iDEM.

Table 5.1 shows that EnDEM can outperform iDEM in most metrics, showing the potential of learning the less noisy energy estimator instead of the score estimator. Furthermore, BEnDEM can further improve performance and achieve the best results in most metrics, indicating that the energy model can fit low-level noise-convolved energy well and results in bootstrap energy estimator being able to provide more useful training signals by utilizing those learned energies. For ablation study, we notice that the performance of DEM-En and BDEM-En is not as well as the energy-based ones, i.e. EnDEM and BEnDEM, possibly as a consequence of the instability of training.

Table 5.2 illustrates the training time of each model in hours. It shows that by targeting the MC energy estimator, EnDEM can achieve faster convergence rate than iDEM, i.e. 0.53 hours v.s. 0.87 hours for GMM-40 and 1.85 hours v.s. 3.07 hours for DW-4, even though it requires additional differentiation of the neural network when simulating the amortized sampler. Though BEnDEM can be slower than EnDEM in our case, as it requires evaluating $K = 400$ times of neural network to compute the energy at bootstrapped time s and evaluating the system energy $\mathcal{E}$ is faster, it serves the same fast convergance-rate as EnDEM and results in faster convergence compared with iDEM, i.e. 0.60 v.s. 0.87 for GMM-40 and 2.83 v.s.

[1] https://github.com/jarridrb/DEM

3.07 for DW-4. It's noticeable that in higher-dimensional and more complex tasks, evaluating the system energy can be much slower than evaluating a neural network, which means that BEnDEM can be faster than EnDEM in training while being capable to achieve better performance. However, the ablation models, DEM-En and BDEM-En, can be slow, as they require differentiating the neural network as well as the system energy function (i.e. $\nabla\mathcal{E}$) during training, while the instability also leads to difficulty in convergence.

Dataset $\downarrow$ Algorithm $\rightarrow$	EnDEM	BEnDEM	DEM-En	BDEM-En	iDEM (rerun)
GMM-40 ($d = 2$)	0.53	0.60	0.58	0.45	0.87
DW-4 ($d = 8$)	1.85	2.83	2.22	4.08	3.07

Table 5.2 Training time ($\downarrow$) results in hours excluding evaluation time.

5.7 Summary

In this chapter, we introduce the datasets, metrics, and settings for model evaluation. The experiment results show that, by learning a less noisy target, EnDEM can perform better than iDEM, while achieving faster convergence rate in training. Furthermore, BEnDEM can further boost performance of EnDEM, achieving the best results among all models in most metrics. Also, we notice that the ablation model, BDEM-En, is not only working worse on most metrics but is also limited to scaling up to higher dimensional datasets, revealing the necessity of targeting the energy directly when using a bootstrap estimator for variance reduction.

Chapter 6

Conclusions

In this thesis, we propose EnDEM and BEnDEM. The former one is a novel method for sampling from Boltzmann-type distribution by learning annealed (time-convolved) energies and leveraging diffusion models for efficient data generation. Compared previous DEM proposed by Akhound-Sadegh et al. (2024), which inspires our work, EnDEM is targeting a less variance estimator and achieves better performance. Though EnDEM requires differentiating the neural network to compute scores, while DEM only requires forward passes, the learned annealed energies allow us to further improve sampler performance. One way is to enhance the learning objective via bootstrapping energy estimation on top of EnDEM, leading to our second method BEnDEM. By using the well-learned energies at low noise levels to reduce the variance of estimators at higher noise levels, BEnDEM can be better than EnDEM and has the potential to outperform EnDEM and DEM more in more complex tasks. Above all, learning the annealed energies instead of scores brings a lot of advantages for sampling from a Boltzmann-type distribution, which, we believe, is a valuable direction for a better neural Boltzmann sampler.

6.1 Limitation

The first limitation of our work is the more expensive time of sampling, compared with DEM (Akhound-Sadegh et al., 2024), which requires differentiating the energy network, i.e. $\nabla E_\theta(x,t)$. It can be very expensive when the dimensionality of data increases, like LJ-55 or even more complex tasks. A possible remedy is via distillation, which reduces the number of neural network evaluations from thousands to few or even one (Salimans and Ho, 2022; Yin et al., 2024). However, the distillation-based method can only speed up the sampling process after training. While the inner-loop in EnDEM training can be still expensive.

Another limitation is the difficulty of learning from sharp energy landscapes. In high dimensional cases, both the energies and scores can be very sharp with extreme values in some regions, resulting in instability of model training. A solution for DEM is clipping the MC score estimators, leading to learning a smoothed energy landscape. For EnDEM and BEnDEM, however, we target the annealed energies and thus cannot simply clip the energy estimators (otherwise it leads to 0 scores). One solution can be to smooth the energy landscape explicitly, e.g. $\tilde{\mathcal{E}}(x) = \mathcal{E}(x)/(1+\mathcal{E}(x))^{\alpha}$.

6.2 Future work

Our future work includes addressing the aforementioned limitations as well as further utilizing the learned energies for higher sample quality.

Efficient EnDEM sampling. The main bottleneck of EnDEM sampling is the differentiation over neural networks. A potential way to solve it is to align the nature of our Boltzmann-type target as well as Tweedie's formula (2.50). In such a way, the scores at (x,t) can be estimated by evaluating the energy network at data points around x, which is shown in Appendix E. Due to the limited schedule of this thesis, we leave its experiments as future work.

On the other hand, we can speed up the sampling phase by using finite steps, e.g. 10. Diffusion Recovery Likelihood (DRL; Gao et al. (2021)) provides us a possible way to do that, which learns the annealed energies at discrete times and sample data at $t-1$ from t by simulating a Langevin Dynamics.

Learning a sharp energy landscape. When the dimensionality of data is high and their modes are separate, the energy landscape can be very sharp. Rather than explicitly smooth energy, a potential way is using contrastive loss (Khosla et al., 2020), which implicitly learns from a smoothed energy landscape (Du et al., 2024; Kim and Ye, 2022). Another way can be combing a Fourier neural network framework, which factorizes the energy landscapes into different frequency domains (Tancik et al., 2020; Wang et al., 2023).

Exploration-exploitation trade-off: combing denoising score matching. The framework of both DEM and EnDEM can be viewed as a reinforcement learning algorithm, which starts from learning targets at initial guess (samples from prior) and iteratively refines the energies or scores by learning from what it generates. A potential improvement can be dynamically combining a DSM target during training, which encourages more exploitation. When the neural sampler starts to generate reasonable samples and the buffer stores samples from the target distribution, we can combine a DSM loss into the training target, which utilizes the information of scores given by the buffer.

EnDEM+MCMC sampling. The advantage of learning annealed energies is not limited to

training targets with smaller variance. Furthermore, EnDEM allows us to evaluate these time-convolved energies efficiently, providing a possibility of embedding a Metropolis-Hastings step in the reverse-SDE integration which resembles a neural Metropolis-Adjusted Langevin Algorithm (MALA), see Appendix F.

Scaling up to larger datasets. DEM, EnDEM, and Bootstrap EnDEM are valuable methods for sampling from a Boltzmann-type distribution as they avoid computationally expensive methods like MD and MCMC by leveraging diffusion sampling. The nature of these methods allows them to scale up to high-dimensional data for more complex datasets than GMM, DW-4, and LJ-n. Therefore, scaling these methods to higher dimensional ones is of interest in the future.

References

Akhound-Sadegh, T., Rector-Brooks, J., Bose, A. J., Mittal, S., Lemos, P., Liu, C.-H., Sendera, M., Ravanbakhsh, S., Gidel, G., Bengio, Y., Malkin, N., and Tong, A. (2024). Iterated denoising energy matching for sampling from boltzmann densities.

Allen, M. P. and Tildesley, D. J. (2017). *Computer Simulation of Liquids*. Oxford University Press.

