Value Gradient Sampler: Learning Invariant Value Functions for Equivariant Diffusion Sampling

Himchan Hwang[1*] **Hyeokju Jeong**[1*] **Dong Kyu Shin**[1*] **Che-Sang Park**[1] **Sehee Kweon**[1]
Sangwoong Yoon[2†] **Frank C. Park**[1†]
[1]Seoul National University [2]Ulsan National Institute of Science and Technology (UNIST)
[*]Equal contribution [†]Corresponding authors

Abstract

We propose the Value Gradient Sampler (VGS), a diffusion sampler parameterized by value functions. VGS generates samples from an unnormalized target density (i.e., energy) by evolving randomly initialized particles along the gradient of the value function. In many sampling problems where the target density exhibits invariant symmetries, value functions provide a novel approach to leveraging invariant networks for sampling by inducing an equivariant gradient flow, without requiring more complex equivariant networks. The value networks are trained via temporal difference learning, which supports off-policy training and other established reinforcement learning (RL) techniques. By combining advanced RL methods with efficient invariant networks, VGS achieves both the highest sample quality and the fastest sampling speed among our baselines on the 55-particle Lennard-Jones system.

1 Introduction

Sampling from complex, high-dimensional distributions is a fundamental problem in statistics, machine learning, and the natural sciences. The goal of a sampling algorithm is to generate independent and identically distributed (i.i.d.) samples from an unnormalized target density $q(\mathbf{x})$. Recently, samplers inspired by diffusion processes have emerged as a powerful class of algorithms for this task. These methods construct samples via a multi-step diffusion process that interpolates between a simple distribution and the target density, demonstrating strong performance across diverse applications (Zhang and Chen, 2022; Vargas et al., 2024; Berner et al., 2022; Vargas et al., 2023; Bengio et al., 2023; Lahlou et al., 2023; Havens et al., 2025).

In many important applications of sampling, the target density exhibits symmetries. For instance, the equilibrium state distributions of particle and molecular systems are governed by Euclidean symmetries, such as translation and rotation, which are formalized by the group SE(3) (or E(3) if reflection is also considered). Respecting these symmetries is crucial, as it can lead to improved performance, better sample efficiency, and enhanced generalization (Karczewski et al., 2024). To sample from an E(3)-invariant target density, most diffusion samplers employ E(3)-equivariant neural network architectures (Satorras et al., 2021) to parametrize E(3)-equivariant score functions (Akhound-Sadegh et al., 2024; Havens et al., 2025).

However, enforcing Euclidean equivariance in neural architectures is a non-trivial endeavor. Equivariant networks often rely on mathematically sophisticated constructions, incur significant computational overhead, or impose structural restrictions (Köhler et al., 2019; Thomas et al., 2018; Fuchs et al., 2020; Geiger and Smidt, 2022; Midgley et al., 2023a). While EGNNs have become widely adopted due to their relatively simple design, they remain slower than general-purpose architectures such as multi-layer perceptrons or graph neural networks (Satorras et al., 2021). In particular, the computation in EGNNs inevitably scales as $\mathcal{O}(n^2)$ with respect to the number of particles n in the system due to all-pairs message passing.

In contrast, achieving E(3)-invariance may be considerably simpler, especially for particle systems where the energy depends on the relative positions of particles. A permutation-invariant neural network over pairwise distances and node features can represent an invariant potential without requiring explicitly equiv-

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

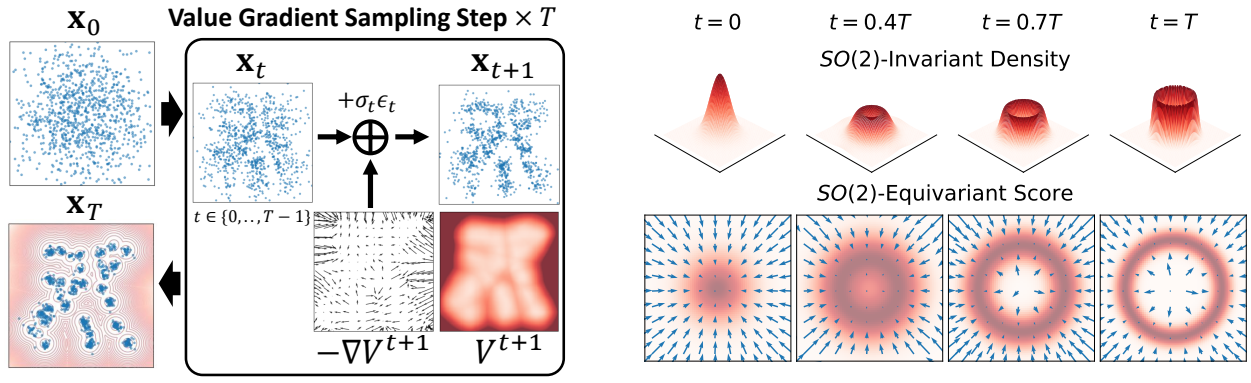

Figure 1: (**Left**) Value Gradient Sampling. Initial samples are iteratively drifted along the gradient of the next-step value function to match the target distribution. (**Right**) Diffusion of an SO(2)-invariant target density ($t = T$) via a variance-preserving diffusion process. The diffused densities also exhibit SO(2)-invariance while their score functions possess SO(2)-equivariance. VGS employs a sequence of invariant value functions (which are related to the diffused log-densities) to parameterize the equivariant gradient flow.

ariant layers. Moreover, taking the gradient of such an invariant network naturally induces an equivariant gradient flow (Papamakarios et al., 2021).

In this work, we propose **Value Gradient Sampler** (VGS), a novel diffusion-based sampling framework that leverages $\mathrm{E}(m)$-invariant rather than $\mathrm{E}(m)$-equivariant neural networks for m-dimensional, n-particle systems. Instead of directly parameterizing equivariant score functions, VGS uses a sequence of invariant value functions to represent the diffusion process (Fig. 1). These value functions are related to the diffused target densities, which preserve the same symmetries as the original target density. Sampling is then performed by following the gradients of these value functions, which are $\mathrm{E}(m)$-equivariant by construction. In this way, VGS yields an equivariant gradient flow without the need for explicitly equivariant architectures.

The value functions in VGS are trained through reinforcement learning (RL). We show that temporal-difference (TD) learning (Sutton, 1988) can be naturally applied to minimize an upper bound of the KL divergence between the generated samples and the target density. This connection to RL allows us to incorporate established techniques such as TD(λ), off-policy exploration, and double value networks (Sec. 3.4) to further enhance the performance of VGS.

We demonstrate the effectiveness of VGS on particle system benchmarks, where the goal is to sample equilibrium states given the energy function of a system. VGS achieves the highest sample-quality metrics among the baselines in the LJ-55 setting, a challenging system of 55 indistinguishable particles with an energy function defined in a 165-dimensional space. Simultaneously, VGS dramatically reduces sampling time compared to existing equivariant samplers by replacing a computationally intensive equivariant network with an efficient invariant network. VGS also remains competitive on standard benchmarks without explicit symmetries. Our results highlight that VGS is a promising approach for symmetry-aware sampling applications. The implementation of VGS is publicly available [1].

2 Preliminaries

Sampling Problems. We consider drawing independent samples from a strictly positive target distribution $q(\mathbf{x})$ on $\mathbb{R}^D$ of Boltzmann-form

$$q(\mathbf{x}) = \frac{1}{Z}\exp\left(-E(\mathbf{x})\right), \tag{1}$$

where $E : \mathbb{R}^D \to \mathbb{R}$ is an energy function and $Z = \int_{\mathbb{R}^D} \exp\left(-E(\mathbf{x})\right)d\mathbf{x}$ is the normalizing constant.

Invariance and Equivariance. Given a group G acting on a set $\mathcal{X}$ via the action $\cdot$, a scalar function $f : \mathcal{X} \to \mathbb{R}$ is G-*invariant* if $f(g \cdot \mathbf{x}) = f(\mathbf{x})$ for all $g \in \mathrm{G}$ and $\mathbf{x} \in \mathcal{X}$. A map $F : \mathcal{X} \to \mathcal{X}$ is G-*equivariant* if $F(g \cdot \mathbf{x}) = g \cdot F(\mathbf{x})$ for all $g \in \mathrm{G}$ and $X \in \mathcal{X}$.

Symmetry Groups in n-Particle Systems. Consider the system of n particles in $\mathbb{R}^m$. Let the configuration space be $\mathcal{X} = \mathbb{R}^{m\times n}$ with $\mathbf{x} = [x_1, \ldots, x_n]$, where $x_i \in \mathbb{R}^m$ denotes the coordinates of the i-th particle. Two symmetry groups act naturally on $\mathcal{X}$: the Euclidean group $\mathrm{E}(m)$ (translations, rotations, reflections) and the symmetric group S_n (permutations of particle indices). Elements $(R, p) \in \mathrm{E}(m)$, where

[1] https://github.com/swyoon/value-gradient-sampler

$R \in \mathrm{O}(m)$ and $p \in \mathbb{R}^m$, act by

$$(R, p) \cdot \mathbf{x} = [Rx_1+p, \ldots, Rx_n+p], \tag{2}$$

and permutations $\sigma \in \mathrm{S}_n$ act by re-indexing $\sigma \cdot \mathbf{x} = [x_{\sigma(1)}, ..., x_{\sigma(n)}]$. These actions commute, inducing a product action of $\mathrm{E}(m) \times \mathrm{S}_n$ on $\mathcal{X}$. Under this product action, the energy $E(\mathbf{x})$ is unchanged, assuming that the particles are indistinguishable. Thus, the energy $E(\mathbf{x})$ is $\mathrm{E}(m) \times \mathrm{S}_n$-invariant.

Training Parametric Samplers. We aim to train a parametric sampler $\pi_\phi(\mathbf{x})$ with parameter ϕ to generate approximate samples from the target density $q(\mathbf{x})$. A natural choice of the objective function is the (reverse) KL divergence between $\pi_\phi(\mathbf{x})$ and $q(\mathbf{x})$: $\min_\phi \mathrm{KL}(\pi_\phi(\mathbf{x})||q(\mathbf{x})) = \min_\phi \mathbb{E}_{\pi_\phi}[\log(\pi_\phi(\mathbf{x})/q(\mathbf{x}))]$, which does not require samples from $q(\mathbf{x})$ for training. Other objectives, such as log-variance loss, have also been considered (Richter and Berner, 2024).

Diffusion Samplers. Diffusion samplers generate samples through the following iterative drift-diffusion process:

$$\begin{aligned} \pi(\mathbf{x}_0) &= \mathcal{N}(0, \sigma_{\text{init}}^2 I), \\ \pi(\mathbf{x}_{t+1}|\mathbf{x}_t) &= \mathcal{N}(\alpha_t \mathbf{x}_t + \mu^t(\mathbf{x}_t), \sigma_t^2 I), \end{aligned} \tag{3}$$

for $t = 0, \ldots, T-1$. The schedules $\{\alpha_t, \sigma_t\}$ and σ_{init} are fixed. In practice, we parameterize the drift as $\mu_\phi^t(\mathbf{x}_t)$ and thus define the transition $\pi_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t)$. Starting from the initial Gaussian sample $\mathbf{x}_0$, we sequentially scale, drift, and diffuse the sample to obtain the final sample $\mathbf{x}_T$. We will often write $\mathbf{x}$ to denote $\mathbf{x}_T$. The sampler distribution $\pi_\phi(\mathbf{x})$ is induced as the marginal distribution of final samples. This drift-diffusion process (Eq. 3) is widely found in various methods, such as diffusion models (Ho et al., 2020; Song et al., 2021), Euler-Maruyama discretization of SDE samplers (Zhang and Chen, 2022; Berner et al., 2022; Vargas et al., 2023), and continuous GFlowNets (Lahlou et al., 2023).

3 Value Gradient Sampling

In this section, we introduce **Value Gradient Sampler** (VGS), a diffusion sampler that parametrizes the drift $\mu_\phi^t(\mathbf{x}_t)$ using a value function $V_\phi^t(\mathbf{x}_t)$. The optimal control formulation and temporal difference training algorithms are described. Incorporating symmetries into VGS will be discussed in Sec. 4.

3.1 Sampling as Optimal Control

We train a diffusion sampler π_ϕ to match the target distribution q by leveraging the optimal control view on diffusion-based sampling proposed in Zhang and Chen (2022). Instead of directly minimizing $\mathrm{KL}(\pi_\phi(\mathbf{x})\|q(\mathbf{x}))$, we minimize an upper bound from the data-processing inequality: $\mathrm{KL}(\pi_\phi(\mathbf{x})\|q(\mathbf{x})) \leq \mathrm{KL}(\pi_\phi(\mathbf{x}_{0:T})\|\tilde{q}(\mathbf{x}_{0:T}))$. We set the joint target distribution as $\tilde{q}(\mathbf{x}_{0:T}) = \tilde{\pi}(\mathbf{x}_{0:T})q(\mathbf{x}_T)/\tilde{\pi}(\mathbf{x}_T)$, which ensures $\tilde{q}(\mathbf{x}_T) = q(\mathbf{x}_T)$. Here, $\tilde{\pi}(\mathbf{x}_{0:T})$ is the joint reference distribution, defined as the uncontrolled process in Eq. 3 with $\tilde{\pi}(\mathbf{x}_0) = \mathcal{N}(0, \sigma_{\text{init}}^2 I)$ and $\tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\alpha_t \mathbf{x}_t, \sigma_t^2 I)$. This construction subsumes prior choices used in PIS (Zhang and Chen, 2022, $\sigma_{\text{init}} = 0, \alpha_t = 1$) and DDS (Vargas et al., 2023, $\sigma_{\text{init}} = 1, \alpha_t = \sqrt{1-\sigma_t^2}$).