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Arts, M., Satorras, V. G., Huang, C.-W., Zuegner, D., Federici, M., Clementi, C., Noé, F., Pinsler, R., and van den Berg, R. (2023). Two for one: Diffusion models and force fields for coarse-grained molecular dynamics.

Barrat, J.-L. and Hansen, J.-P. (2003). *Brownian motion, diffusion and the Langevin equation*, page 228–256. Cambridge University Press.

Berner, J., Richter, L., and Ullrich, K. (2024). An optimal control perspective on diffusion-based generative modeling.

Betancourt, M. (2018). A conceptual introduction to hamiltonian monte carlo.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Du, Y., Mao, J., and Tenenbaum, J. B. (2024). Learning iterative reasoning through energy diffusion.

Durumeric, A. E. P., Chen, Y., Noé, F., and Clementi, C. (2024). Learning data efficient coarse-grained molecular dynamics from forces and noise.

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

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

Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. (2021). Learning energy-based models by diffusion recovery likelihood.

Gee, J. C., Alsop, D. C., and Aguirre, G. K. (1997). Effect of spatial normalization on analysis of functional data. In Hanson, K. M., editor, *Medical Imaging 1997: Image Processing*, volume 3034, pages 550 – 560. International Society for Optics and Photonics, SPIE.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. *arXiv preprint arXiv:2203.17003*.

Hsu, T., Sadigh, B., Bulatov, V., and Zhou, F. (2024). Score dynamics: scaling molecular dynamics with picoseconds timestep via conditional diffusion model.

Hyvärinen, A. (2007). Some extensions of score matching. *Computational Statistics & Data Analysis*, 51(5):2499–2512.

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Kim, B. and Ye, J. C. (2022). Energy-based contrastive learning of visual representations. *Advances in Neural Information Processing Systems*, 35:4358–4369.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Klein, L., Krämer, A., and Noe, F. (2023). Equivariant flow matching. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 59886–59910. Curran Associates, Inc.

Köhler, J., Klein, L., and Noe, F. (2020). Equivariant flows: Exact likelihood generative learning for symmetric densities. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5361–5370. PMLR.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2020). Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Lazim, R., Suh, D., and Choi, S. (2020). Advances in molecular dynamics simulations and enhanced sampling methods for the study of protein systems. *International journal of molecular sciences*, 21(17):6339.

Leimkuhler, B. and Matthews, C. (2012). Rational construction of stochastic numerical methods for molecular sampling. *Applied Mathematics Research eXpress*.

Luhman, S. and Luhman, T. (2021). Knowledge distillation in iterative generative models for fast sampling. *arXiv preprint arXiv:2101.02388*.

Midgley, L. I., Stimper, V., Simm, G. N. C., Schölkopf, B., and Hernández-Lobato, J. M. (2023). Flow annealed importance sampling bootstrap.

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

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

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

Owen, A. B. (2023). *Practical Quasi-Monte Carlo Integration*. https://artowen.su.domains/mc/practicalqmc.pdf.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of langevin distributions and their discrete approximations.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2021). Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*.

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

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models.

Sarmadi, H., Entezami, A., and Magalhães, F. (2023). Unsupervised data normalization for continuous dynamic monitoring by an innovative hybrid feature weighting-selection algorithm and natural nearest neighbor searching. *Structural Health Monitoring*, 22(6):4005–4026.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2022). E(n) equivariant graph neural networks.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. *arXiv preprint arXiv:2303.01469*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

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

Song, Y. and Kingma, D. P. (2021). How to train your energy-based models.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

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

Tong, A., Fatras, K., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Wolf, G., and Bengio, Y. (2024). Improving and generalizing flow-based generative models with minibatch optimal transport.

Tzen, B. and Raginsky, M. (2019). Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit.

Vargas, F., Grathwohl, W., and Doucet, A. (2023). Denoising diffusion samplers.

Wang, Y., Shen, Y., Chen, S., Wang, L., Ye, F., and Zhou, H. (2023). Learning harmonic molecular representations on riemannian manifold. *arXiv preprint arXiv:2303.15520*.

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. (2024). One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6613–6623.

Zhang, Q. and Chen, Y. (2022). Path integral sampler: a stochastic control approach for sampling.

Appendix A

Supplementary maths

A.1 Optimization of neural network

Suppose we are training a neural network $f_\theta(x)$ to target noisy observations $y(x) = f(x) + \varepsilon(x)$ using a l_2 loss, where $x \in \mathbb{R}^d$ and $\varepsilon(x) = [\varepsilon_1(x), ..., \varepsilon_d(x)]^T$ is a random function with 0 mean (i.e. $\mathbb{E}\varepsilon(x) = \mathbf{0}, \forall x$). Then the optimal network f_θ^* is obtained by minimizing $\mathcal{L}(\theta) := \mathbb{E}\|f_\theta(x) - y(x)\|_2^2$:

$$f_\theta^* = \arg\min_{f_\theta} \mathbb{E}\|f_\theta(x) - y(x)\|_2^2 \tag{A.1}$$

$$= \arg\min_{f_\theta} \mathbb{E}\left[\|f_\theta(x) - f(x)\|^2 - 2\varepsilon^T(x)(f_\theta(x) - f(x)) + \|\varepsilon(x)\|^2\right] \tag{A.2}$$

$$= \arg\min_{f_\theta} \mathbb{E}\|f_\theta(x) - f(x)\|^2 + \mathbb{E}\|\varepsilon(x)\|^2 \tag{A.3}$$

$$= \arg\min_{f_\theta} \mathbb{E}\|f_\theta(x) - f(x)\|^2 \tag{A.4}$$

$$= f \tag{A.5}$$

the optimal f_θ is f, resulting to the optimal loss:

$$\mathcal{L}(\theta^*) = \mathbb{E}\|f_{\theta^*}(x) - y(x)\|_2^2 \tag{A.6}$$

$$= \mathbb{E}\|\varepsilon(x)\|_2^2 \tag{A.7}$$

$$= \mathbb{E}_x\left[\sum_{i=1}^{d} \mathrm{Var}(\varepsilon_i(x))\right] := \sum_{i=1}^{d} \sigma_i^2 \tag{A.8}$$

where σ_i^2 is the expected variance of the i^{th} element of $\varepsilon(x)$ over x, i.e. $\sigma_i^2 = \mathbb{E}_x[\mathrm{Var}(\varepsilon_i(x))]$. Therefore, given a powerful optimizer and sufficient data, a perfect neural network can

minimize the expected loss to a minimum, which is the sum of the element-wise expected variance of the noise.

A.2 Basic Concentration Inequalities

A.2.1 Sub-Gaussian and its concentration

A random variable X is sub-Gaussian, if there exists a constant $c > 0$ such that:

$$\mathbb{E}[e^{\lambda X}] \leq e^{\frac{\lambda^2 c^2}{2}}, \quad \forall \lambda \in \mathbb{R} \tag{A.9}$$

Given a sub-Gaussian random variable, X, with mean m and variance v, it has the following concentration inequality:

$$P(|X - m| \geq \varepsilon) \leq 2e^{-\frac{\varepsilon^2}{2v}} \tag{A.10}$$

Then, with probability $1 - \delta$, we have:

$$|X - m| \leq \sqrt{2v \log \frac{2}{\delta}} = C\sqrt{\log \frac{2}{\delta}} \tag{A.11}$$

where $C = \sqrt{2v}$.

A.2.2 Bounded random variable and its concentration

If a random variable X is bounded by $M > 0$, i.e. $|X| \leq M$, then we have $\mathbb{E}[e^{\lambda X}] \leq e^{\lambda M}$ and $\mathbb{E}[e^{\lambda X}] \leq e^{-\lambda M}$, which is equivalent to:

$$\mathbb{E}[e^{\lambda X}] \leq e^{|\lambda M|} = e^{\frac{\lambda^2 M^2}{2}}, \quad \forall \lambda \in \mathbb{R} \tag{A.12}$$

Thus, a bounded r.v. X is sub-Gaussian.

Appendix B

Proofs in EnDEM and BEnDEM

B.1 Probability Error Bound for MC Score Estimator

B.1.1 For MC Score Estimator

Let $Z_K(x_t,t) := \frac{1}{K}\sum_{i=1}^{K} z_{0t}^{(i)}(x_t)$, $z_{0t}^{(i)}(x_t) = \exp(-\mathcal{E}_0(x_{0|t}^{(i)}))$. And

$$m_{0t}(x_t) = \mathbb{E}[z_{0t}(x_t)] = \exp(-\mathcal{E}_t(x_t)) \tag{B.1}$$

$$v_{0t}(x_t) = \mathrm{Var}(z_{0t}(x_t)) \tag{B.2}$$

Assume that $z_{0t}(x_t)$ is sub-Gaussian. Then Z_K is sub-Gaussian with $\mathbb{E}[Z_K(x_t,t)] = m_{0t}(x_t)$ and $\mathrm{Var}(Z_K(x_t,t)) = \frac{v_{0t}(x_t)}{K}$. Thus, we have the following concentration for Z:

$$P(|Z_K(x_t,t) - \mathbb{E}Z_K(x_t,t)| \geq \varepsilon) \leq 2e^{-\frac{\varepsilon^2}{2v_{0t}(x_t)/K}} \tag{B.3}$$

we thus can show the constant C in Akhound-Sadegh et al. (2024) is $C = \sqrt{2v_{0t}(x_t)}$, and their $c(x_t)$ is therefor:

$$c(x_t) = \frac{2\sqrt{2v_{0t}(x_t)\frac{\log(2/\delta)}{\log(1/\delta)}}(1 + \|\nabla\mathcal{E}_t(x_t)\|)}{m_{0t}(x_t)\sqrt{K}} \tag{B.4}$$

B.2 Properties of MC Energy Estimator

B.2.1 logarithm of sub-Gaussian r.v. and its concentration inequality

Now, let's consider an estimator $\log Z$, where Z is an unbiased estimator with mean m and variance v. By second-order Taylor expansion around $m = \mathbb{E}Z$:

$$\log Z \approx \log m + \frac{1}{m}(Z-m) - \frac{1}{2m^2}(Z-m)^2 + O(\|Z-m\|^2) \tag{B.5}$$

Under the sub-Gaussian assumption, $O(\|Z-m\|^2)$ is negligible. Then,

$$\mathbb{E}\log Z \approx \log m - \frac{1}{2m^2}\mathbb{E}[(Z-m)^2] \tag{B.6}$$

$$= \log m - \frac{v}{2m^2} \tag{B.7}$$

$$\mathrm{Var}(\log Z) \approx \frac{1}{m^2}\mathrm{Var}(Z) + \frac{1}{4m^2}\mathrm{Var}((Z-m)^2) \tag{B.8}$$

$$= \frac{v}{m^2} \tag{B.9}$$

Let's consider the concentration of $\log Z$ now:

$$P(|\log Z - \mathbb{E}\log Z| \geq \varepsilon) = P(|\frac{Z-m}{m}| \geq \varepsilon) \text{ (by first-order Taylor expansion)} \tag{B.10}$$

$$= P(|Z-m| \geq \varepsilon m) \tag{B.11}$$

$$\leq 2e^{-\frac{\varepsilon^2 m^2}{2v}} \tag{B.12}$$

thus, with probability $1-\delta$:

$$|\log Z - \mathbb{E}\log Z| \leq \sqrt{2\frac{v}{m^2}\log\frac{2}{\delta}} \tag{B.13}$$

B.2.2 Variance, Bias and Concentration Inequality for MC Energy Estimator

Our MC energy estimator can be defined as $E_K(x_t,t) = \log Z_K(x_t,t)$, where $Z_K(x_t,t)$ is an unbiased estimator with mean $m_{0t}(x_t)$ and variance $v_{0t}(x_t)$. Substituting relevant terms in Equations B.7, B.8, and B.13, respective, gives us the bias, variance, and concentration

inequality for E_K as follows:

$$\mathbb{E}E_K(x_t,t) \approx -\log m_{0t}(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K} \tag{B.14}$$

$$\mathrm{Var}(E_K(x_t,t)) \approx \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K} \tag{B.15}$$

$$|E_K(x_t,t) - \mathbb{E}[E_K(x_t,t)]| \leq \sqrt{2\frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K}\log\frac{2}{\delta}} \tag{B.16}$$

B.3 Proof of Proposition 3

Proposition 3 *(ours) If* $\exp(-\mathcal{E}(x_0^{(i)}|t))$ *is sub-Gaussian, then with probability* $1-\delta$ *(over* $x_{0|t}^{(i)} \sim \mathcal{N}(x_t,\sigma_t^2)$*) we have*

$$\|E_K(x_t,t) - \mathcal{E}_t(x_t)\| \leq \frac{\sqrt{2v_{0t}(x_t)\log\frac{2}{\delta}}}{m_{0t}(x_t)\sqrt{K}} \tag{B.17}$$

□**Proof.** First consider an estimator $\log Z$ defined in B.2.1. Using the triangle inequality and Z's concentration inequality (B.13), we can have:

$$|\log Z - \log \mathbb{E}Z| = |\log Z - \mathbb{E}\log Z + \mathbb{E}\log Z - \log \mathbb{E}Z| \tag{B.18}$$

$$\leq |\log Z - \mathbb{E}\log Z| + |\mathbb{E}\log Z - \log \mathbb{E}Z| \text{ (by triangle inequality)} \tag{B.19}$$

$$\leq \sqrt{2\frac{v}{m^2}\log\frac{2}{\delta}} + |\log m + \frac{v}{2m^2} - \log m| \text{ (by B.7 and B.13)} \tag{B.20}$$

$$= \sqrt{2\frac{v}{m^2}\log\frac{2}{\delta}} + \frac{v}{2m^2} \tag{B.21}$$

Now, by substituting Z with $Z_K(x_t,t)$ and plugging its mean (B.14) and variance (B.15), the bias of our MC energy estimator, $E_K(x_t,t) - \mathcal{E}_t(x_t)$, can be expressed by:

$$|E_K(x_t,t) - \mathcal{E}_t(x_t)| \leq \sqrt{2\frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K}\log\frac{2}{\delta}} + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K} \tag{B.22}$$

$$= \frac{\sqrt{2v_{0t}(x_t)\log\frac{2}{\delta}}}{m_{0t}(x_t)\sqrt{K}} + O(1/K) \tag{B.23}$$

□

B.4 Variance of MC Score Estimator

A MC score estimator is defined as $S_K(x_t,t) := \nabla E_K(x_t,t) = \nabla \log Z_K(x_t,t)$, with assumptions that $\exp(-\mathcal{E}(x))$ and $\|\nabla_{x_t}\exp(-\mathcal{E}(x))\|$ are sub-Gaussian where $x \sim N(x;x_t,\sigma_t^2)$ (i.e. $x = x_t + \sigma_t\varepsilon$, $\varepsilon \sim N(0,I)$).

MC score estimator as an importance-weighted estimate (Akhound-Sadegh et al., 2024). Equation 2.61 can be rewritten as:

$$S_K(x_t,t) = \nabla_{x_t}\log\frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(x_t+\sigma_t\varepsilon^{(i)})) \tag{B.24}$$

$$= \frac{\frac{1}{K}\sum_{i=1}^{K}\nabla\exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)}))} \tag{B.25}$$

$$= -\frac{\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)}))\nabla\mathcal{E}(x_{0|t}^{(i)})}{\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)}))} \tag{B.26}$$

$$= \frac{\sum_{i=1}^{K} w_i f_i}{\sum_{i=1}^{K} w_i} \tag{B.27}$$

where $w_i = \exp(-\mathcal{E}(x_{0|t}^{(i)}))$ and $f_i = -\nabla\mathcal{E}(x_{0|t}^{(i)})$. Therefore, S_K can be expressed as an estimator by self-normalized importance sampling, with (unnormalized) importance weight w_i.