We use a value-based dynamic programming approach to this minimization problem. Minimizing $\mathrm{KL}(\pi(\mathbf{x}_{0:T})\|\tilde{q}(\mathbf{x}_{0:T}))$ over the admissible class Π of policies given by Eq. 3 yields the following optimal control problem:

$$\min_{\pi \in \Pi} \mathbb{E}_{\pi(\mathbf{x}_{0:T})}\left[\sum_{t=0}^{T-1} \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + \tilde{E}(\mathbf{x}_T)\right], \tag{4}$$

where $\tilde{E}(\mathbf{x}_T) = E(\mathbf{x}_T) + \log \tilde{\pi}(\mathbf{x}_T)$. See Appendix A.1 for the derivation. Here, the policy $\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)$ is optimized to minimize the sum of running costs $\|\mu^t(\mathbf{x}_t)\|^2/2\sigma_t^2$, and the terminal cost $\tilde{E}(\mathbf{x}_T)$.

A value function $V_\pi^t(\mathbf{x}_t)$ is the expected sum of future costs starting from $\mathbf{x}_t$ and following π:

$$V_\pi^t(\mathbf{x}_t) = \mathbb{E}_{\pi(\cdot|\mathbf{x}_t)}\left[\sum_{i=0}^{T-t-1} \frac{\|\mu^{t+i}(\mathbf{x}_{t+i})\|^2}{2\sigma_{t+i}^2} + \tilde{E}(\mathbf{x}_T)\middle|\mathbf{x}_t\right]. \tag{5}$$

The optimal value function is the minimum over all possible $\pi(\cdot|\mathbf{x}_t)$, $V_*^t(\mathbf{x}_t) = \min_{\pi(\cdot|\mathbf{x}_t) \in \Pi} V_\pi^t(\mathbf{x}_t)$. If $\tilde{q}(\mathbf{x}_{t+1:T}|\mathbf{x}_t) \in \Pi$, then $V_*^t(\mathbf{x}_t)$ is the energy of the marginal density ratio $\tilde{q}(\mathbf{x}_t)/\tilde{\pi}(\mathbf{x}_t)$:

$$\tilde{q}(\mathbf{x}_t)/\tilde{\pi}(\mathbf{x}_t) = \frac{1}{Z}\exp\left(-V_*^t(\mathbf{x}_t)\right). \tag{6}$$

The derivation is in Appendix A.2. Intuitively, $V_*^t(\mathbf{x}_t)$ is lower on states that are more probable under marginal target density $\tilde{q}(\mathbf{x}_t)$ compared to the reference density $\tilde{\pi}(\mathbf{x}_t)$.

3.2 Value Gradient Sampler

We present **Value Gradient Sampler** (VGS), a sampler that approximately solves this optimal control problem by drifting a particle along the gradient of the next-step value function. For our policy at time t defined in Eq. 3, we will find the drift vector $\mu^t(\mathbf{x}_t)$ that minimizes the expected future cost from $\mathbf{x}_t$, which can be represented using the next-step value function

Algorithm 1 Value Gradient Sampler

Input: Value $V_\phi^t(\mathbf{x}_t)$,
constants $\{\alpha_t\}_{t=0}^{T-1}$, $\{\sigma_t\}_{t=0}^{T-1}$, σ_{init}.
$\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_{\text{init}}^2 I)$ // Initial samples
for $t = 0$ **to** $T-1$ **do**
  $\mu_\phi^t(\mathbf{x}_t) = -\sigma_t^2 \nabla_{\alpha_t \mathbf{x}_t} V_\phi^{t+1}(\alpha_t \mathbf{x}_t)$ // Eq. (8)
  $\epsilon_t \sim \mathcal{N}(0, I)$
  $\mathbf{x}_{t+1} = \alpha_t \mathbf{x}_t + \mu_\phi^t(\mathbf{x}_t) + \sigma_t \epsilon_t$
end for
Output: $\mathbf{x}_T$

$V_\pi^{t+1}(\mathbf{x}_{t+1})$:

$$\min_{\mu^t(\mathbf{x}_t)} \mathbb{E}_{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)} \left[\frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + V_\pi^{t+1}(\mathbf{x}_{t+1}) \middle| \mathbf{x}_t \right]. \quad (7)$$

As this objective lacks a closed-form solution, we apply a first-order Taylor expansion to $V_\pi^{t+1}(\mathbf{x}_{t+1})$ around $\alpha_t \mathbf{x}_t$ to obtain:

$$\begin{aligned} \mathop{\mathbb{E}}_{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)} \left[V_\pi^{t+1}(\mathbf{x}_{t+1}) \right] &= \mathbb{E}_{\epsilon_t}[V_\pi^{t+1}(\alpha_t \mathbf{x}_t + \mu^t + \sigma_t \epsilon_t)] \\ &\approx V_\pi^{t+1}(\alpha_t \mathbf{x}_t) + \mu^t(\mathbf{x}_t)^\top \nabla_{\alpha_t \mathbf{x}_t} V_\pi^{t+1}(\alpha_t \mathbf{x}_t), \end{aligned}$$

where $\epsilon_t \sim \mathcal{N}(0, I)$. Substituting this approximation back into Eq. 7 and setting the derivative with respect to μ^t to zero yields the optimal drift:

$$\mu^t(\mathbf{x}_t) = -\sigma_t^2 \nabla_{\alpha_t \mathbf{x}_t} V_\pi^{t+1}(\alpha_t \mathbf{x}_t). \quad (8)$$

The detailed derivation is presented in Appendix A.3. Note that $\mu^t(\mathbf{x}_t)$ is now determined entirely by the gradient of the next-step value function. In practice, we use a parameterized value network $V_\phi^t(\mathbf{x}_t)$ mapping $(\mathbf{x}_t, t)$ to a scalar, substituting V_ϕ for V_π in Eq. 8 for sampling (Algorithm 1). To approach the optimal value function and policy, we follow the framework of generalized policy iteration, which alternates between policy evaluation and improvement. VGS corresponds to the policy improvement step, refining the policy based on the current value function. The policy evaluation step is described in the next section.

3.3 Temporal Difference Learning

We employ TD learning (Sutton, 1988) to estimate the value function from samples. From the definition of the value function (Eq. 5), we obtain the following recurrence relation: $V_\pi^t(\mathbf{x}_t) = \mathbb{E}_{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)} \left[\|\mu^t(\mathbf{x}_t)\|^2 / 2\sigma_t^2 + V_\pi^{t+1}(\mathbf{x}_{t+1}) | \mathbf{x}_t \right]$. To approximate V_π^t, we train a parameterized value network V_ϕ^t to satisfy this recurrence, resulting in a regression-type loss with the TD target $\hat{V}_{\text{TD}}^t(\mathbf{x}_{t:t+1})$. The TD target is defined using a target network $V_{\phi^-}^t(\mathbf{x}_t)$, a detached copy of the value network parametrized by ϕ^-. Specifically, given a transition $\mathbf{x}_{t+1} \sim \pi_{\phi^-}(\mathbf{x}_{t+1}|\mathbf{x}_t)$, we minimize the mean squared TD error:

$$\min_\phi \mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+1}|\mathbf{x}_t)} \left[(V_\phi^t(\mathbf{x}_t) - \hat{V}_{\text{TD}}^t(\mathbf{x}_{t:t+1}))^2 \right], \quad (9)$$

$$\text{where} \quad \hat{V}_{\text{TD}}^t(\mathbf{x}_{t:t+1}) = \frac{\|\mu_{\phi^-}^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + V_{\phi^-}^{t+1}(\mathbf{x}_{t+1}).$$

Gradients are not backpropagated into the target network parameters ϕ^-. Instead, ϕ^- is updated separately via exponential moving averaging: $\phi^- \leftarrow \kappa\phi^- + (1-\kappa)\phi$ for $0 < \kappa < 1$. This target-network scheme is known to stabilize training with function approximation (Mnih et al., 2015; Haarnoja et al., 2018). The overall training procedure is summarized in Algorithm 2 in the Appendix.

Note that there exist critical differences between our TD learning and Detailed Balance (DB) (Bengio et al., 2023). First, we employ a fixed target network during the update (Eq. 9). Second, the TD loss does not directly update the policy. Instead, the policy is implicitly optimized via taking the gradient of the next-step value (Eq. 8). Consequently, only the current state value is updated based on the future state value, allowing information from terminal states to propagate to earlier states in a stable and rectified manner.

TD(λ). The TD learning described above, often referred to as TD(0), can be slow in propagating information in time, as only a single-step transition is used in the update. Alternatively, TD(λ) (Sutton, 1988) incorporates all available future temporal differences by computing a weighted average with exponentially decaying weights. We first define the TD error as follows:

$$\delta_t(\mathbf{x}_t, \mathbf{x}_{t+1}) = \frac{\|\mu_{\phi^-}^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + V_{\phi^-}^{t+1}(\mathbf{x}_{t+1}) - V_{\phi^-}^t(\mathbf{x}_t). \quad (10)$$

Given a trajectory $\mathbf{x}_{t+1:T} \sim \pi_{\phi^-}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)$ and a constant $\lambda \in [0, 1]$, the TD(λ) target is defined as:

$$\hat{V}_{\text{TD}(\lambda)}^t(\mathbf{x}_{t:T}) = V_{\phi^-}^t(\mathbf{x}_t) + \sum_{i=0}^{T-t-1} \lambda^i \delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}_{t+i+1}), \quad (11)$$

which can be used in Eq. 9 instead of $\hat{V}_{\text{TD}}^t(\mathbf{x}_{t:t+1})$. This formulation offers a principled trade-off between bias and variance, interpolating between one-step TD ($\lambda = 0$) and Monte Carlo estimation ($\lambda = 1$).

SubTB(λ) (Madan et al., 2023) implements a similar idea but applies a weighted average over subtrajectory losses, whereas TD(λ) incorporates it directly into the target calculation. By looking ahead multiple steps, TD(λ) is often more efficient and effective than TD(0) in assigning credit to earlier actions. This benefit becomes more pronounced for larger T, as demonstrated in our comparison experiments in Sec. 5.2.

3.4 Leveraging RL Techniques in VGS

The TD formulation enables us to leverage well-established techniques in deep RL.

Training with an Exploration Policy. An exploration policy π_{expl} is often used in RL to promote exploration. Assuming the transition density $\pi_{\text{expl}}(\mathbf{x}_{t+1}|\mathbf{x}_t)$ is tractable, off-policy TD(λ) targets can be computed on a sampled trajectory $\mathbf{x}_{0:T} \sim \pi_{\text{expl}}(\mathbf{x}_{0:T})$ using importance sampling. The TD targets for all timesteps $t = 0, \dots, T$ can be computed from the stored tuples $\{\mathbf{x}_t, \mu^t_{\phi^-}(\mathbf{x}_t)\}_{t=0}^T$, collected during the sampling phase. Details on off-policy TD(λ) computation are given in Appendix A.4. In our experiments, we set the exploration policy as the current policy with an amplified noise level $(\sigma_t)_{\text{expl}} = \eta\sigma_t$ for a scale factor $\eta > 1$, following prior works (Lahlou et al., 2023; Madan et al., 2023; Sendera et al., 2024).

Double Value Networks. Using a single value function for both target evaluation and action optimization induces an overestimation bias in RL (Thrun and Schwartz, 2014). TD3 addresses this issue by maintaining two independent value networks: the policy is optimized using one critic, while targets are computed as the minimum of the two critics' estimates (Fujimoto et al., 2018). In VGS, we adopt the same principle: using a single-value network for the value-gradient sampling (policy optimization) step and computing TD targets as the minimum of two value estimates. This design increases training time computation but leaves the sampling cost unchanged. We empirically found that the double-value approach improves training stability and convergence for VGS.

Training with Sample Buffers. In some cases, important samples $\mathbf{x}_T$ may be externally provided, such as MCMC samples, or states selected from a prioritized replay buffer. To leverage such data, we sample the backward trajectory $\mathbf{x}_{0:T-1} \sim \tilde{q}(\mathbf{x}_{0:T-1}|\mathbf{x}_T)$ and perform TD updates along the resulting trajectory. We prefer TD(0) in this setting, as TD(λ) requires computationally expensive resampling of full future trajectories $\mathbf{x}_{t+1:T} \sim \pi_{\phi^-}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)$ at each step, whereas TD(0) efficiently uses a single transition. We evaluate VGS with sample buffers in our synthetic distribution experiments in Sec. 5.2. Following Sendera et al. (2024), we alternate training VGS on forward and backward trajectories.

4 Value Gradient Sampler for n-Particle Systems

A sampler that generates equilibrium states of a many-body system is called a Boltzmann generator (Noé et al., 2019). In addition to being a challenging sampling benchmark, inferring stable configurations given an energy function is an important scientific problem in chemistry and biology (Jumper et al., 2021; Hoogeboom et al., 2022; Havens et al., 2025). The energy of an n-body system is invariant to translation, rotation, reflection, and permutation of particles, exhibiting $\mathrm{E}(m) \times \mathrm{S}_n$-invariance. To respect this symmetry, previous approaches have built samplers using $\mathrm{E}(m)$-*equivariant* networks (Akhound-Sadegh et al., 2024; He et al., 2024; Midgley et al., 2023a). In this section, we describe how VGS is applied to the task of sampling equilibrium states of n-particle systems. We formulate the problem on a zero-mean space, establish the invariance of the value function, and thus parameterize it using $\mathrm{E}(m)$-*invariant* networks.

4.1 Zero-mean Space

The configuration of a system $\mathbf{x}$ can be projected onto the zero-mean subspace $\mathcal{X}$, utilizing the translation invariance of the system. The zero-mean space $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^{m\times n} \mid \sum_{i=1}^n x_i = 0\}$ is an $m(n-1)$-dimensional subspace of $\mathbb{R}^{m\times n}$ where the particles satisfy the zero-mean constraint. As $\mathcal{X}$ is a vector space isomorphic to $\mathbb{R}^{m(n-1)}$, the arguments in Sec. 3 remain valid, except that the effective dimension reduces to $D_{\text{eff}} = m(n-1)$. Working in $\mathcal{X}$ also makes the Boltzmann distribution (Eq. 1) well defined, as the normalizing constant becomes finite. In practice, we project both the drift μ^t_ϕ and the noise ϵ_t onto $\mathcal{X}$ at each step.