To derive the variance of S_K, we break it into two parts: one-dimensional data and high-dimensional one. For high-dimensional data, we derive the variance of each entry of S_K.

Different from Akhound-Sadegh et al. (2024), we assume that the norm of $\nabla\exp(-\mathcal{E}(x))$ is bounded by $M > 0$, i.e. $\|\nabla\exp(-\mathcal{E}(x))\| \le M$ instead of a sub-Gaussian assumption directly. It is usually the case since we'll clip the norm of S_K to a certain level in practice for stable training. First notice that $\|\nabla\exp(-\mathcal{E}(x))\|$ is a bounded random variable. By A.2.2, $\|\nabla\exp(-\mathcal{E}(x))\|$ is sub-Gaussian which aligns previous assumption by Akhound-Sadegh et al. (2024). While

$$|\nabla\exp(-\mathcal{E}(x))[i]| \le \sqrt{|\nabla\exp(-\mathcal{E}(x))[i]|^2} \le \sqrt{\sum_i |\nabla\exp(-\mathcal{E}(x))[i]|^2} = \|\nabla\exp(-\mathcal{E}(x))\| \le M \tag{B.28}$$

, the i^{th} entry of $\nabla\exp(-\mathcal{E}(x))$ is also bounded by M as well, which means $\nabla\exp(-\mathcal{E}(x))[i]$ is also sub-Gaussian. And we further assume that the energy function $\mathcal{E}$ is positive.

B.4.1 For $\mathbb{R}^1$ data

We first assume x is one-dimensional, i.e. $x \in \mathbb{R}^1$. In this case, $\nabla \exp(-\mathcal{E}(x))$ and $|\nabla \exp(-\mathcal{E}(x))|$ are both sub-Gaussian.

In low-energy regions. $\exp(-\mathcal{E}(x))$ is concentrated away from 0 as $\mathcal{E}(x)$ is small. Since $\exp(-\mathcal{E}(x_{0|t}^{(i)}))$ are sub-Gaussian, mean of sub-Gaussian random variables, $\frac{1}{K}\sum_i \exp(-\mathcal{E}(x_{0|t}^{(i)}))$, is also sub-Gaussian concentrated on the same mean $m_{0t}(x_t) > 0$. Then, there exists a constant c such that $\exp(-\mathcal{E}(x_{0|t}^{(i)})) \geq c > 0$ and thus:

$$\left| \frac{\frac{1}{K}\sum_{i=1}^{K} \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))}{\frac{1}{K}\sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \right| \leq \left| \frac{\frac{1}{K}\sum_{i=1}^{K} \nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))}{c} \right| \tag{B.29}$$

And since $\nabla \exp(-\mathcal{E}(x_{0|t}^{(i)}))$ is sub-Gaussian, then ration B.29 will be sub-Gaussian. On the other hand, we have the probability error bound of S_K (3.11), while this can be related to the variance of S_K when S_K is sub-Gaussian:

$$\mathrm{Var}(S_K(x_t,t)) = \frac{4v_{0t}(x_t)(1+\nabla\mathcal{E}_t(x_t))^2}{m_{0t}^2(x_t)K} > \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K} = \mathrm{Var}(E_K(x_t,t)) \tag{B.30}$$

In high-energy regions. It's reasonable to assume that a high-energy region always relates to a steep vector field, i.e. $\|\nabla\mathcal{E}(x)\|^2$ is large. Therefore, we can assume that $\exp(-\mathcal{E}(x))$ and $-\nabla\mathcal{E}(x)$ are positively related in high energy region, i.e. their covariance is positive. According to Section 9.2 in Owen (2023), the asymptotic variance of a self-normalized importance sampling estimator is given by:

$$\mu = \mathbb{E}_q[f(X)] \tag{B.31}$$

$$\tilde{\mu}_q = \frac{\sum_{i=1}^{K} w_i f_i}{\sum_{i=1}^{K} w_i} \tag{B.32}$$

$$\mathrm{Var}(\tilde{\mu}_q) \approx \frac{1}{K}\mathbb{E}_q[w(X)]^{-2}\mathbb{E}_q[w(X)^2(f(X)-\mu)^2] \tag{B.33}$$

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

positive related, we have:

$$\mathrm{Var}(S_K(x_t,t)) \geq \frac{1}{K}\mathbb{E}_q[w(X)]^{-2}\mathbb{E}_q[w^2(X)]\mathbb{E}_q[(f(X)-\mu)^2] \tag{B.34}$$

$$= \frac{v_{0t}(x_t)+m_{0t}^2(x_t)}{m_{0t}^2(x_t)K}\mathrm{Var}_q(\nabla\mathcal{E}(x)) \tag{B.35}$$

$$> \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K} = \mathrm{Var}(E_K(x_t,t)) \tag{B.36}$$

B.4.2 For $\mathbb{R}^d$ data

Now we consider higher dimensional data, say d-dimensional. Recap that $\nabla\exp(-\mathcal{E}(x))[i]$ and $\|\nabla\exp(-\mathcal{E}(x))\|$ are all sub-Gaussian. We further assume that the elements of S_K are independent, i.e. $\mathrm{Cov}(S_K(x_t,t))$ is diagonal.

In low-energy regions. We can generalize the previous analysis element-wise and derive that there exists a constant c such that $\exp(-\mathcal{E}(x_{0|t}^{(i)})) \geq c > 0$ and for each element j,

$$\left\| \frac{\frac{1}{K}\sum_{i=1}^K \nabla\exp(-\mathcal{E}(x_{0|t}^{(i)}))[j]}{\frac{1}{K}\sum_{i=1}^K \exp(-\mathcal{E}(x_{0|t}^{(i)}))} \right\| \leq \left\| \frac{\frac{1}{K}\sum_{i=1}^K \nabla\exp(-\mathcal{E}(x_{0|t}^{(i)}))[j]}{c} \right\| \leq \frac{M}{c} \tag{B.37}$$

for $j = 1,\ldots,d$, are all sub-Gaussian. While Inequality 3.11 can be expressed as:

$$\sqrt{\sum_{j=1}^d (\mathcal{S}_K(x_t,t)[j]-S_t(x_t)[j])^2} \leq \frac{2\sqrt{2v_{0t}(x_t)\log(\frac{2}{\delta})}(1+\|\nabla\mathcal{E}_t(x_t)\|)}{m_{0t}(x_t)\sqrt{K}} \tag{B.38}$$

And we can roughly assume an element-wise error bound as follows:

$$|\mathcal{S}_K(x_t,t)[j]-S_t(x_t)[j]| \leq \frac{2\sqrt{2v_{0t}(x_t)\log(\frac{2}{\delta})}(1+\|\nabla\mathcal{E}_t(x_t)\|)}{m_{0t}(x_t)\sqrt{Kd}} \tag{B.39}$$

which suggests that we can approximate the variance of the j^{th} entry of S_K by:

$$\mathrm{Var}(S_K(x_t,t)[j]) = \frac{4v_{0t}(x_t)(\frac{1+\|\nabla\mathcal{E}_t(x_t)\|}{\sqrt{d}})^2}{m_{0t}^2(x_t)K} \tag{B.40}$$

then

$$\sum_{j=1}^{d} \mathrm{Var}(S_K(x_t,t)[j]) = \frac{4v_{0t}(x_t)(1+\|\nabla\mathcal{E}_t(x_t)\|)^2}{m_{0t}^2(x_t)K} > \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K} = \mathrm{Var}(E_K(x_t,t)) \tag{B.41}$$