4.2 Invariance of the Value Function

On the zero-mean space $\mathcal{X}$, translations are removed, so an $\mathrm{E}(m) \times \mathrm{S}_n$–invariant energy E becomes $\mathrm{O}(m) \times \mathrm{S}_n$–invariant. The main claim of this subsection is that the value function of VGS V^t_π and the optimal value function V^t_* inherit exactly the same $\mathrm{O}(m)\times\mathrm{S}_n$ invariance. Let $\circ$ denote the product action of $\mathrm{O}(m)\times\mathrm{S}_n$ on $\mathcal{X}$: $(R,\sigma)\circ\mathbf{x} = [Rx_{\sigma(1)}, ..., Rx_{\sigma(n)}]$.

Proposition 4.1 (Invariance of V^t_π and V^t_*)**.** *Assume that the energy function is* $\mathrm{O}(m) \times \mathrm{S}_n$*–invariant as follows:*

$$E((R,\sigma)\circ\mathbf{x}) = E(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}, (R,\sigma) \in \mathrm{O}(m)\times\mathrm{S}_n.$$

then both the value function of VGS and the optimal value function preserves $\mathrm{O}(m) \times \mathrm{S}_n$*-invariance:*

$$V^t_\pi((R,\sigma)\circ\mathbf{x}_t) = V^t_\pi(\mathbf{x}_t), \quad V^t_*((R,\sigma)\circ\mathbf{x}_t) = V^t_*(\mathbf{x}_t),$$

for all $\mathbf{x}_t \in \mathcal{X}$*,* $(R,\sigma) \in \mathrm{O}(m)\times\mathrm{S}_n$*.*

Proof sketch. The claim for V^t_π follows by backward induction from $t = T$ to 0, using that the gradient of a G–invariant function is G–equivariant when G acts orthogonally on $\mathcal{X}$ (Papamakarios et al., 2021,

Table 1: Results of n-body particle system experiments. The metrics are evaluated with 10,000 samples. We report the average and standard deviation over three random seeds. Metrics shown in **bold** denote the best mean performance, while underlined entries indicate means are not statistically distinguishable from the best mean, using a one-sided Welch's t-test with a significance threshold of $p < 0.1$. $*$ indicates divergent training. For details, see Appendix C.3.1.

Energy	DW-4 (D=8)			LJ-13 (D=39)			LJ-55 (D=165)		
Metric	TVD-D	TVD-E	$\mathcal{W}_2$	TVD-D	TVD-E	$\mathcal{W}_2$	TVD-D	TVD-E	$\mathcal{W}_2$
FAB	$0.073_{\pm 0.004}$	$0.154_{\pm 0.011}$	$1.931_{\pm 0.063}$	$0.278_{\pm 0.072}$	$0.950_{\pm 0.044}$	$5.350_{\pm 0.231}$	$0.148_{\pm 0.006}$	$1.000_{\pm 0.000}$	$17.217_{\pm 0.466}$
iDEM	$0.061_{\pm 0.003}$	$0.118_{\pm 0.013}$	$1.597_{\pm 0.011}$	$0.029_{\pm 0.005}$	$0.154_{\pm 0.024}$	$\mathbf{3.976_{\pm 0.004}}$	$0.160_{\pm 0.104}$	$0.951_{\pm 0.086}$	$16.460_{\pm 1.123}$
DiKL	$0.061_{\pm 0.009}$	$0.193_{\pm 0.079}$	$\mathbf{1.566_{\pm 0.013}}$	$0.070_{\pm 0.016}$	$0.488_{\pm 0.111}$	$4.058_{\pm 0.035}$	$0.279_{\pm 0.121}$	$1.000_{\pm 0.000}$	$18.050_{\pm 2.158}$
PIS	$0.144_{\pm 0.015}$	$0.388_{\pm 0.014}$	$2.107_{\pm 0.120}$	$0.285_{\pm 0.069}$	$0.523_{\pm 0.045}$	$4.187_{\pm 0.296}$	$*$	$*$	$*$
DDS	$0.259_{\pm 0.024}$	$0.573_{\pm 0.062}$	$3.327_{\pm 0.250}$	$*$	$*$	$*$	$*$	$*$	$*$
GFN-DB	$0.476_{\pm 0.005}$	$0.891_{\pm 0.092}$	$2.739_{\pm 0.938}$	$*$	$*$	$*$	$*$	$*$	$*$
GFN-SubTB	$0.218_{\pm 0.007}$	$0.585_{\pm 0.011}$	$1.606_{\pm 0.006}$	$0.385_{\pm 0.007}$	$0.999_{\pm 0.001}$	$5.210_{\pm 0.036}$	$*$	$*$	$*$
GFN-TB	$0.350_{\pm 0.021}$	$0.743_{\pm 0.020}$	$1.687_{\pm 0.046}$	$0.325_{\pm 0.030}$	$0.982_{\pm 0.002}$	$4.914_{\pm 0.185}$	$*$	$*$	$*$
VGS (Ours)	$\mathbf{0.053_{\pm 0.008}}$	$\mathbf{0.097_{\pm 0.016}}$	$1.587_{\pm 0.007}$	$\mathbf{0.025_{\pm 0.001}}$	$\mathbf{0.109_{\pm 0.024}}$	$4.029_{\pm 0.011}$	$\mathbf{0.061_{\pm 0.023}}$	$\mathbf{0.799_{\pm 0.297}}$	$\mathbf{15.906_{\pm 0.261}}$

Lemma 2). The statement for V_*^t follows from Eq. 6. See Appendix A.5 for details.

An analogous result holds when particles are distinguishable and the symmetry reduces to $\mathrm{E}(m)$–invariance. The corresponding proposition and proof are also given in Appendix A.5

The results of this section establish $\mathrm{O}(m) \times \mathrm{S}_n$ invariance on $\mathcal{X}$, which lifts to $\mathrm{E}(m) \times \mathrm{S}_n$ invariance on the full space $\mathbb{R}^{m \times n}$.

4.3 Invariant Architectures for VGS

Building on these theoretical results, we consider two invariant architectures in our experiments: an $\mathrm{E}(m)$-invariant graph neural network (IGNN) and an $\mathrm{E}(m) \times \mathrm{S}_n$-invariant multilayer perceptron (IMLP). IGNN can be scaled to general systems when only $\mathrm{E}(m)$ invariance is present. IMLP is a simple and efficient choice when both $\mathrm{E}(m)$ and S_n invariances hold.

$\mathrm{E}(m)$-Invariant Graph Neural Network. IGNN is a simplification of EGNN, introduced in Satorras et al. (2021). For particles $x_i \in \mathbb{R}^m$, let the pairwise distance be $d_{ij} = \|x_i - x_j\|$. Given initial node embeddings $h^0 = \{h_i^0\}_{i=1}^n$, a layer updates as follows:

$$m_{ij} = \Phi_e(h_i^l, h_j^l, d_{ij}), \qquad h_i^{l+1} = \Phi_h\left(h_i^l, \sum_{j \neq i} m_{ij}\right).$$

Aggregating the final-layer node embeddings and applying MLPs yields an $\mathrm{E}(m)$-invariant output. To handle distinguishable particles, per-particle features (e.g., atom types) can be embedded in h^0. In our experiments where particles are indistinguishable, we use a uniform h^0 for both EGNN and IGNN, resulting in $\mathrm{E}(m) \times \mathrm{S}_n$ equivariance and invariance, respectively. A limitation of this architecture is the $\mathcal{O}(n^2)$ computational scaling induced by all-pairs message passing.

$\mathrm{E}(m) \times \mathrm{S}_n$-Invariant Multilayer Perceptron. IMLP is an MLP that takes sorted pairwise distances $\{d_{ij}\}_{i<j\leq n}$ as input instead of particle coordinates $\mathbf{x}$. By construction, the input representation, and thus the output of the network, is $\mathrm{E}(m) \times \mathrm{S}_n$ invariant. Furthermore, replacing $\mathbf{x}$ with the sorted distances incurs no loss of information up to an $\mathrm{E}(m) \times \mathrm{S}_n$ transformation under mild conditions. This follows from (Boutin and Kemper, 2004, Theorem 2.6), stating that the distance multiset almost always reconstructs the spatial configuration if $n \geq m + 2$, a condition satisfied across all of our evaluated benchmark systems. Although the input dimension scales as $\mathcal{O}(n^2)$, the forward cost is governed mainly by network width and depth. In the LJ-13 to LJ-55 setting, we observed that it is sufficient to only double the hidden dimension, even though the input grew by about 19×. This computational advantage is reflected in the sampling time experiment results in Sec. 5.1.

5 Experiments

We evaluate VGS on two tasks: sampling equilibrium states of n-body systems (Sec. 5.1) and sampling from synthetic distributions (Sec. 5.2). In this section, we provide a brief overview of the experimental setups and present the main results. Detailed descriptions of the target distributions are provided in Appendix C.1, and the evaluation metrics are provided in Appendix C.2. Additional experimental details and hyperparameters are given in Appendix C.3.

Table 2: Sampling time measured in the LJ-55 experiments. We measured the average time required for batch size 512 sample generation over 10 trials on a single 24GB RTX 4090 GPU.

	T	time/step (ms)	time (s)	# Param.
FAB	1[2]	5907.94	5.91±0.51	5.44M
iDEM	1000	135.39	135.39±0.04	0.58M
DiKL	1	368.41	0.37±0.00	2.08M
PIS	200	136.91	27.38±0.05	0.58M
DDS	200	136.91	27.38±0.05	0.58M
GFN[3]	100	134.51	13.45±0.01	0.58M
VGS-IMLP	100	**1.28**	**0.13±0.00**	10.05M
VGS-IGNN	100	124.43	12.43±0.01	0.32M

5.1 Sampling n-Body Particle Systems

Target Distributions. We evaluate our method on three standard benchmark n-body systems: DW-4, LJ-13, and LJ-55 (Köhler et al., 2020). DW-4 is a 2D 4-particle system with a double-well potential, while LJ-13 and LJ-55 are 3D systems with 13 and 55 particles interacting via the Lennard-Jones potential. The input space dimensionality of DW-4, LJ-13, and LJ-55 is 8, 39, and 165, respectively. The potential function of DW-4 exhibit $\mathrm{E}(2) \times \mathrm{S}_4$-invariance, while the potentials of LJ-13 and LJ-55 have $\mathrm{E}(3) \times \mathrm{S}_n$-invariance ($n = 13, 55$).

Performance Metrics. Following He et al. (2024), we use three metrics: total variation distance of interatomic distances (TVD-D ↓), total variation distance of energy (TVD-E ↓), and Wasserstein-2 distance ($\mathcal{W}^2$ ↓) between test and generated samples. TVD-D and TVD-E are computed from histograms of interatomic distances and energy, respectively. Samples are normalized to zero mean when computing $\mathcal{W}^2$.

Baselines. Baselines include FAB with an SE(3)-augmented coupling flow (Midgley et al., 2023b,a), iDEM (Akhound-Sadegh et al., 2024), and DiKL (He et al., 2024), all explicitly designed for particle system benchmarks by leveraging symmetry. Additionally, we include PIS (Zhang and Chen, 2022) and DDS (Vargas et al., 2023) as implemented in Akhound-Sadegh et al. (2024). We also evaluate GFlowNet (GFN) variants, including GFN-DB (Bengio et al., 2023), GFN-TB (Lahlou et al., 2023), and GFN-SubTB (Zhang et al., 2023), using the codebase from Sendera et al. (2024). However we exclude their proposed variant (GFN-TB-Imp in Sec. 5.2) as its MCMC exploration caused divergence. For these additional baselines, we use an EGNN-parametrized equivariant policy configured identically to iDEM. We omit adjoint sampling (Havens et al., 2025) due to the lack of publicly available code for the particle systems.

[2]FAB uses 1 step normalizing flow followed by multiple MCMC steps

[3]Includes GFN-DB, GFN-TB, and GFN-SubTB.

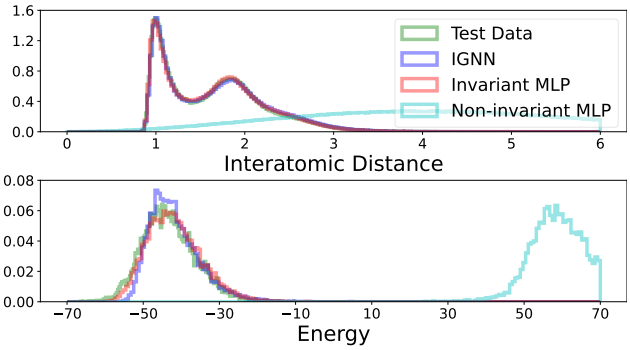

Figure 2: Interatomic Distance (**Top**) and Energy (**Bottom**) histograms from LJ-13. Samples from VGS with an invariant MLP, IGNN, and a non-invariant MLP are compared against the test samples. VGS generates accurate samples with invariant networks.

Table 3: Ablation study on training techniques in VGS. Experiments are conducted in the LJ-13 system.

	TVD-D	TVD-E	$\mathcal{W}_2$
VGS	0.025±0.00	0.109±0.02	4.029±0.01
− Double Value	0.023	0.144	4.030
− TD(λ)	0.040	0.278	4.098
− Exploration Policy	0.063	0.346	4.160
VGS-IGNN	0.023	0.111	4.044

Results. Experimental results are shown in Table 1. VGS outperforms all baseline methods across all particle systems in terms of the TVD-D and TVD-E metrics. For the $\mathcal{W}^2$ metric, DiKL and iDEM achieve the best performance on DW-4 and LJ-13, respectively, while VGS records the lowest average $\mathcal{W}^2$ on LJ-55, demonstrating its scalability to higher dimensions. The results for VGS are obtained using IMLP, double value functions, and an exploration policy. We set $T = 100$ and $\lambda = 0.9$ for LJ-13 and LJ-55, and $T = 50$ and $\lambda = 0$ for DW-4. Sample buffers are not used for particle system experiments.