In high-energy regions. It's usually the case 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 $\nabla\mathcal{E}(x)[j]$. For such j, we can derive the corresponding variance of $S_K(x_t,t)[j]$ similar to Inequality B.34:

$$\mathrm{Var}(S_K(x_t,t)[j]) \geq \frac{1}{K}\mathbb{E}_q[w(X)]^{-2}\mathbb{E}_q[w^2(X)]\mathbb{E}_q[(f(X)[j]-\mu[j])^2] \tag{B.42}$$

$$= \frac{v_{0t}(x_t)+m_{0t}^2(x_t)}{m_{0t}^2(x_t)K}\mathrm{Var}_q(\nabla\mathcal{E}(x)[j]) \tag{B.43}$$

$$> \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K} = \mathrm{Var}(E_K(x_t,t)) \tag{B.44}$$

B.4.3 Conclusion

Above all, we can always have:

$$\mathrm{Var}(E_K(x_t,t)) < \sum_{j=1}^{d} \mathrm{Var}(S_K(x_t,t)[j]) \tag{B.45}$$

referring to the MC energy estimator has a smaller variance and can provide a more useful and less noisy training signal. In terms of optimization (see Appendix A.1), one can learn a perfect energy network E_{θ^*} with optimal loss $\mathbb{E}_{t,x_0,x_t}[\mathrm{Var}(E_K(x_t,t))]$, which is smaller than that of an optimal score network $\mathbb{E}_{t,x_0,x_t}[\sum_{j=1}^{d}\mathrm{Var}(S_K(x_t,t)[j])]$.

B.5 Proof of Proposition 4

Proposition 4 *(ours) 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 $x_{0|t}\sim\overline{\mathcal{N}}(x_{0|t};x_t,\sigma_t^2)$ is G-invariant, i.e.,*

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

□**Proof.** 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

$$E_K(g\circ x_t,t) = -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(g\circ x_{0|t}^{(i)})) \tag{B.46}$$

$$= -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(i)})) = E_K(x_t,t) \tag{B.47}$$

$$x_{(0|t)}^{(i)} \sim \overline{\mathcal{N}}(x_{0|t};x_t,\sigma_t^2) \tag{B.48}$$

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

B.6 Bias of Bootstrap Energy Estimator

B.6.1 Bootstrap(1) estimator

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

$$E_K^{\mathrm{Seq}}(x_t,t) := -\log\frac{1}{K}\sum_{i=1}^{K}\exp(-E_K(x_{s|t}^{(i)},s)),\quad x_{s|t}^{(i)}\sim\mathcal{N}(x;x_t,(\sigma_t^2-\sigma_s^2)I) \tag{B.49}$$

$$= -\log\frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\exp(-\mathcal{E}(x_{0|t}^{(ij)})),\quad x_{0|t}^{(ij)}\sim\mathcal{N}(x;x_t,\sigma_t^2 I) \tag{B.50}$$

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

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

$$\mathbb{E}[E_K^{\mathrm{Seq}}(x_t,t)] = \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^2} \quad\text{and}\quad \mathrm{Var}(E_K^{\mathrm{Seq}}(x_t,t)) = \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K^2} \tag{B.52}$$

While an optimal network obtained by targeting the original MC energy estimator 3.3 at s is:

$$E_{\phi^*}(x_s,s) = \mathbb{E}[E_K(x_s,s)] = -\log m_{0s}(x_s) + \frac{v_{0s}(x_s)}{2m_{0s}^2(x_s)K} \tag{B.53}$$

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

$$E_K^{B(1)}(x_t,t,s;\phi^*) = -\log\frac{1}{K}\sum_{i=1}^{K}\exp\left(-\left(-\log m_{0s}(x_{s|t}^{(i)}) + \frac{v_{0s}(x_{s|t}^{(i)})}{2m_{0s}^2(x_{s|t}^{(i)})K}\right)\right) \tag{B.54}$$

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

$$\log \frac{1}{K}\sum_{i=1}^{K}\exp(-(a_i+b_i)) = \log \frac{1}{K}\left\{\sum_{i=1}^{K}\exp(-a_i)\left[\frac{\sum_{i=1}^{K}\exp(-(a_i+b_i))}{\sum_{i=1}^{K}\exp(-a_i)}\right]\right\} \tag{B.55}$$

$$= \log \frac{1}{K}\sum_{i=1}^{K}\exp(-a_i) + \log \frac{\sum_{i=1}^{K}\exp(-(a_i+b_i))}{\sum_{i=1}^{K}\exp(-a_i)} \tag{B.56}$$

$$\approx \log \frac{1}{K}\sum_{i=1}^{K}\exp(-a_i) + \log \frac{\sum_{i=1}^{K}\exp(-a_i)(1-b_i)}{\sum_{i=1}^{K}\exp(-a_i)} \tag{B.57}$$

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

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

$$\approx \log \frac{1}{K}\sum_{i=1}^{K}\exp(-a_i) - m_b \tag{B.60}$$

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 $\sigma_t^2 - \sigma_s^2$ is small[1], $\{\frac{v_{0s}(x_{s|t}^{(i)})}{2m_{0s}^2(x_{s|t}^{(i)})K}\}_{i=1}^K$ are close to 0 and concentrated at $\frac{v_{0s}(x_t)}{2m_{0s}^2(x_t)K}$. Therefore, by plugging them into Equation B.60, Equation B.54 can be approximated by

$$E_K^{B(1)}(x_t,t,s;\phi^*) \approx -\log \frac{1}{K}\sum_{i=1}^{K} m_{0s}(x_{s|t}^{(i)}) + \frac{v_{0s}(x_t)}{2m_{0s}^2(x_t)K} \tag{B.61}$$

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

$$-\log \frac{1}{K}\sum_{i=1}^{K} m_{0s}(x_{s|t}^{(i)}) \approx -\log \frac{1}{K}\sum_{i=1}^{K} E_K(x_{s|t}^{(i)},s) = E_K^{\text{Seq}}(x_t,t) \tag{B.62}$$

[1]It is often the case since we have an hyperparameter β to control that $\sigma_t^2 - \sigma_s^2 \le \beta \forall t \in [t_n, t_{n+1}], s \in [t_{n-1}, t_n]$.

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

$$E_K^{B(1)}(x_t,t,s;\phi^*) \approx E_K^{\text{Seq}}(x_t,t) + \frac{v_{0s}(x_t)}{2m_{0s}^2(x_t)K} \tag{B.63}$$

where its mean and variance depend on those of the Sequential estimator (B.52):

$$\mathbb{E}[E_K^{B(1)}(x_t,t,s;\phi^*)] = \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^2} + \frac{v_{0s}(x_t)}{2m_{0s}^2(x_t)K} \tag{B.64}$$

$$\text{Var}(E_K^{B(1)}(x_t,t,s;\phi^*)) = \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K^2} \tag{B.65}$$

B.6.2 Bootstrap(n) estimator

Definition 2 *Given time splits* $0 = t_0 < t_1 < ...t_{n+1} < 1$ *and* $E_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}$. *Given* $n \in 1,...,N$, *for* $\forall (x_t,t,s) \in \mathbb{R} \times [t_n,t_{n+1}] \times [t_{n-1},t_n]$, *a Bootstrap(n) estimator is defined as:*

$$E_K^{B(n)}(x_t,t,s;\theta) := -\log \frac{1}{K}\sum_{i=1}^{K} \exp(-E_\theta(x_{s|t}^{(i)},s)) \tag{B.66}$$

where $x_{s|t}^{(i)} \sim \mathcal{N}(x;x_t,(\sigma_t^2-\sigma_s^2)I)$ *and with initial condition*

$$E_K^{B(0)}(x_t,t,s;\theta) := E_K(x_t,t), \forall \theta \in \Theta, \forall s \in \mathbb{R} \tag{B.67}$$