Consistent with the findings of Akhound-Sadegh et al. (2024), the training of PIS and DDS tends to diverge in higher dimensions. Similar divergent behavior is also observed in the GFN baselines, which, to the best of our knowledge, we are the first to evaluate on particle system benchmarks. We consider a sampler to have diverged if its TVD-D metric is higher than that of an untrained sampler. Despite explicitly incorporating symmetry via the EGNN, these additional baselines yielded poor performance.

Sampling Time. Table 2 reports the sampling

Table 4: Results of synthetic distribution experiments. The metrics are evaluated with 2,000 samples. We report the average and standard deviation over three random seeds. Metrics shown in **bold** denote the best mean performance, while underlined entries indicate means are not statistically distinguishable from the best mean, using a one-sided Welch's t-test with a significance threshold of $p < 0.1$. For details, see Appendix C.2 and Appendix C.3.2.

Energy	25GMM (D=2)			Funnel (D=10)			Manywell (D=32)		
Metric	$\lvert\Delta \log Z_r\rvert$	$\lvert\Delta \log Z_f\rvert$	$\mathcal{W}_2$	$\lvert\Delta \log Z_r\rvert$	$\lvert\Delta \log Z_f\rvert$	$\mathcal{W}_2$	$\lvert\Delta \log Z_r\rvert$	$\lvert\Delta \log Z_f\rvert$	$\mathcal{W}_2$
SMC	$0.345_{\pm 0.073}$		$5.207_{\pm 0.286}$	$0.614_{\pm 0.100}$		$31.495_{\pm 15.342}$	$30.170_{\pm 0.557}$		$5.882_{\pm 0.237}$
AFT	$0.092_{\pm 0.052}$		$4.930_{\pm 0.175}$	$0.197_{\pm 0.122}$		$23.516_{\pm 5.167}$	$13.692_{\pm 4.730}$		$8.905_{\pm 0.384}$
CRAFT	$0.131_{\pm 0.108}$		$4.938_{\pm 0.257}$	$0.126_{\pm 0.059}$		$26.269_{\pm 1.960}$	$13.076_{\pm 4.453}$		$9.244_{\pm 0.579}$
PIS-LV	$1.020_{\pm 0.006}$	$0.248_{\pm 0.050}$	$4.577_{\pm 0.039}$	$0.357_{\pm 0.373}$	$0.253_{\pm 0.037}$	$22.567_{\pm 2.753}$	$0.721_{\pm 0.058}$	$\mathbf{0.134_{\pm 0.078}}$	$5.462_{\pm 0.029}$
DIS-LV	$1.083_{\pm 0.004}$	$0.034_{\pm 0.018}$	$4.656_{\pm 0.044}$	$\mathbf{0.035_{\pm 0.019}}$	$0.337_{\pm 0.014}$	$20.630_{\pm 1.471}$	$2.172_{\pm 0.302}$	$1.253_{\pm 0.642}$	$6.144_{\pm 0.025}$
DDS-LV	$1.018_{\pm 0.002}$	-	$4.678_{\pm 0.087}$	$0.070_{\pm 0.017}$	-	$20.681_{\pm 3.627}$	$2.281_{\pm 0.314}$	-	$6.021_{\pm 0.097}$
GFN-DB	$0.559_{\pm 0.053}$	$0.071_{\pm 0.050}$	$4.976_{\pm 0.159}$	$0.552_{\pm 0.095}$	$0.040_{\pm 0.017}$	$28.070_{\pm 9.069}$	$8.321_{\pm 5.800}$	$7.057_{\pm 2.670}$	$6.719_{\pm 0.531}$
GFN-TB	$1.035_{\pm 0.015}$	$0.014_{\pm 0.011}$	$4.685_{\pm 0.055}$	$0.284_{\pm 0.073}$	$0.016_{\pm 0.015}$	$27.684_{\pm 9.120}$	$2.726_{\pm 0.109}$	$0.198_{\pm 0.184}$	$6.161_{\pm 0.027}$
GFN-SubTB	$0.016_{\pm 0.009}$	$0.013_{\pm 0.016}$	$1.334_{\pm 0.090}$	$0.384_{\pm 0.039}$	$0.013_{\pm 0.010}$	$20.683_{\pm 2.514}$	$2.660_{\pm 0.045}$	$0.337_{\pm 0.051}$	$6.092_{\pm 0.013}$
GFN-TB-Imp	$0.016_{\pm 0.009}$	$0.018_{\pm 0.004}$	$1.186_{\pm 0.221}$	$0.154_{\pm 0.024}$	$0.028_{\pm 0.016}$	$23.497_{\pm 3.416}$	$\mathbf{0.153_{\pm 0.051}}$	$0.243_{\pm 0.297}$	$\mathbf{5.330_{\pm 0.030}}$
VGS (Ours)	$\mathbf{0.003_{\pm 0.005}}$	$0.022_{\pm 0.020}$	$1.253_{\pm 0.107}$	$0.203_{\pm 0.082}$	$0.040_{\pm 0.022}$	$17.506_{\pm 0.769}$	$2.427_{\pm 0.812}$	$3.583_{\pm 2.775}$	$5.666_{\pm 0.0.069}$
+Buffer	$0.021_{\pm 0.016}$	$0.021_{\pm 0.013}$	$\mathbf{1.181_{\pm 0.067}}$	$0.168_{\pm 0.087}$	$0.046_{\pm 0.031}$	$21.787_{\pm 4.134}$	$1.360_{\pm 1.565}$	$2.562_{\pm 0.354}$	$5.474_{\pm 0.032}$
+TD(λ)	$0.006_{\pm 0.001}$	$\mathbf{0.012_{\pm 0.004}}$	$1.214_{\pm 0.104}$	$0.208_{\pm 0.070}$	$\mathbf{0.005_{\pm 0.001}}$	$\mathbf{16.325_{\pm 0.797}}$	$0.845_{\pm 0.050}$	$0.517_{\pm 0.277}$	$5.492_{\pm 0.012}$

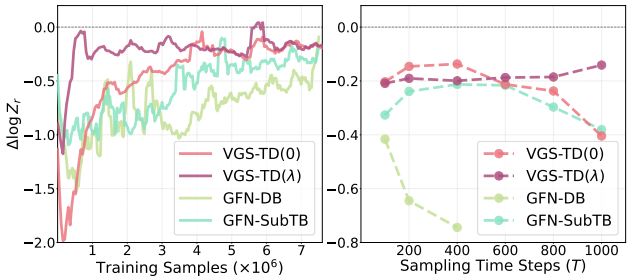

Figure 3: (**Left**) Evolution of $\Delta \log Z_r$ during training. (**Right**) Effect of varying T. VGS can reliably learn with a large number of time steps using TD(λ). GFN-DB diverged when $T > 400$. Experiments are conducted on Funnel.

times measured on LJ-55, where scalability poses a challenge. VGS-IMLP achieves the fastest per-step and total sampling times among all baselines, despite utilizing the largest number of parameters. Specifically, its per-step sampling is over 100× faster than other models. In terms of total time, it is faster than the single-step generator DiKL (He et al., 2024) and approximately $1,000\times$ faster than iDEM (Akhound-Sadegh et al., 2024), the best-performing baseline. Note that VGS uses the network's gradient during sampling, which incurs additional computational overhead. However, our results indicate that the efficiency of IMLP over equivariant networks outweighs this overhead. VGS-IGNN demonstrates a moderate sampling time; while slower than single-step generators, it remains faster than all other multi-step baselines.

Importance of Invariance. Fig. 2 visualizes the energy and interatomic distance histograms for LJ-13 from VGS. Three different neural network architectures, IGNN, IMLP, and conventional non-invariant MLP, are compared. The distributions of distances and energy align well with the test data only when the invariance is respected using IGNN and IMLP.

RL Techniques in VGS. We confirm the benefit of the RL techniques introduced in Sec. 3.4 by ablating each method (see Table 3). Removing the off-policy exploration method brings the most significant degradation in performance metrics. Using TD(λ) and the double value function techniques are also important in achieving the best performance.

5.2 Sampling from Synthetic Distributions

Target Distributions. We use three distributions as our sampling benchmarks: a 25-component Gaussian Mixture Model (25GMM), a funnel distribution, and a manywell distribution.

Baselines. We compare VGS against four classes of baseline samplers on each target distribution: Sequential Monte Carlo–based methods (SMC (Del Moral et al., 2006), AFT (Arbel et al., 2021), and CRAFT (Matthews et al., 2023)); SDE–based methods (PIS, DIS (Berner et al., 2022), and DDS with log-variance objective (Richter and Berner, 2024)); and GFlowNets (GFN-DB (Bengio et al., 2023), GFN-TB (Lahlou et al., 2023), GFN-SubTB (Zhang et al., 2023) and GFN-TB-Imp (Sendera et al., 2024)).

Performance Metrics. The comparisons are

made based on three metrics: the bias in reverse importance-weighted estimation of log partition function ($|\Delta \log Z_r| \downarrow$), the bias in forward importance-weighted estimation of log partition function ($|\Delta \log Z_f| \downarrow$) (Blessing et al., 2024) and the Wasserstein-2 distance ($\mathcal{W}_2 \downarrow$).

Results. We evaluate VGS trained with TD(0), TD(λ), and sample buffers, with the results summarized in Table 4. We set $T = 50$ for 25GMM, and $T = 100$ for Funnel and Manywell. For all TD(λ) experiments, we use $\lambda = 0.9$.

VGS-TD(0) and VGS-TD(λ) achieve performance comparable to baseline models that do not employ additional exploration strategies (e.g., PIS-LV, GFN-TB). When a sample buffer is used, VGS shows results that are on par with baselines incorporating exploration mechanisms (e.g., GFN-TB-Imp).

Benefit of TD(λ). Fig. 3 compares TD(0) and TD(λ) applied to VGS. For comparison, we also present GFN-DB and GFN-SubTB. Specifically, TD(0) and GFN-DB are aligned in that they both minimize single-step differences, whereas TD(λ) and GFN-SubTB share the common feature of operating over subtrajectories. VGS-TD(0) exhibits more stable training dynamics than GFN-DB, as expected from the use of target networks. TD(λ) demonstrates improved performance at larger T, addressing the limitations of TD(0) in long-term credit assignment.

Training with Buffer. We utilize sampler buffer as introduced in Sec. 3.4. To quantify the impact of the buffer, we compare VGS with and without the buffer in Table 4. We adopt the buffer design and MCMC exploration strategy from Sendera et al. (2024). The performance gain from the buffer is not always consistent, possibly due to sensitivity to the hyperparameters of the MCMC exploration.

6 Related Work

VGS builds on diffusion-based samplers, where the learned trajectory corresponds to the reversed diffusion process. Following diffusion generative models (Ho et al., 2020; Song et al., 2021), samplers have been developed to model the stochastic bridge between initial and target distributions. Some train neural SDEs (Zhang and Chen, 2022; Berner et al., 2022; Vargas et al., 2023; Richter and Berner, 2024; Havens et al., 2025), while others model PDEs (Shi et al., 2024; Sun et al., 2024). A central challenge is estimating the score of diffused densities, since denoising score matching is inapplicable without training data. Alternatives rely on various mathematical formulations (Akhound-Sadegh et al., 2024; Phillips et al., 2024; Huang et al., 2024; McDonald and Barron, 2022; Wang et al., 2024; Chen et al., 2024; He et al., 2024; Vargas et al., 2024). GFlowNets view sampling as a sequential decision-making problem (Bengio et al., 2021, 2023; Malkin et al., 2022; Lahlou et al., 2023; Madan et al., 2023). Similar to GFlowNets, VGS sidesteps score estimation via value-based RL, estimating the value function corresponding to the energy of the target-reference density ratio. The idea of leveraging value functions and TD learning in diffusion was explored in Yoon et al. (2024).

7 Conclusion

This paper introduces the Value Gradient Sampler (VGS), a diffusion sampler defined by value functions and trained via RL. VGS is particularly effective and efficient when sampling from target densities that exhibit invariance symmetries.

Limitations. First, VGS needs to compute gradients during sampling, which imposes overhead in both computation and memory. However, in particle system applications, this drawback is outweighed by the ability to use more efficient network architectures. Second, we were not able to experiment with a recent large-scale conformer sampling benchmark (Havens et al., 2025) due to computational constraints. Third, no theoretical guarantee is provided for VGS. We left larger-scale applications and rigorous mathematical analysis as future work.

Acknowledgements

H. Hwang, H. Jeong, C. Park, S. Kweon, and F. Park were supported in part by IITP-MSIT under Grants 2022-220480 and RS-2022-II220480 (Training and Inference Methods for Goal Oriented AI Agents), and Grant RS-2024-00436680 (Collaborative Research Projects with Microsoft Research); KIAT under Grant P0020536 (HRD Program for Industrial Innovation); MOTIR under Grants RS-2025-25462891 and RS-2025-25460896; Hyundai Motor Company and Kia; Microsoft Research Asia; and SNU-IPAI, SNU-AIIS, SNU-IAMD, the SNU BK21+ Program in Mechanical Engineering, and the SNU Institute for Engineering Research. S. Yoon was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2020-II201336, Artificial Intelligence Graduate School Program (UNIST) and No. RS-2025-25442824, AI Star Fellowship Program (Ulsan National Institute of Science and Technology)), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00408003), and the Center for Advanced

Computation at Korea Institute for Advanced Study.

References

Akhound-Sadegh, T., Rector-Brooks, J., Bose, 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. In *Forty-first International Conference on Machine Learning*.

Arbel, M., Matthews, A. G. D. G., and Doucet, A. (2021). Annealed flow transport monte carlo.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394. Curran Associates, Inc.

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

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

Blessing, D., Jia, X., Esslinger, J., Vargas, F., and Neumann, G. (2024). Beyond elbos: A large-scale evaluation of variational methods for sampling.

Boutin, M. and Kemper, G. (2004). On reconstructing n-point configurations from the distribution of distances or areas. *Advances in Applied Mathematics*, 32(4):709–735.

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

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

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.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. (2020). Se(3)-transformers: 3d roto-translation equivariant attention networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1970–1981. Curran Associates, Inc.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR.

Geiger, M. and Smidt, T. (2022). e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR.

Havens, A. J., Miller, B. K., Yan, B., Domingo-Enrich, C., Sriram, A., Levine, D. S., Wood, B. M., Hu, B., Amos, B., Karrer, B., Fu, X., Liu, G.-H., and Chen, R. T. Q. (2025). Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. In *Forty-second International Conference on Machine Learning*.

He, J., Chen, W., Zhang, M., Barber, D., and Hernández-Lobato, J. M. (2024). Training neural samplers with reverse diffusive kl divergence. *arXiv preprint arXiv:2410.12456*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

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

Huang, X., Dong, H., HAO, Y., Ma, Y., and Zhang, T. (2024). Reverse diffusion monte carlo. In *The Twelfth International Conference on Learning Representations*.

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

Karczewski, R., Souza, A. H., and Garg, V. (2024). On the generalization of equivariant graph neural networks. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 23159–23186. PMLR.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Klein, L., Krämer, A., and Noé, F. (2023). Equivariant flow matching.

Köhler, J., Klein, L., and Noé, F. (2019). Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*.

Köhler, J., Klein, L., and Noé, F. (2020). Equivariant flows: Exact likelihood generative learning for symmetric densities.

Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-Garcıa, A., Ezzine, L. N., Bengio, Y., and Malkin, N. (2023). A theory of continuous generative flow networks. In *International Conference on Machine Learning*, pages 18269–18300. PMLR.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. (2023). Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022). Trajectory balance: Improved credit assignment in gflownets. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances*

in Neural Information Processing Systems, volume 35, pages 5955–5967. Curran Associates, Inc.

Matthews, A. G. D. G., Arbel, M., Rezende, D. J., and Doucet, A. (2023). Continual repeated annealed flow transport monte carlo.

McDonald, C. and Barron, A. (2022). Proposal of a score based approach to sampling using monte carlo estimation of score and oracle access to target density. *arXiv preprint arXiv:2212.03325*.

Midgley, L., Stimper, V., Antorán, J., Mathieu, E., Schölkopf, B., and Hernández-Lobato, J. M. (2023a). Se (3) equivariant augmented coupling flows. *Advances in Neural Information Processing Systems*, 36:79200–79225.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Neal, R. M. (2003). Slice sampling. the annals of statistics. *Project Euclid*.

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

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

Phillips, A., Dau, H.-D., Hutchinson, M. J., Bortoli, V. D., Deligiannidis, G., and Doucet, A. (2024). Particle denoising diffusion sampler. In *Forty-first International Conference on Machine Learning*.

Richter, L. and Berner, J. (2024). Improved sampling via learned diffusions. In *The Twelfth International Conference on Learning Representations*.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E(n) equivariant graph neural networks. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9323–9332. PMLR.

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

Shi, Z., Yu, L., Xie, T., and Zhang, C. (2024). Diffusion-pinn sampler. *arXiv preprint arXiv:2410.15336*.

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

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

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. (2018). Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*.

Thrun, S. and Schwartz, A. (2014). Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pages 255–263. Psychology Press.

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

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

Wang, Y., Guo, L., Wu, H., and Zhou, T. (2024). Energy based diffusion generator for efficient sampling of boltzmann distributions. *arXiv preprint arXiv:2401.02080*.

Yoon, S., Hwang, H., Kwon, D., Noh, Y.-K., and Park, F. C. (2024). Maximum entropy inverse reinforcement learning of diffusion models with energy-based models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhang, D., Chen, R. T., Liu, C.-H., Courville, A., and Bengio, Y. (2023). Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *arXiv preprint arXiv:2310.02679*.

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

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. No, but the code will be open-sourced once the manuscript is published.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes.
   (b) Complete proofs of all theoretical results. Yes.
   (c) Clear explanations of any assumptions. Yes.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator if your work uses existing assets. Yes.
   (b) The license information of the assets, if applicable. Not Applicable.
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable.
   (d) Information about consent from data providers/curators. Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

Supplementary Materials for Value Gradient Sampler: Learning Invariant Value Functions for Equivariant Diffusion Sampling

A Proofs and Derivations

A.1 Derivation of Optimal control objective

We start from the joint KL-divergence minimization objective:

$$\min_{\pi\in\Pi} \mathrm{KL}(\pi(\mathbf{x}_{0:T})\|\tilde{q}(\mathbf{x}_{0:T})) = \min_{\pi\in\Pi} \mathbb{E}_{\pi(\mathbf{x}_{0:T})}\left[\log \frac{\pi(\mathbf{x}_{0:T})}{\tilde{q}(\mathbf{x}_{0:T})}\right] \tag{12}$$

Substituting our choice of the joint target distribution $\tilde{q}(\mathbf{x}_{0:T}) = \tilde{\pi}(\mathbf{x}_{0:T})q(\mathbf{x}_T)/\tilde{\pi}(\mathbf{x}_T)$ and using Eq. 1, and dropping constant terms, we obtain

$$\min_{\pi\in\Pi} \mathbb{E}_{\pi(\mathbf{x}_{0:T})}\left[\log \frac{\pi(\mathbf{x}_0)}{\tilde{\pi}(\mathbf{x}_0)} + \sum_{t=0}^{T-1} \log \frac{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t)} + E(\mathbf{x}_T) + \log \tilde{\pi}(\mathbf{x}_T)\right] \tag{13}$$

From our construction of π (Eq. 3) and the reference distribution $\tilde{\pi}$, we have $\pi(\mathbf{x}_0)/\tilde{\pi}(\mathbf{x}_0) = 1$, and

$$\mathbb{E}_{\pi_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t)}\left[\log \frac{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t)}\middle|\mathbf{x}_t\right] = \mathbb{E}_{\epsilon_t\sim\mathcal{N}(0,I)}\left[-\frac{\|\sigma_t\epsilon_t\|^2}{2\sigma_t^2} + \frac{\|\mu^t(\mathbf{x}_t) + \sigma_t\epsilon_t\|^2}{2\sigma_t^2}\right] = \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2}, \tag{14}$$

Therefore, by the tower property, Eq. 13 can be rewritten as

$$\min_{\pi\in\Pi} \mathbb{E}_{\pi(\mathbf{x}_{0:T})}\left[\sum_{t=0}^{T-1} \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + \tilde{E}(\mathbf{x}_T)\right], \tag{15}$$

where $\tilde{E}(\mathbf{x}_T) = E(\mathbf{x}_T) + \log \tilde{\pi}(\mathbf{x}_T)$.

A.2 Derivation of the Optimal Value Function

For the proof, using Eq. 14 we rewrite the value function back to its general form:

$$\begin{aligned}
V_\pi^t(\mathbf{x}_t) &= \mathbb{E}_{\pi(\cdot|\mathbf{x}_t)}\left[\sum_{i=0}^{T-t-1} \frac{\|\mu^{t+i}(\mathbf{x}_{t+i})\|^2}{2\sigma_{t+i}^2} + \tilde{E}(\mathbf{x}_T)\middle|\mathbf{x}_t\right] && (16)\\
&= \mathbb{E}_{\pi(\cdot|\mathbf{x}_t)}\left[\sum_{i=0}^{T-t-1} \log \frac{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t)} + \tilde{E}(\mathbf{x}_T)\middle|\mathbf{x}_t\right] && (17)\\
&= \mathbb{E}_{\pi(\cdot|\mathbf{x}_t)}\left[\log \frac{\pi(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}{\tilde{\pi}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)} + \log \frac{\tilde{\pi}(\mathbf{x}_T)}{q(\mathbf{x}_T)} - \log Z\right] && (18)\\
&= \mathbb{E}_{\pi(\cdot|\mathbf{x}_t)}\left[\log \frac{\pi(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}{\tilde{q}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)} + \log \frac{\tilde{\pi}(\mathbf{x}_t)}{\tilde{q}(\mathbf{x}_t)} - \log Z\right]. && (19)
\end{aligned}$$

The last equality follows from Bayes' rule applied to our choice of the joint target distribution, $\tilde{q}(\mathbf{x}_{t:T}) = \tilde{\pi}(\mathbf{x}_{t:T})q(\mathbf{x}_T)/\tilde{\pi}(\mathbf{x}_T)$.

Now, by the definition of the optimal value function,

$$V_*^t(\mathbf{x}_t) = \min_{\pi(\cdot|\mathbf{x}_t)\in\Pi} V_\pi^t(\mathbf{x}_t) \tag{20}$$

$$= \log\frac{\tilde{\pi}(\mathbf{x}_t)}{\tilde{q}(\mathbf{x}_t)} - \log Z + \min_{\pi(\cdot|\mathbf{x}_t)\in\Pi} \mathbb{E}_{\pi(\cdot|\mathbf{x}_t)}\left[\log\frac{\pi(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}{\tilde{q}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}\right] \tag{21}$$

$$= \log\frac{\tilde{\pi}(\mathbf{x}_t)}{\tilde{q}(\mathbf{x}_t)} - \log Z + \min_{\pi(\cdot|\mathbf{x}_t)\in\Pi} \mathrm{KL}\big(\pi(\mathbf{x}_{t+1:T}|\mathbf{x}_t)\|\tilde{q}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)\big) \tag{22}$$

$$= \log\frac{\tilde{\pi}(\mathbf{x}_t)}{\tilde{q}(\mathbf{x}_t)} - \log Z. \tag{23}$$

The last step uses the assumption that $\tilde{q}(\mathbf{x}_{t+1:T}|\mathbf{x}_t) \in \Pi$, so the minimum KL divergence is attained at zero.

Exponentiating both sides and rearranging terms gives the desired relation:

$$\frac{\tilde{q}(\mathbf{x}_t)}{\tilde{\pi}(\mathbf{x}_t)} = \frac{1}{Z}\exp\left(-V_*^t(\mathbf{x}_t)\right). \tag{24}$$

A.3 Derivation of VGS

In this section, we present a detailed derivation of VGS. To obtain an approximate solution to the optimization problem in Eq. 7, we apply a first-order Taylor expansion of V_π^{t+1} around $\alpha_t\mathbf{x}_t$:

$$\mathbb{E}_{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)}\left[\frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + V_\pi^{t+1}(\mathbf{x}_{t+1})\middle|\mathbf{x}_t\right] = \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + \mathbb{E}_{\epsilon_t\sim\mathcal{N}(0,I)}\left[V_\pi^{t+1}(\alpha_t\mathbf{x}_t + \mu^t(\mathbf{x}_t) + \sigma_t\epsilon_t)\right] \tag{25}$$

$$\approx \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + \mathbb{E}_{\epsilon_t\sim\mathcal{N}(0,I)}\left[V_\pi^{t+1}(\alpha_t\mathbf{x}_t) + (\mu^t(\mathbf{x}_t) + \sigma_t\epsilon_t)^T\nabla_{\alpha_t\mathbf{x}_t}V_\pi^{t+1}(\alpha_t\mathbf{x}_t)\right] \tag{26}$$

$$= \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + V_\pi^{t+1}(\alpha_t\mathbf{x}_t) + \mu^t(\mathbf{x}_t)^T\nabla_{\alpha_t\mathbf{x}_t}V_\pi^{t+1}(\alpha_t\mathbf{x}_t). \tag{27}$$

Substituting this approximation into Eq. 7 and dropping constant terms, we obtain the following surrogate problem:

$$\min_{\mu^t(\mathbf{x}_t)} \frac{\|\mu^t(\mathbf{x}_t)\|^2}{2\sigma_t^2} + \mu^t(\mathbf{x}_t)^T\nabla_{\alpha_t\mathbf{x}_t}V_\pi^{t+1}(\alpha_t\mathbf{x}_t). \tag{28}$$

Setting the derivative with respect to $\mu^t(\mathbf{x}_t)$ to zero yields the closed-form optimal drift:

$$\mu^t(\mathbf{x}_t) = -\sigma_t^2\nabla_{\alpha_t\mathbf{x}_t}V_\pi^{t+1}(\alpha_t\mathbf{x}_t). \tag{29}$$

We use this first-order approximation throughout the main text and experiments. However, a second-order Taylor expansion in Eq. 26 also admits an analytic solution, which we include here for completeness. In that case, the optimal drift is

$$\mu^t(\mathbf{x}_t) = -\left(I + \sigma_t^2\nabla^2_{\alpha_t\mathbf{x}_t}V_\pi^{t+1}(\alpha_t\mathbf{x}_t)\right)^{-1}\sigma_t^2\nabla_{\alpha_t\mathbf{x}_t}V_\pi^{t+1}(\alpha_t\mathbf{x}_t). \tag{30}$$

A.4 Off-policy TD(λ)