Proposition 5 *Given time splits* $0 = t_0 < t_1 < ... < t_N = 1$, $n \in \{0,...,N\}$ *and a neural network* E_θ. *For any fixed trajectory* $\{s_i\}_{i=0}^n$ *with* $s_i \in [t_{i-1},t_i]$ *and* $s_0 = 0$, *if* E_θ *is optimal for any* $u \le t_n$ *by sequentially optimising*

$$\theta^{(i)} = \arg\min_\theta \|E_\theta(x_u,u) - E_K^{B(i)}(x_u,u,s_i;\theta^{(i-1)})\|^2 \tag{B.68}$$

for $\forall i \in \{0,...,n-1\}$ *and* $\forall u \in [t_i,t_{i+1}]$, *then for* $\forall (x_t,t) \in \mathbb{R}^d \times [t_n,t_{n+1}]$, *the neural network optimized by targeting* $E_K^{B(n)}(x_t,t,s_n;\theta^{(n-1)})$ *has the optimal value:*

$$E_{\theta^{(n)}}(x_t,t) = \mathcal{E}_t(x_t) + \sum_{j=1}^{n+1} \frac{v_{0,s_j}(x_t)}{2m_{0,s_j}^2(x_t)K^j} \tag{B.69}$$

where $s_{n+1} = t$. Especially, when $n = 0$ it resembles the optimal values by targeting the MC energy estimator (3.3), i.e.

$$E_{\theta^{(0)}}(x_t, t) = \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K} \tag{B.70}$$

□**Proof.** The proof of Proposition 5 is by induction.

Let the energy network be optimal for $t \leq t_n$ by learning a sequence of Bootstrap(i) energy estimators ($i \leq n-1$). Then for any $s \in [t_{n-1}, t_n]$, 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:

$$E_{\theta^{(n-1)}}(x_s, s) = \mathcal{E}_s(x_s) + \sum_{j=1}^{n} \frac{v_{0,s_j}(x_s)}{2m_{0,s_j}^2(x_s)K^j} \tag{B.71}$$

where $\{s_i\}_{i=0}^n$ is a fixed trajectory for each $s_i \in [t_{i-1}, t_i]$, $s_0 = 0$ and $s_n = s$. Then for any $t \in [t_n, t_{n+1}]$, the learning target of $E_\theta(x_t, t)$ is bootstrapped from $s_n = s$,

$$E_K^{B(n)}(x_t, t) = -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-E_{\theta^{(n-1)}}(x_{s|t}^{(i)}, s)), \quad x_{s|t}^{(i)} \sim \mathcal{N}(x; x_t, (\sigma_t^2 - \sigma_s^2)I) \tag{B.72}$$

$$= -\log \frac{1}{K} \sum_{i=1}^{K} \exp\left(-\mathcal{E}_s(x_{s|t}^{(i)}) - \sum_{j=1}^{n} \frac{v_{0,s_j}(x_{s|t}^{(i)})}{2m_{0,s_j}^2(x_{s|t}^{(i)})K^j} \right) \tag{B.73}$$

Assume that $\sigma_t^2 - \sigma_s^2$ is small and K is large, then apply Approximation B.60:

$$E_K^{B(n)}(x_t, t) \approx -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}_s(x_{s|t}^{(i)})) + \sum_{j=1}^{n} \frac{v_{0,s_j}(x_{s|t}^{(i)})}{2m_{0,s_j}^2(x_{s|t}^{(i)})K^j} \tag{B.74}$$

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$(n+1)$ estimator which is equivalent to $E_{K^{n+1}}$,

$$E_K^{B(n)}(x_t,t) \approx E_{K^{n+1}}(x_t,t) + \sum_{j=1}^{n} \frac{v_{0,s_j}(x_{s|t}^{(i)})}{2m_{0,s_j}^2(x_{s|t}^{(i)})K^j} \tag{B.75}$$

$$\mathbb{E}[E_K^{B(n)}(x_t,t)] \approx \mathcal{E}_t(x_t) + \frac{v_{0t}(x_t)}{2m_{0t}^2(x_t)K^{n+1}} + \sum_{j=1}^{n} \frac{v_{0,s_j}(x_{s|t}^{(i)})}{2m_{0,s_j}^2(x_{s|t}^{(i)})K^j} \tag{B.76}$$

$$\mathrm{Var}(E_K^{B(n)}(x_t,t)) \approx \frac{v_{0t}(x_t)}{m_{0t}^2(x_t)K^{n+1}} \tag{B.77}$$

Therefore, by appending $s_{n+1} = t$ into the trajectory, the optimal value of the energy network at (x_t,t) can be expressed as:

$$E_{\theta^{(n)}}(x_t,t) = \mathcal{E}_t(x_t) + \sum_{j=1}^{n+1} \frac{v_{0,s_j}(x_t)}{2m_{0,s_j}^2(x_t)K^j} \tag{B.78}$$

□

Appendix C

AIS estimator for Energy and Score

In this section, we provide a fully detailed description of computing the AIS estimators, for both energy and score. A complete pseudo-code is provided in Algorithm 4, which assumes the Hamiltonian Monte Carlo simulator H is already given.

Algorithm 4 AIS estimator for Energy and Score

Require: Energy function $\mathcal{E}(x)$, Initial proposal $q(x) = N(x; x_t, \sigma_t^2 I)$, Intermediate targets $\tilde{\pi}_k$, Num. AIS steps L, HMC simulator $g(x, \pi)$, Num. MC samples K

1: $\{x_0^{(i)}\}_{i=1}^K \sim N(x; x_t, \sigma_t^2 I)$
2: $\log w_i \leftarrow -\mathcal{E}(x_0^{(i)})/L$
3: **for** k=1,...,L-1 **do**
4: $\quad x_k^{(i)} \leftarrow g(x_{k-1}^{(i)}, \tilde{\pi}_k)$ ▷ simulate HMC to generate sample from π_k
5: $\quad \log w_i \leftarrow \log w_i - \mathcal{E}(x_{k-1}^{(i)})/L$
6: **end for**
7: $E_K^{\mathrm{AIS}(L)} \leftarrow -\mathrm{LogSumExp}(\log w) + \log K$
8: $S_K^{\mathrm{AIS}(L)} \leftarrow -\nabla_{x_t} E_K^{\mathrm{AIS}(L)}$

Ensure: $E_K^{\mathrm{AIS}(L)}, S_K^{\mathrm{AIS}(L)}$

Notice that the intermediate targets $\tilde{\pi}_k$ is given in 3.24, then according to Equation 2.7, the Hamiltonian for $\tilde{\pi}_k$ is defined as:

$$H_k(x,p) = \mathcal{E}^{(k)}(x) + \frac{1}{2} p^T M^{-1} p \tag{C.1}$$

$$\mathcal{E}^{(k)}(x) := \frac{k}{L}\mathcal{E}(x) + \frac{1}{2\sigma_t^2}\|x - x_t\|_2^2 \tag{C.2}$$

where M is a given mass matrix. According to Section 2.2.1, we provide a complete description of HMC in Algorithm 5. Then the one for targeting $\tilde{\pi}_k$ can be obtained by plugging $\mathcal{E}^{(k)}$ as the energy function.