Recall that the TD(λ) target for on-policy samples $\mathbf{x}_{0:T} \sim \pi_{\phi^-}(\mathbf{x}_{0:T})$ is given by Eq. 11. Now suppose a trajectory is collected under a different policy π_{expl}, i.e., $\mathbf{x}_{0:T} \sim \pi_{\text{expl}}(\mathbf{x}_{0:T})$. Using importance sampling, we can construct an unbiased estimator of the on-policy TD(λ) target's expectated value $\mathbb{E}_{\pi_{\phi^-}(\cdot|\mathbf{x}_t)}\left[\hat{V}^t_{\mathrm{TD}(\lambda)}(\mathbf{x}_{t:T})\right]$ based on samples

from π_{expl}. Building on Eq. 11 and applying the tower property, we have

$$\mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}[\hat{V}^t_{\text{TD}(\lambda)}(\mathbf{x}_{t:T})] \tag{31}$$

$$= \mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}\left[V^t_{\phi^-}(\mathbf{x}_t) + \sum_{i=0}^{T-t-1} \lambda^i \delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}_{t+i+1})\right] \tag{32}$$

$$= V^t_{\phi^-}(\mathbf{x}_t) + \mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}\left[\sum_{i=0}^{T-t-1} \lambda^i \mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+i+1}|\mathbf{x}_{t+i})}\left[\delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}_{t+i+1})\right]\right] \tag{33}$$

$$= V^t_{\phi^-}(\mathbf{x}_t) + \mathbb{E}_{\pi_{\text{expl}}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}\left[\sum_{i=0}^{T-t-1} \lambda^i \frac{\pi_{\phi^-}(\mathbf{x}_{t+i}|\mathbf{x}_t)}{\pi_{\text{expl}}(\mathbf{x}_{t+i}|\mathbf{x}_t)} \mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+i+1}|\mathbf{x}_{t+i})}\left[\delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}_{t+i+1})\right]\right] \tag{34}$$

$$= V^t_{\phi^-}(\mathbf{x}_t) + \mathbb{E}_{\pi_{\text{expl}}(\mathbf{x}_{t+1:T}|\mathbf{x}_t)}\left[\sum_{i=0}^{T-t-1} \prod_{j=0}^{i-1} \left(\lambda \frac{\pi_{\phi^-}(\mathbf{x}_{t+j+1}|\mathbf{x}_{t+j})}{\pi_{\text{expl}}(\mathbf{x}_{t+j+1}|\mathbf{x}_{t+j})}\right) \mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+i+1}|\mathbf{x}_{t+i})}\left[\delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}_{t+i+1})\right]\right]. \tag{35}$$

Thus, given off-policy samples $\mathbf{x}_{0:T} \sim \pi_{\text{expl}}(\mathbf{x}_{0:T})$, the importance-sampling–based estimate of the TD(λ) target can be computed as

$$\hat{V}^t_{\text{off-TD}(\lambda)}(\mathbf{x}_{t:T}) = V^t_{\phi^-}(\mathbf{x}_t) + \sum_{i=0}^{T-t-1} \left(\prod_{j=0}^{i-1} \lambda \frac{\pi_{\phi^-}(\mathbf{x}_{t+j+1}|\mathbf{x}_{t+j})}{\pi_{\text{expl}}(\mathbf{x}_{t+j+1}|\mathbf{x}_{t+j})}\right) \delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}'_{t+i+1}), \tag{36}$$

where $\delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}'_{t+i+1})$ is the TD error computed with a freshly sampled $\mathbf{x}'_{t+i+1} \sim \pi_{\phi^-}(\cdot|\mathbf{x}_{t+i})$, and is therefore an unbiased estimator of $\mathbb{E}_{\pi_{\phi^-}(\mathbf{x}_{t+i+1}|\mathbf{x}_{t+i})}[\delta_{t+i}(\mathbf{x}_{t+i}, \mathbf{x}_{t+i+1})]$. We used $\mathbf{x}'_{t+i+1}$ to distinguish it from the off-policy trajectory sample $\mathbf{x}_{t+i+1}$. The targets $\hat{V}^t_{\text{off-TD}(\lambda)}(\mathbf{x}_{t:T})$ for $t = 0, \dots, T-1$ can be efficiently computed in a recursive manner using the tuples $\mathcal{D} = \{(\mathbf{x}_t, \mu^t_{\phi^-}(\mathbf{x}_t))\}_{t=0}^T$ stored during sampling. The full procedure is described in Algorithm 3.

A.5 Details on the Invariance of the Value Function

Proposition 4.1 (Invariance of V^t_π and V^t_*.) *Assume that the energy function is* $\mathrm{O}(m) \times \mathrm{S}_n$*–invariant as follows:*

$$E((R, \sigma) \circ \mathbf{x}) = E(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}, (R, \sigma) \in \mathrm{O}(m) \times \mathrm{S}_n.$$

then both the value function of VGS and the optimal value function preserves $\mathrm{O}(m) \times \mathrm{S}_n$*-invariance:*

$$V^t_\pi((R, \sigma) \circ \mathbf{x}_t) = V^t_\pi(\mathbf{x}_t), \quad V^t_*((R, \sigma) \circ \mathbf{x}_t) = V^t_*(\mathbf{x}_t),$$

for all $\mathbf{x}_t \in \mathcal{X}$, $(R, \sigma) \in \mathrm{O}(m) \times \mathrm{S}_n$.

Proof. In this proof, we omit the subscripts of ∇ and write $\nabla V(\mathbf{x})$ in place of $\nabla_{\mathbf{x}} V(\mathbf{x})$ for brevity.

Invariance of V^t_π. Substituting $\mu_t(\mathbf{x}_t) = -\sigma_t^2 \nabla V^{t+1}_\pi(\alpha_t \mathbf{x}_t)$ to Eq. 7, we obtain the recurrence relation for the value function of VGS

$$V^t_\pi(\mathbf{x}_t) = \frac{\sigma_t^2 \|\nabla V^{t+1}_\pi(\alpha_t \mathbf{x}_t)\|^2}{2} + \mathbb{E}_{\pi(\mathbf{x}_{t+1}|\mathbf{x}_t)}\left[V^{t+1}_\pi(\mathbf{x}_{t+1})\right] \tag{37}$$

$$= \frac{\sigma_t^2 \|\nabla V^{t+1}_\pi(\alpha_t \mathbf{x}_t)\|^2}{2} + \mathbb{E}_{\epsilon_t \sim \mathcal{N}(0,I)}\left[V^{t+1}_\pi(\alpha_t \mathbf{x}_t - \sigma_t^2 \nabla V^{t+1}_\pi(\alpha_t \mathbf{x}_t) + \sigma_t \epsilon_t)\right]. \tag{38}$$

Assume V^{t+1}_π is $\mathrm{O}(m) \times \mathrm{S}_n$–invariant. We show that V^t_π inherits this invariance. First, the action $\circ$ acts orthogonally on $\mathcal{X}$: for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, $(R, \sigma) \in \mathrm{O}(m) \times \mathrm{S}_n$,

$$\langle (R, \sigma) \circ \mathbf{x}, (R, \sigma) \circ \mathbf{y} \rangle = \langle [Rx_{\sigma(1)}, ..., Rx_{\sigma(n)}], [Ry_{\sigma(1)}, ..., Ry_{\sigma(n)}] \rangle = \sum_{i=1}^n x^T_{\sigma(i)} R^T R y_{\sigma(i)} = \sum_{i=1}^n x_i^T y_i = \langle \mathbf{x}, \mathbf{y} \rangle. \tag{39}$$

Hence, the invariance of V_π^{t+1} implies the equivariance of its gradient under $\circ$: $\nabla V_\pi^{t+1}\big((R,\sigma)\circ\mathbf{x}\big) = (R,\sigma)\circ\nabla V_\pi^{t+1}(\mathbf{x})$ (Papamakarios et al., 2021, Lemma 2). Using this property and the recurrence relation in Eq. 38, we have for all $\mathbf{x}_t \in \mathcal{X}$ and $(R,\sigma) \in \mathrm{O}(m)\times S_n$,

$$V_\pi^t((R,\sigma)\circ\mathbf{x}_t) = \frac{\sigma_t^2\|\nabla V_\pi^{t+1}((R,\sigma)\circ\alpha_t\mathbf{x}_t)\|^2}{2} + \mathbb{E}_{\epsilon_t\sim\mathcal{N}(0,I)}\left[V_\pi^{t+1}((R,\sigma)\circ\alpha_t\mathbf{x}_t - \sigma_t^2\nabla V_\pi^{t+1}((R,\sigma)\circ\alpha_t\mathbf{x}_t) + \sigma_t\epsilon_t)\right] \tag{40}$$

$$= \frac{\sigma_t^2\|(R,\sigma)\circ\nabla V_\pi^{t+1}(\alpha_t\mathbf{x}_t)\|^2}{2} + \mathbb{E}_{\epsilon_t\sim\mathcal{N}(0,I)}\left[V_\pi^{t+1}((R,\sigma)\circ(\alpha_t\mathbf{x}_t - \sigma_t^2\nabla V_\pi^{t+1}(\alpha_t\mathbf{x}_t) + \sigma_t(R,\sigma)^{-1}\circ\epsilon_t)\right] \tag{41}$$

$$= \frac{\sigma_t^2\|\nabla V_\pi^{t+1}(\alpha_t\mathbf{x}_t)\|^2}{2} + \mathbb{E}_{(R,\sigma)^{-1}\circ\epsilon_t\sim\mathcal{N}(0,I)}\left[V_\pi^{t+1}((R,\sigma)\circ(\alpha_t\mathbf{x}_t - \sigma_t^2\nabla V_\pi^{t+1}(\alpha_t\mathbf{x}_t) + \sigma_t(R,\sigma)^{-1}\circ\epsilon_t))\right] \tag{42}$$

$$= \frac{\sigma_t^2\|\nabla V_\pi^{t+1}(\alpha_t\mathbf{x}_t)\|^2}{2} + \mathbb{E}_{\epsilon'_t\sim\mathcal{N}(0,I)}\left[V_\pi^{t+1}(\alpha_t\mathbf{x}_t - \sigma_t^2\nabla V_\pi^{t+1}(\alpha_t\mathbf{x}_t) + \sigma_t\epsilon'_t)\right] \tag{43}$$

$$= V_\pi^t(\mathbf{x}_t). \tag{44}$$

The second equality uses gradient equivariance. The third uses that $\mathcal{N}(0,I)$ is invariant under rotations, reflections, and permutations, so $(R,\sigma)^{-1}\circ\epsilon_t \sim \mathcal{N}(0,I)$. The fourth equality uses the assumed invariance of V_π^{t+1}.

Finally, we verify the terminal case $t = T$. By construction of the reference process ($\mu = 0$ in Eq. 3), $\tilde{\pi}(\mathbf{x}_T)$ is an isotropic zero-mean Gaussian. Consequently, $V_\pi^T(\mathbf{x}_T) = \tilde{E}(\mathbf{x}_T) = E(\mathbf{x}_T) + \log\tilde{\pi}(\mathbf{x}_T)$ is $\mathrm{O}(m)\times\mathrm{S}_n$–invariant whenever $E(\mathbf{x}_T)$ is invariant. The result then follows by backward induction on t.

Invariance of V_*^t. By Eq. 6, it suffices to show that $\tilde{q}(\mathbf{x}_t)$ inherits the $\mathrm{O}(m)\times\mathrm{S}_n$-invariance of $E(\mathbf{x}_T)$, since $\tilde{\pi}(\mathbf{x}_t)$ is an isotropic zero-mean Gaussian and therefore invariant. If $E(\mathbf{x}_T)$ is $\mathrm{O}(m)\times\mathrm{S}_n$-invariant, then by Eq. 1 the density $q(\mathbf{x}_T)$ is also $\mathrm{O}(m)\times\mathrm{S}_n$-invariant. Moreover, $\tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t)$ is invariant in the sense that for all $\mathbf{x}_t, \mathbf{x}_{t+1} \in \mathcal{X}$ and $(R,\sigma) \in \mathrm{O}(m)\times\mathrm{S}_n$, we have

$$\tilde{\pi}((R,\sigma)\circ\mathbf{x}_{t+1}|(R,\sigma)\circ\mathbf{x}_t) = \frac{1}{(2\pi\sigma_t^2)^{D/2}}\exp -\frac{\|(R,\sigma)\circ\mathbf{x}_{t+1} - (R,\sigma)\circ\alpha_t\mathbf{x}_t\|^2}{2\sigma_t^2} \tag{45}$$

$$= \frac{1}{(2\pi\sigma_t^2)^{D/2}}\exp -\frac{\|(R,\sigma)\circ(\mathbf{x}_{t+1} - \alpha_t\mathbf{x}_t)\|^2}{2\sigma_t^2} \tag{46}$$

$$= \frac{1}{(2\pi\sigma_t^2)^{D/2}}\exp -\frac{\|\mathbf{x}_{t+1} - \alpha_t\mathbf{x}_t\|^2}{2\sigma_t^2} \tag{47}$$

$$= \tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t). \tag{48}$$

The third equality follows from the orthogonality of the action $\circ$, as established in Eq. 39.

Using these properties, for all $\mathbf{x}_t \in \mathcal{X}$ and $(R, \sigma) \in \mathrm{O}(m) \times \mathrm{S}_n$, we obtain

$$\tilde{q}((R,\sigma)\circ\mathbf{x}_t) = \int_{\mathcal{X}^{T-t}} \tilde{q}((R,\sigma)\circ\mathbf{x}_{t:T}) d((R,\sigma)\circ\mathbf{x})_{t+1:T} \tag{49}$$

$$= \int_{((R,\sigma)^{-1}\circ\mathcal{X})^{T-t}} \tilde{q}((R,\sigma)\circ\mathbf{x}_{t:T}) \left|\det \frac{\partial (R,\sigma)\circ\mathbf{x}}{\partial \mathbf{x}}\right|_{t+1:T} d\mathbf{x}_{t+1:T} \tag{50}$$

$$= \int_{\mathcal{X}^{T-t}} \tilde{q}((R,\sigma)\circ\mathbf{x}_{t:T}) d\mathbf{x}_{t+1:T} \tag{51}$$

$$= \int_{\mathcal{X}^{T-t}} \tilde{\pi}((R,\sigma)\circ\mathbf{x}_{t:T}) \frac{q((R,\sigma)\circ\mathbf{x}_T)}{\tilde{\pi}((R,\sigma)\circ\mathbf{x}_T)} d\mathbf{x}_{t+1:T} \tag{52}$$

$$= \int_{\mathcal{X}^{T-t}} \tilde{\pi}((R,\sigma)\circ\mathbf{x}_t) \prod_{i=0}^{T-t-1} \tilde{\pi}((R,\sigma)\circ\mathbf{x}_{t+i+1}|(R,\sigma)\circ\mathbf{x}_{t+i}) \frac{q((R,\sigma)\circ\mathbf{x}_T)}{\tilde{\pi}((R,\sigma)\circ\mathbf{x}_T)} d\mathbf{x}_{t+1:T} \tag{53}$$

$$= \int_{\mathcal{X}^{T-t}} \tilde{\pi}(\mathbf{x}_t) \prod_{i=0}^{T-t-1} \tilde{\pi}(\mathbf{x}_{t+i+1}|\mathbf{x}_{t+i}) \frac{q(\mathbf{x}_T)}{\tilde{\pi}(\mathbf{x}_T)} d\mathbf{x}_{t+1:T} \tag{54}$$

$$= \int_{\mathcal{X}^{T-t}} \tilde{q}(\mathbf{x}_{t:T}) d\mathbf{x}_{t+1:T} \tag{55}$$

$$= \tilde{q}(\mathbf{x}_t). \tag{56}$$

Eq. 50 follows from a change of variables in Eq. 49. In Eq. 51 we use the fact that $(R,\sigma)^{-1}\circ\mathcal{X} = \mathcal{X}$ and that $\mathbf{x} \mapsto (R,\sigma)\circ\mathbf{x}$ is an orthogonal transformation, so that $\left|\det \frac{\partial (R,\sigma)\circ\mathbf{x}}{\partial \mathbf{x}}\right| = 1$. Eq. 52 and Eq. 55 follow from the definition of the joint target distribution $\tilde{q}(\mathbf{x}_{t:T}) = \tilde{\pi}(\mathbf{x}_{t:T})\, q(\mathbf{x}_T)/\tilde{\pi}(\mathbf{x}_T)$. Finally, Eq. 54 uses the already established invariance of $q(\mathbf{x}_T)$, $\tilde{\pi}(\mathbf{x}_t)$ and $\tilde{\pi}(\mathbf{x}_{t+1}|\mathbf{x}_t)$.

Therefore, $\tilde{q}(\mathbf{x}_t)$ is $\mathrm{O}(m) \times \mathrm{S}_n$-invariant, which yields the desired invariance of V_*^t via Eq. 6.

□

Proposition A.1 ($\mathrm{O}(m)$-Invariance of V_π^t and V_*^t)**.** *Assume that the energy function is $\mathrm{O}(m)$–invariant as follows:*

$$E(R\cdot\mathbf{x}) = E(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}, R \in \mathrm{O}(m)$$

then both the value function of VGS and the optimal value function preserves $\mathrm{O}(m)$-invariance:

$$V_\pi^t(R\cdot\mathbf{x}_t) = V_\pi^t(\mathbf{x}_t), \quad V_*^t(R\cdot\mathbf{x}_t) = V_*^t(\mathbf{x}_t),$$

for all $\mathbf{x}_t \in \mathcal{X}$, $R \in \mathrm{O}(m)$.

Proof. It is straightforward to check that $\cdot : (R, \mathbf{x}) \mapsto R\cdot\mathbf{x}$ is an orthogonal action and that the isotropic zero-mean Gaussian is invariant under it. Therefore, the proof in Proposition 4.1 applies after replacing the product action $\circ : ((R,\sigma),\mathbf{x}) \mapsto ((R,\sigma)\circ\mathbf{x})$ with $\cdot : (R,\mathbf{x}) \mapsto R\cdot\mathbf{x}$. We omit the technical details. □

B Algorithms

The detailed algorithm for training the value function in VGS is presented in Algorithm 2. For clarity, we first present a minimal TD(0) version. In the more general setting where off-policy TD(λ) targets are computed using an exploration policy, the TD target computation step in Algorithm 2 is substituted with the procedure in Algorithm 3. To satisfy the boundary condition at $t = T$, $V_{\phi^-}^T(\mathbf{x}_T)$ is replaced by $\tilde{E}(\mathbf{x}_T)$ in the value target calculation.

C Experimental Setup

C.1 Target Distributions

GMM (Sendera et al., 2024). We use a 2-dimensional Gaussian Mixture Model, with its density given by $\gamma(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m \mathcal{N}(\mathbf{x};\mu_i,\Sigma_i)$., where $m = 25, (\mu_i)_{i=1}^{25} = \{-10,-5,0,5,10\} \times \{-10,-5,0,5,10\} \subset \mathbb{R}^2$ and

Algorithm 2 Learning Value Functions in VGS (TD(0))

Input: Energy $E(\mathbf{x})$, value $V_\phi^t(\mathbf{x}_t)$, schedules $\{\alpha_t\}_{t=0}^{T-1}$, $\{\sigma_t\}_{t=0}^{T-1}$.
for $i = 1$ **to** n_{iter} **do**
  Sample trajectory $\mathbf{x}_{0:T} \sim \pi_{\phi^-}(\mathbf{x}_{0:T})$ // Alg. 1
  Store tuples $\mathcal{D} = \{(\mathbf{x}_t, \mu_{\phi^-}^t(\mathbf{x}_t))\}_{t=0}^T$
  for $(\mathbf{x}_t, \mu_{\phi^-}^t(\mathbf{x}_t))$ **in** $\mathcal{D}$ **do**
    Compute the value target $\hat{V}^t(\mathbf{x}_t)$ as $\hat{V}_{\text{TD}}^t(\mathbf{x}_{t:t+1})$ for $t < T$, and as $\tilde{E}(\mathbf{x}_T)$ for $t = T$
  end for
  for $j = 1$ **to** n_{update} **do**
    for $(\mathbf{x}_t, \_)$ **in** $\mathcal{D}$ **do**
      Optimize $\min_\phi((V_\phi^t(\mathbf{x}_t) - \hat{V}^t(\mathbf{x}_t))^2$
    end for
  end for
  Update target value parameters $\phi^- \leftarrow \kappa\phi^- + (1-\kappa)\phi$
end for

Algorithm 3 Computing off-policy TD(λ) targets with an exploration policy

Input: Target value network $V_{\phi^-}^t(\mathbf{x}_t)$, schedules $\{\alpha_t\}_{t=0}^{T-1}$, $\{\sigma_t\}_{t=0}^{T-1}$.
Sampled trajectory $\mathbf{x}_{0:T} \sim \pi_{\text{expl}}(\mathbf{x}_{0:T})$ using $\mu_{\text{expl}}^t(\mathbf{x}_t) = \mu_{\phi^-}^t(\mathbf{x}_t)$, $(\sigma_t)_{\text{expl}} = \eta\sigma_t$ // Alg. 1
Stored tuples $\mathcal{D} = \{(\mathbf{x}_t, \mu_{\phi^-}^t(\mathbf{x}_t))\}_{t=0}^T$

Initialize advantage estimate: $\hat{A}_T = 0$
for $t = T-1, \ldots, 0$ **do**
  Resample $\mathbf{x}'_{t+1} \sim \pi_{\phi^-}(\cdot|\mathbf{x}_t)$ as $\mathbf{x}'_{t+1} = \alpha_t\mathbf{x}_t + \mu_{\phi^-}^t(\mathbf{x}_t) + \sigma_t\epsilon_t$ // Eq. (3)
  Compute TD error $\delta_t(\mathbf{x}_t, \mathbf{x}'_{t+1})$ // Eq.(10)
  Update advantage estimate $\hat{A}_t = \lambda \frac{\pi_{\phi^-}(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\pi_{\text{expl}}(\mathbf{x}_{t+1}|\mathbf{x}_t)} \hat{A}_{t+1} + \delta_t(\mathbf{x}_t, \mathbf{x}'_{t+1})$
  Compute TD target $\hat{V}_{\text{off-TD}(\lambda)}^t(\mathbf{x}_{t:T}) = V_{\phi^-}^t(\mathbf{x}_t) + \hat{A}_t$ // Eq.(36)
end for

$(\Sigma_i)_{i=1}^{25} = 0.3\mathbf{I} \subset \mathbb{R}^{2\times 2}$. With these parameters, a 25-component Gaussian Mixture density with evenly separated modes is obtained. A well-trained sampler must be able to sample from all twenty five modes.

Funnel (Neal, 2003). We use a 10-dimensional funnel distribution with its density given by $\gamma(\mathbf{x}) = \mathcal{N}(x_1; 0, \sigma^2)\prod_{i=1}^d \mathcal{N}(x_i; 0, e^{x_1})$, where $\mathbf{x} \in \mathbb{R}^d$. The specific parameters are $d = 10$ and $\sigma = 3$. The funnel distribution is known to be a challenging distribution for testing MCMC methods, as the variance grows exponentially as the first dimension x_1 increases.

ManyWell (Noé et al., 2019). We use a 32-dimensional manywell distribution with its density given by $\gamma(\mathbf{x}) = \prod_{i=1}^{16} \mu(x_{2i-1}, x_{2i})$, where $\mu(x_i, x_j) = \exp(-x_i^4 + 6x_i^2 + 0.5x_i - 0.5x_j^2)$ and $x \in \mathbb{R}^{32}$.

DW-4 (Köhler et al., 2020). This system consists of 4 particles in two dimensions, where each pair of particles interacts via a *double-well* potential: $\gamma^{\text{DW}}(\mathbf{x}) = \frac{1}{2\tau}\sum_{i,j}\left[a\,(d_{ij} - d_0) + b\,(d_{ij} - d_0)^2 + c\,(d_{ij} - d_0)^4\right]$, where d_{ij} denotes the Euclidean distance between particles i and j. These pairwise interactions induce multiple metastable states in the system. We use the same parameters as in prior works (Midgley et al., 2023a; Akhound-Sadegh et al., 2024; He et al., 2024), setting $d_0 = 4$, $a = 0$, $b = -4$, $c = 0.9$, and $\tau = 1$. We split the MCMC samples from Klein et al. (2023) into validation and test sets, which serve as our reference ground truth samples. We use 1,000 samples for validation dataset, and 10,000 for test dataset.

LJ-13/LJ-55 (Köhler et al., 2020). We consider systems of 13 and 55 particles in three dimensions, where particles interact via the *Lennard-Jones* (LJ) potential: $\gamma^{\text{LJ}}(\mathbf{x}) = \frac{\epsilon}{2\tau}\sum_{i,j}\left[\left(\frac{r_m}{d_{ij}}\right)^{12} - 2\left(\frac{r_m}{d_{ij}}\right)^6\right]$, with d_{ij} denoting the Euclidean distance between particles i and j. In this setting, the energy and its gradient become unbounded as $d_{ij} \to 0$, making the optimization landscape challenging. To prevent particle dissociation, we follow prior works (Midgley et al., 2023a; Akhound-Sadegh et al., 2024; He et al., 2024) and add a harmonic potential

Table 5: Hyperparameters for VGS used in n-body system experiments

Target	T	n_{iter}	n_{update}	λ	σ_0^2	σ_{T-1}^2	var schedule	α_t	σ_{init}	η	κ	lr	Clip $\tilde{E}(\mathbf{x})$	Clip $\hat{A}_t$	Hidden dim
DW-4	50	50000	3	0	0.2	0.001	quad	1 (VE)	0	1.2	0.9	1e-5	-	-	256
LJ-13	100	50000	3	0.9	0.05	0.0001	exp	1 (VE)	0	1.2	0.98	1e-5	1e4	100	512
LJ-55	100	50000	3	0.9	0.03	0.0001	quad	1 (VE)	0	1.2	0.95	1e-5	1e6	100	1024

centered at the system's center of mass (CoM): $\gamma^{\text{osc}}(\mathbf{x}) = \frac{1}{2}\sum_i \|x_i - x_{\text{CoM}}\|^2$. The same works also set $\epsilon = 1$, $r_m = 1$, and $\tau = 1$, which we adopt in our experiments.We split the MCMC samples from Klein et al. (2023) into validation and test sets, which serve as our reference ground truth samples. We use 1,000 samples for validation dataset, and 10,000 for test dataset.

C.2 Performance Metrics

Total Variation Distance (TVD-D, E). The total variation distance (TVD) between two distributions P and Q is defined as $\text{TVD}(P,Q) = \frac{1}{2}\int_{\mathcal{X}} |P(x) - Q(x)|\, dx$. However, accurate computation of TVD directly in high-dimensional data space is intractable in practice. Instead, we project the data to physically meaningful one-dimensional statistics—interatomic distances and total energies—and compute TVD on the resulting distributions. We refer to these metrics as TVD-D (for distances) and TVD-E (for energies), respectively. Each is approximated using histogram-based probability distributions, with the number of bins determining the discretization scale. We use 200 bins for all experiments.

Log Partition Function Error ($\Delta \log Z_r, \Delta \log Z_f$). The log partition function error is defined as the difference between the target density's true log partition function and its estimated value, denoted as, $\Delta \log Z = |\log Z - \log \hat{Z}|$ (Blessing et al., 2024). The estimate $\hat{Z}$ is computed by using samples from either the trained distribution ($\pi_\phi(\mathbf{x})$) or target ($q(\mathbf{x}) = \gamma(\mathbf{x})/Z$) distribution. Depending on the distribution used, the estimates are named as reversed estimate ($\hat{Z}_r$) or forward estimate ($\hat{Z}_f$) respectively. Each estimates and its corresponding log partition function error is defined as

$$\hat{Z}_r = \mathbb{E}_{\substack{\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_{\text{init}}^2 I) \\ \mathbf{x}_{t+1} \sim \pi_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t)}} \left[\frac{\gamma(\mathbf{x}_T) \prod_{t=0}^{T-1} \tilde{q}(\mathbf{x}_t|\mathbf{x}_{t+1})}{p(\mathbf{x}_0) \prod_{t=0}^{T-1} \pi_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t)} \right] \quad \Delta \log Z_r = \left| \log Z - \log \hat{Z}_r \right|$$

$$\hat{Z}_f = 1/\mathbb{E}_{\substack{\mathbf{x}_T \sim q(\mathbf{x}_T) \\ \mathbf{x}_{t-1} \sim \tilde{q}(\mathbf{x}_{t-1}|\mathbf{x}_t)}} \left[\frac{p(\mathbf{x}_0) \prod_{t=0}^{T-1} \pi_\phi(\mathbf{x}_{t+1}|\mathbf{x}_t)}{\gamma(\mathbf{x}_T) \prod_{t=0}^{T-1} \tilde{q}(\mathbf{x}_t|\mathbf{x}_{t+1})} \right] \quad \Delta \log Z_f = \left| \log Z - \log \hat{Z}_f \right|.$$

Wasserstein-2 Distance ($\mathcal{W}^2$). The 2-Wasserstein distance between two distributions is defined as $\mathcal{W}_2(P,Q) = \left(\inf_\pi \int \pi(x,y) d(x,y)^2\, dx\, dy\right)^{\frac{1}{2}}$, where π is the transport plan with marginals constrained to P and Q respectively (Akhound-Sadegh et al., 2024). We use the Euclidean distance $||x-y||_2$ for $d(x,y)$ and calculate the 2-Wasserstein distance between generated samples from the sampler and the ground truth data. For the actual computation, we use the Python Optimal Transport Library (Flamary et al., 2021).