Algorithm 5 Hamiltonian Monte Carlo

Require: Energy function $\mathcal{E}(x)$, Initial samples $\{x^{(i)}\}_{i=1}^K$, Mass M, Num. steps T, Step size Δ

function: $H(x,p) = \mathcal{E}(x) + \frac{1}{2}p^T M^{-1} p$
$\{p^{(i)}\}_{i=1}^K \sim N(p;0,M)$
$t \leftarrow 0$
while $t < T$ **do**
    $p' \leftarrow p - \frac{\Delta}{2}\nabla\mathcal{E}(x)$
    $x' \leftarrow x + \Delta M^{-1} p'$
    $p' \leftarrow p - \frac{\Delta}{2}\nabla\mathcal{E}(x')$
    $\alpha(x',x) \leftarrow \min\left(1, \exp(H(x,p) - H(x',p'))\right)$
    with probability α: $x, p \leftarrow x', p'$ ▷ Metropolis-Hastings
    $t \leftarrow t + \Delta$
end while

Ensure: $\{x^{(i)}\}_{i=1}^K$

Appendix D

EnDEM for general SDEs

Diffusion models can be generalized to any SDEs with the following form for $t \geq 0$:

$$dx_t = f(x_t, t)dt + g(t)d\tilde{\mathbf{w}} \tag{D.1}$$

where $d\tilde{\mathbf{w}}$ is a standard Wienner process. Particularly, we consider $f(x,t) := -\alpha(t)x$, i.e.

$$dx_t = -\alpha(t)x_t dt + g(t)d\tilde{\mathbf{w}} \tag{D.2}$$

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

$$x_t = \beta(t)x_0 + \beta(t)\sqrt{\int_0^t (g(s)\beta(s))^2 ds}\varepsilon \tag{D.3}$$

$$\beta(t) = e^{-\int_0^t \alpha(s)ds} \tag{D.4}$$

where $\varepsilon \sim \mathcal{N}(0, I)$. 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 $\bar{\beta}_{\text{min}}$ to $\bar{\beta}_{\text{max}}$, the above SDE resembles a Variance Preserving (VP) process (Song et al., 2020b). 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 EnDEM and DEM to general SDEs.

In this chapter, we provide a solution for general SDEs (D.2) rather than a VE SDE (2.44). The derivation aligns the idea from Akhound-Sadegh et al. (2024), but we also notice that there are some typos and misleadings in their equation and we, therefore, repropose with our notations.

For simplification, we exchangeably use $\beta(t)$ and β_t. Given a SDE as Equation D.2 for any integrable functions α and g, we can first derive its marginal as Equation D.3, which can

be expressed as:

$$\beta_t^{-1} x_t = x_0 + \sqrt{\int_0^t (g(s)\beta(s))^2 ds}\varepsilon \tag{D.5}$$

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

$$y_t = y_0 + \sqrt{\int_0^t (g(s)\beta(s))^2 ds}\varepsilon \tag{D.6}$$

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

$$\begin{aligned} dy_t &= (\beta^{-1}(t))' x_t dt + \beta^{-1}(t) dx_t && \text{(D.7)} \\ &= \beta^{-1}(t)\alpha(t) x_t dt + \beta^{-1}(t)(-\alpha(t) x_t dt + g(t) d\tilde{\mathbf{w}}) && \text{(D.8)} \\ &= \beta^{-1}(t) g(t) d\tilde{\mathbf{w}} && \text{(D.9)} \end{aligned}$$

which also leads to Equation D.6. Let $\tilde{p}_t$ be the marginal distribution of y_t and p_t the marginal distribution of x_t, similar to Equation 2.58, it can be expressed as:

$$\begin{aligned} \tilde{p}_t(y_t) &= \int \frac{\exp(-\mathcal{E}(y))}{Z_0} \mathcal{N}(y_t; y, \tilde{\sigma}_t^2 I) && \text{(D.10)} \\ &\approx \frac{1}{Z_0} \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(y_{0|t}^{(i)})) && \text{(D.11)} \\ \tilde{S}_t(y_t) &= \nabla_{y_t} \log \tilde{p}_t(y_t) \approx \nabla_{y_t} \log \sum_{i=1}^{K} \exp(-\mathcal{E}(y_{0|t}^{(i)})) && \text{(D.12)} \\ \tilde{\mathcal{E}}_t(y_t) &\approx -\log \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(y_{0|t}^{(i)})) && \text{(D.13)} \\ y_{0|t}^{(i)} &\sim \mathcal{N}(y; y_t, \tilde{\sigma}_t^2 I) && \text{(D.14)} \end{aligned}$$

Therefore, we can learn scores and energies of y_t simply by following DEM and EnDEM for VE SDEs. Then for sampling, we can simulate the reverse SDE of y_t and eventually, we have $x_0 = y_0$. And we can use x_t as the neural network input for numerical stability, as described by Akhound-Sadegh et al. (2024).

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

$$\begin{aligned} p_t(x_t) &= \beta_t^{-1} \tilde{p}_t(\beta_t^{-1} x_t) = \beta_t^{-1} \tilde{p}_t(y_t) && \text{(D.15)} \\ S_t(x_t) &= \beta_t^{-1} \tilde{S}_t(\beta_t^{-1} x_t) = \beta_t^{-1} \tilde{S}_t(y_t) && \text{(D.16)} \end{aligned}$$

which provides us the energy and score estimator for x_t:

$$\mathcal{E}_t(x_t) \approx -\log \beta_t^{-1} \frac{1}{K} \sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)})) \tag{D.17}$$

$$S_t(x_t) \approx \beta_t^{-1} \nabla_{x_t} \log \sum_{i=1}^{K} \exp(-\mathcal{E}(x_{0|t}^{(i)})) \tag{D.18}$$

$$x_{0|t}^{(i)} \sim \mathcal{N}(x; \beta_t^{-1} x_t, \tilde{\sigma}^2(t) I) \tag{D.19}$$

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 (D.9) 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.

Besides, as discussed in Section 2.2.4, the MC energy and score estimators are based on an IS estimator. In this perspective, the native proposal $\mathcal{N}(y; y_t, \tilde{\sigma}^2(t) I)$ can be far away from the target which is proportional to $\exp(-\mathcal{E}(y))\mathcal{N}(y_t; y, \tilde{\sigma}^2(t) I)$. Therefore, these MC estimators can be very unreliable, and an AIS estimator discussed in Section 2.2.5 can be useful. A generalized implementation (w/ or w/o AIS) will be of interest in the future.

Appendix E

Efficient Sampling for EnDEM

There's an essential issue of EnDEM that it requires additional differentiation over its input, i.e. $\nabla E_\theta(x,t)$, during sampling. A possible remedy can be raised by reviewing Tweedie's formula (2.50). Due to the limited schedule for this thesis, we only propose the method below, and leave further experiments as future work and will be potentially shown in ICLR 2025 or other conferences after.

In a VE setting, by Tweedie's formula:

$$S_t(x_t) = -\frac{x_t - \mathbb{E}[x_0|x_t]}{\sigma_t^2} \tag{E.1}$$

$$\mathbb{E}[x_0|x_t] = \int x_0 p(x_0|x_t) dx_0 \tag{E.2}$$

$$= \int x_0 \frac{p(x_t|x_0)p(x_0)}{p_t(x_t)} dx_0 \tag{E.3}$$

$$= \int x_0 \frac{\mathcal{N}(x_t; x_0, \sigma_t^2 I)\exp(-\mathcal{E}(x_0))/Z_0}{p_t(x_t)} dx_0 \tag{E.4}$$

$$= \int x_0 \frac{\mathcal{N}(x_t; x_0, \sigma_t^2 I)\exp(-\mathcal{E}(x_0))}{\exp(-\mathcal{E}_t(x_t))} dx_0 \tag{E.5}$$

where $p_t(x_t) = \exp(-\mathcal{E}_t(x_t))/Z_0$. And we're learning an energy model:

$$E_\theta(x,t) \approx \mathcal{E}_t(x), \forall t \in [0,1] \tag{E.6}$$

then we can approximate the expectation using the learned network:

$$\mathbb{E}[x_0|x_t] \approx D_\theta(x_t,t) := \int x_0 \frac{\mathcal{N}(x_t;x_0,\sigma_t^2 I)\exp(-E_\theta(x_0,0))}{\exp(-E_\theta(x_t,t))} dx_0 \tag{E.7}$$

$$= \exp(E_\theta(x_t,t))\mathbb{E}_{\mathcal{N}(x;x_t,\sigma_t^2 I)}[x\exp(-E_\theta(x,0))] \tag{E.8}$$

$$\approx \frac{1}{K}\sum_{i=1}^{K} x_{0|t}^{(i)} \exp(E_\theta(x_t,t) - E_\theta(x_{0|t}^{(i)},0)) := D_\theta^K(x_t,t) \tag{E.9}$$

$$s_\theta(x_t,t) = -\frac{x_t - D_\theta(x_t,t)}{\sigma_t^2} \tag{E.10}$$

where $x_{0|t}^{(i)} \sim \mathcal{N}(x;x_t,\sigma_t^2 I)$. Especially when σ_t is small (i.e. for t close to 0), since $x_{0|t}^{(i)} = x_t + \sigma_t \varepsilon^{(i)}$ with $\varepsilon^{(i)} \sim \mathcal{N}(0,I)$, we can take a first order Taylor expansion of E_θ around x_t

$$D_\theta^K(x_t,t) = \frac{1}{K}\sum_{i=1}^{K} x_{0|t}^{(i)} \exp(E_\theta(x_t,t) - E_\theta(x_t,0) - \sigma_t(\varepsilon^{(i)})^T \nabla E_\theta(x_t,0)) \tag{E.11}$$

$$\approx \frac{1}{K}\sum_{i=1}^{K} x_{0|t}^{(i)} \exp(-\sigma_t(\varepsilon^{(i)})^T \nabla E_\theta(x_t,0)) \tag{E.12}$$

$$\approx \frac{1}{K}\sum_{i=1}^{K} (x_t + \sigma_t\varepsilon^{(i)})(1 - \sigma_t(\varepsilon^{(i)})^T \nabla E_\theta(x_t,0)) \tag{E.13}$$

therefore, when t close to 0, in expectation we will have

$$\mathbb{E}[D_\theta^K(x_t,t)] = x_t - \sigma_t^2 \nabla E_\theta(x_t,0) \tag{E.14}$$

$$\mathbb{E}[s_\theta^K(x_t,t)] = -\nabla E_\theta(x_t,0) \tag{E.15}$$

which recovers scores at $t = 0$. Finally, plugging the new approximated scores into Equation 2.53, we have the following more efficient discretized reverse SDE for EnDEM

$$x_{t-1} = D_\theta(x_t,t) + \sigma_t \varepsilon_t \tag{E.16}$$

$$\approx D_\theta^K(x_t,t) + \sigma_t \varepsilon_t \tag{E.17}$$

where $\varepsilon_t \sim \mathcal{N}(0,I)$.

Appendix F

Incorporating Metropolis-Hastings

Leveraging the learned annealed energies, we're able to combine MH during sampling to boost sample quality. At a specific time t and corresponding sample x_t, one can simulate the reverse SDE by discretizing it into a small time step Δ and generating $x' = x_{t-\Delta}$,

$$x'_t = x_t + \sigma_t^2 \nabla \log p_t(x_t) + \sigma_t \varepsilon \tag{F.1}$$

$$= x_t - \sigma_t^2 \nabla \mathcal{E}_t(x_t) + \sigma_t \varepsilon \tag{F.2}$$

$$\approx x_t - \sigma_t^2 \nabla E_\theta(x_t, t) + \sigma_t \varepsilon \tag{F.3}$$

where $\varepsilon \sim \mathcal{N}(0, I)$ and we stick using the VE SDE introduced in Equation 2.44. Therefore, the above Langevin Dynamics introduces a non-symmetric proposal, $q_t(x'|x)$, towards marginal distribution $p_{t-\Delta}$

$$q_t(x'|x) = \mathcal{N}(x'; x - \sigma_t^2 \nabla E_\theta(x,t), \sigma_t^2 I) \tag{F.4}$$

Therefore, one can employ MH to correct samples, with an acceptance rate

$$\alpha_t(x', x; \theta) = \min\left\{1, \frac{\exp(-E_\theta(x', t-\Delta))\mathcal{N}(x; x' - \sigma_t^2 \nabla E_\theta(x', t), \sigma_t^2 I)}{\exp(-E_\theta(x, t-\Delta)\mathcal{N}(x'; x - \sigma_t^2 \nabla E_\theta(x, t), \sigma_t^2 I)}\right\} \tag{F.5}$$

resembling a neural MALA. By leveraging the aforementioned denoiser $D_\theta(x_t, t)$, and its approximation $D_\theta^K(x_t, t)$, we can further simplify Equation F.5 as

$$\alpha_t(x', x; \theta) = \min\left\{1, \frac{\exp(-E_\theta(x', t-\Delta))\mathcal{N}(x; D_\theta(x', t), \sigma_t^2 I)}{\exp(-E_\theta(x, t-\Delta)\mathcal{N}(x'; D_\theta(x, t), \sigma_t^2 I)}\right\} \tag{F.6}$$

$$\approx \min\left\{1, \frac{\exp(-E_\theta(x', t-\Delta))\mathcal{N}(x; D_\theta^K(x', t), \sigma_t^2 I)}{\exp(-E_\theta(x, t-\Delta)\mathcal{N}(x'; D_\theta^K(x, t), \sigma_t^2 I)}\right\} \tag{F.7}$$

Appendix G

Supplementary Experiments

Supplementary Experiments Settings

Dataset: LJ-n

This dataset describes a system consist 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

$$\mathcal{E}^{\mathrm{LJ}}(x) = \frac{\varepsilon}{2\tau} \sum_{ij} \left(\left(\frac{r_m}{d_{ij}} \right)^6 - \left(\frac{r_m}{d_{ij}} \right)^{12} \right) \tag{G.1}$$

and the harmonic potential

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

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 $r_m = 1$, $\tau = 1$, $\varepsilon = 1$ 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. (2023) as a test set.

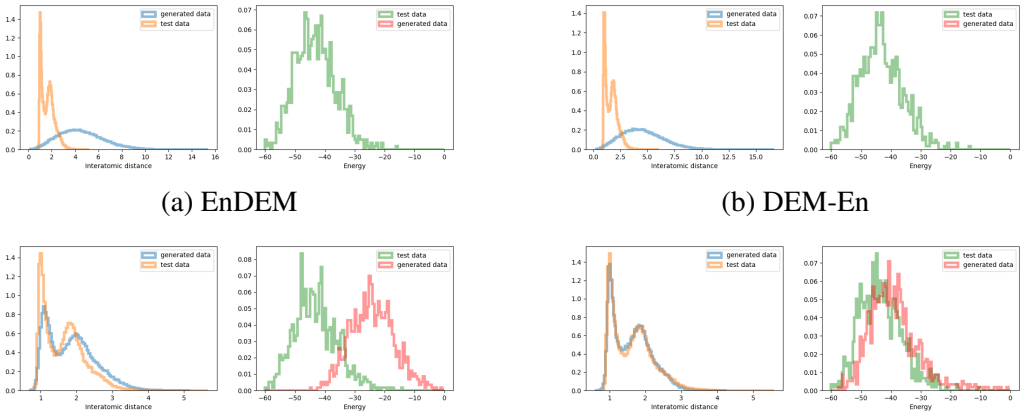
Hyperparameters for LJ-n

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

of $1e-3$, $K = 1000$ samples for S_K and E_K, $K = 400$ samples for E_K^B and S_K^B, and we clipped S_K and S_K^B to a max norm of 20.

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 $\sigma_{\text{min}} = 0.5$ and $\sigma_{\text{max}} = 4$, a learning rate of $1e-3$, $K = 100$ samples for S_K and E_K, $K = 400$ samples for E_K^B and S_K^B, and we clipped S_K and S_K^B to a max norm of 20.

Supplementary Experiments Results



(a) EnDEM (b) DEM-En

(c) BEnDEM (d) DEM

Fig. G.1 For each subplot, **Left:** histograms of the interatomic distance of drawn samples and test data (i.e. ground-truth samples); **Right:** histograms of energy of drawn samples and test data.

Figure G.1 shows that EnDEM is diverging possibly as a result of the sharp energy landscape, while DEM-En is not stable for training as well. Even though BEnDEM helps, and can match the modes of interatomic distance distribution, it underperforms DEM so far. Therefore, exploring more stable training of EnDEM as well as BEnDEM for a sharp energy landscape is our most urgent direction.