C.3 Experimental Details

C.3.1 Sampling from n-Body System Experiments

For the n-body system experiments, we conducted 3 independent runs with different random seeds for each setting. All reported metrics are the mean and standard deviation over these 3 runs. Each experiment was conducted on a single RTX 3090 GPU or RTX 4090 GPU with 24 GB of memory. Below, we provide detailed implementation and hyperparameters for each algorithm used.

We applied early stopping during training based on 1,000 validation samples and saved the checkpoint with the lowest TVD-D score, following the protocol in He et al. (2024). For evaluation, we report metrics computed on 10,000 samples generated from the final checkpoint and compare them against the test set. Since direct sampling from the target potentials is infeasible, we use the validation and test sets provided in the official iDEM

repository at `https://github.com/jarridrb/DEM` (Akhound-Sadegh et al., 2024), which is adopted from Klein et al. (2023). As the original validation set contains 10,000 samples, we randomly select and fix a subset of 1,000 samples for early stopping. We provide implementation details and hyperparameter settings for each algorithm below.

Value Gradient Sampler. The hyperparameters used for VGS are summarized in Table 5. As described in Algorithm 2, n_{iter} denotes the total number of training iterations, and n_{update} controls how many times the rollout samples $\mathcal{D}$ is reused for gradient updates. We use a batch size of 512 during the trajectory sampling phase and 2048 for TD updates. We use the Adam optimizer (Kingma and Ba, 2017) for gradient update.

We use the IMLP value-function architecture in our main experiments as it is more computationally efficient than IGNN. IMLP takes the sorted pairwise distances $\{d_{ij}\}_{i<j\leq n}$ as input; for the LJ experiments, we instead use their inverses, motivated by the analytic form of the LJ potential. To prevent numerical divergence at $d = 0$, a small constant is added. The network is a four-layer MLP with ReLU activations. We encode the time step with a 128-dimensional sinusoidal embedding and project it to the hidden dimension via a fully connected layer. We adjust model complexity via the hidden dimension, which we report across experiments in the table.

The variance schedule $\{\sigma_t^2\}_{t=0}^{T-1}$ is defined by choosing the endpoints σ_0^2 and σ_{T-1}^2, then interpolating intermediate values according to a predefined rule (see "var schedule" in the table). In all experiments we use a variance-exploding(VE) reference process $\tilde{\pi}$ initialized from a Dirac delta, i.e., $\alpha_t = 1$ and $\sigma_{\text{init}} = 0$.

The hyperparameter η determines the noise scale of the exploration policy $(\sigma_t)_{\text{expl}} = \eta\sigma_t$, and thus the exploration level during sampling. For simplicity, we do not decay η during training. The parameter κ is the momentum coefficient for the exponential moving average used to update the target value network.

We find that clipping the terminal rewards $\tilde{E}(\mathbf{x})$ and the advantage estimates $\hat{A}_t$ (defined in Algorithm 3) improves training stability. The clipping thresholds are reported in the table as "Clip $\tilde{E}(\mathbf{x})$" and "Clip $\hat{A}_t$".

The LJ potential is particularly sensitive to sample noise, so during evaluation we omit the final diffusion step by setting $\sigma_{T-1} = 0$.

Baseline Methods. We conduct baseline experiments using the following repositories: FAB with an SE(3)-augmented coupling flow architecture (Midgley et al., 2023a,b) from `https://github.com/lollcat/se3-augmented-coupling-flows`, iDEM (Akhound-Sadegh et al., 2024) from `https://github.com/jarridrb/DEM`, DiKL (He et al., 2024) from `https://github.com/jiajunhe98/DiKL`, PIS and DDS (Berner et al., 2022; Richter and Berner, 2024) from `https://github.com/juliusberner/sde_sampler`, and GFlowNets (GFN-DB, GFN-TB, GFN-SubTB) from `https://github.com/GFNOrg/gfn-diffusion`.

For all baselines, we apply early stopping based on validation TVD-D, following the same protocol used for VGS. A fixed validation set of 1,000 samples and a test set of 10,000 samples are used across all experiments (Appendix C.1). We use the default hyperparameters provided in each repository unless otherwise specified, with three exceptions.

First, FAB and DiKL do not provide default configurations for LJ-55, so we adopt their LJ-13 hyperparameters. Because their equivariant networks inherently scale computation with the number of particles, we do not explicitly increase the hidden dimensions or number of layers.

Second, to address out-of-memory issues in certain experiments, we selected the largest batch size our hardware allows. For instance, the batch size for FAB is reduced from 1024 to 128 on LJ-13, and to 8 on LJ-55. For DiKL on LJ-55, the batch size is decreased from 256 to 64.

Third, since PIS, DDS, and GFlowNets did not originally include particle system experiments, we replaced their MLP-based policy networks with EGNNs, following the default configurations in iDEM (Akhound-Sadegh et al., 2024). We used an EGNN with a hidden dimension of 128, utilizing 3 hidden layers for DW-4 and 5 hidden layers for LJ-13 and LJ-55.

C.3.2 Sampling from Synthetic Distribution Experiments

For the synthetic distribution experiments, we conducted 3 independent runs with different seeds for each experiment. All reported metrics are the averages of the 3 runs, along with their standard deviation. Every experiment is performed on a single NVIDIA RTX 3090 (24GB) GPU. To compute the metrics, we use 2,000 randomly sam-

Table 6: Hyperparameters for VGS used in synthetic distribution experiments

Target	Methods	T	n_{iter}	n_{update}	λ	σ_0^2	σ_{T-1}^2	var schedule	α_t	σ_{init}	η	κ	lr	Clip $E(\mathbf{x})$	Clip $\hat{A}_t$	Hidden dim
GMM	**VGS**	50	5000	3	0	0.1	0.1	const	1 (VE)	0	1.2	0.98	1e-4	-	-	256
	+TD(λ)	50	5000	3	0.9	0.1	0.1	const	1 (VE)	0	1.2	0.98	1e-4	-	-	256
	+Buffer	50	5000	1	0	0.1	0.1	const	1 (VE)	0	1.1	0.98	1e-4	-	-	256
Funnel	**VGS**	100	20000	3	0	0.03	0.0001	quad	1 (VE)	0	1.0	0.98	1e-5	1e3	100	256
	+TD(λ)	100	20000	3	0.9	0.03	0.0001	quad	1 (VE)	0	1.0	0.98	1e-5	1e3	100	256
	+Buffer	100	20000	1	0	0.03	0.0001	quad	1 (VE)	0	1.0	0.98	1e-5	1e3	100	256
ManyWell	**VGS**	100	20000	3	0	0.03	0.0001	quad	1 (VE)	0	1.2	0.98	1e-5	1e3	100	512
	+TD(λ)	100	20000	3	0.9	0.03	0.0001	quad	1 (VE)	0	1.2	0.98	1e-5	1e3	100	512
	+Buffer	100	20000	1	0	0.03	0.0001	quad	1 (VE)	0	1.1	0.98	1e-5	1e3	100	512

Table 7: Hyperparameters for SMC-based Methods used in synthetic distribution experiments

Methods / Parameters	**GMM**	**Funnel**	**ManyWell**
SMC			
Initial σ	10.0	1.0	12.0
HMC stepsize ($\beta \leq 0.5$)	0.2	0.01	0.01
HMC stepsize ($\beta > 0.5$)	0.2	0.1	0.1
AFT			
Initial σ	9.0	1.0	10.0
Learning Rate	1e-3	1e-3	1e-3
CRAFT			
Initial σ	9.0	1.0	10.0
Learning Rate	1e-2	1e-3	1e-3

pled data from a target density and another 2,000 randomly sampled data from a learned sampler to evaluate the metrics. Below, we provide detailed implementation and hyperparameters for each algorithm used.

Value Gradient Sampler. The hyperparameters used for VGS are summarized in Table 6. Consistent with the n-body system experiments, we use a batch size of 512 during trajectory sampling and 2048 for training. In our experiments with buffer, we adopt the buffer design from Sendera et al. (2024), which maintains a prioritized mixture of MCMC and policy-generated samples based on sample energy. Following their approach, we alternate training VGS on forward and backward trajectories.

SMC-based Methods. The implementation of SMC-based method is based on `https://github.com/google-deepmind/annealed_flow_transport`. The implementation of SMC variants, AFT (Arbel et al., 2021) and CRAFT (Matthews et al., 2023) are provided together. For GMM, an initial gaussian distribution with $\sigma = 10.0$ is used to ensure suitable exploration in the range $[-10, 10] \times [-10, 10] \in \mathbb{R}^2$ where the modes exists. The default hyper-parameter for the funnel distribution and the manywell distribution is provided in the code. For all the experiments, the resample threshold of 0.3 is used. Additional hyperparameters are reported in Table 7.

SDE-based Methods. For SDE-based methods, we follow the implementation of the official repository of the DIS Berner et al. (2022); Richter and Berner (2024). The implementation of PIS (Zhang and Chen, 2022), DIS (Berner et al., 2022) and DDS (Vargas et al., 2023; Richter and Berner, 2024) are all available. The PIS, DIS and DDS is trained using the default setting provided by the authors using the log-variance training objective (Richter and Berner, 2024). Specifically, we used timesteps of $T = 200$ for PIS, DIS and $T = 256$ for DDS. We trained all models for 30,000 iterations with the learning rate value of 5e-3. The Fourier MLP architecture (Zhang and Chen, 2022) was used, which is fundamentally a MLP architecture with sinusoidal time step embeddings. The network is composed of 4 layers with the hidden dimension of size 64 and the time step embedding dimension of size 64.

GFN Methods. For the GFN baselines, we use the official implementation available at `https://github.`

com/GFNOrg/gfn-diffusion , corresponding to Sendera et al. (2024). This repository provides implementations of prior GFN studies on training diffusion samplers, which we refer to as GFN-TB (Lahlou et al., 2023), GFN-SubTB (Zhang et al., 2023), and GFN-TB-Imp (Sendera et al., 2024) in Table 4. For all baselines, we adopt the default architectural configurations and training techniques provided by the authors, including linearly decaying exploration noise during training.

Specifically, for GFN-TB (Lahlou et al., 2023), we employ the trajectory balance (TB) loss as described in the original work. For GFN-SubTB (Zhang et al., 2023), we use the sub-trajectory balance (SubTB) loss with $\lambda = 2$, and incorporate an energy-informed flow together with a score-informed policy network, following the architectural design in Zhang et al. (2023). For GFN-TB-Imp (Sendera et al., 2024), we apply the log-variance loss, a variant of the TB loss, and similarly use a score-informed policy network. In addition, we employ a prioritized replay buffer combined with local search, an improved off-policy strategy introduced in Sendera et al. (2024). Lastly, although GFN-DB (Bengio et al., 2023) was not originally evaluated with diffusion samplers, we include it for completeness. We use its basic architecture and a training procedure consistent with the other baselines, tuning only the learning rates.

C.3.3 Ablation Experiments

VGS Architecture Ablation (Fig. 2). Experiments were conducted on the LJ-13 potential. The histogram was plotted using 10,000 samples. Details of the test samples are provided in Appendix C.3.1.

For invariant MLP(IMLP), we used the same hyperparameters as in the main results (Table 1). For non-invariant MLP, we kept the setup identical to IMLP but replaced the input from the sorted pairwise distances $\{d_{ij}\}_{i<j\leq n}$ to the raw 39-dimensional coordinate vector $\mathbf{x}$. To isolate architectural effects, non-invariant MLP was also trained and evaluated in the zero-mean space $\mathcal{X}$.

For IGNN, we used three message-passing layers with hidden dimension 64; the final node embeddings were sum-pooled and passed to a two-layer MLP with hidden dimension 64. Due to computational constraints, we reduced the batch size to 64 for both sampling and training. Due to altered training dynamics under this smaller batch and different architecture, we set the learning rate to 1e$-$4 and the momentum coefficient κ to 0.9.

TD(0) vs. TD(λ) Comparison (Fig. 3). We provide the experimental details for Fig. 3 which demonstates the benefit of TD(λ) over TD(0). The experiment was conducted on the funnel distribution and the experimental details are given as follows.

The left panel shows sample efficiency experiment demonstrating that TD(λ) converges faster than TD(0). This behavior is mirrored by their GFN counterparts, GFN-SubTB and GFN-DB, respectively. For all methods, we searched over the learning rate grid $\{1\text{e-}2, 1\text{e-}3, 1\text{e-}4, 1\text{e-}5\}$. For each method, the largest value that did not cause divergence was selected. All hyperparameters were kept identical to those used in Table 6.

The right panel of Fig. 3 shows how sampling steps effects the sampler performance, demonstrating that TD(λ) achieves better performance than TD(0) for large timesteps T. This trend is likewise mirrored in their GFN counterparts. For VGS, we used a quadratic variance schedule and adjusted the endpoints σ_0^2 and σ_{T-1}^2, to ensure that the total injected noise remains constant regardless of T. For GFN, we adopted a constant variance schedule consistent with the main experimental settings (Table 4), while similarly ensuring that the total injected noise was preserved. All other hyperparameters were kept identical to the main experiment.