

## A Experiment details

Code is available at <https://anonymous.4open.science/r/SGDS-D38C>.

And the reference samples can be downloaded from <https://zenodo.org/records/15436773>.

### A.1 MCMC samplers for Searcher

**Annealed importance sampling (AIS).** Annealed importance sampling (AIS) [30] is an MCMC sampling method for estimating the partition functions of target distributions. AIS bridges between an easy-to-sample initial distribution  $\pi_0(x)$  and a target distribution  $\pi_T(x)$  through a sequence of intermediate distributions  $\{\pi_t(x)\}_{t=0}^T$ , where  $T$  is the length of a trajectory or chain. Each intermediate distribution  $\pi_t(x)$  typically has the form:

$$\pi_t(x) \propto \pi_0(x)^{1-\beta_t} \pi_T(x)^{\beta_t}, \quad 0 = \beta_0 < \beta_1 < \dots < \beta_T = 1, \quad (10)$$

where  $\{\beta_t\}$  is a predefined annealing schedule, and we use  $\beta_t = \frac{t}{T}$  in our framework. AIS generates samples through an MCMC transition kernel at each intermediate distribution with the following SDE simulation:

$$dx_t = \nabla \log \pi_t(x_t) dt + \sqrt{2} dW_t, \quad (11)$$

where  $\nabla \log \pi_t(x_t) = (1 - \beta_t) \nabla \log \pi_0(x_t) + \beta_t \nabla \log \pi_T(x_t)$  is the score function of the annealed distribution (unnormalized). Then it accumulates importance weights given by:

$$w_{\text{AIS}} = \prod_{t=1}^T \frac{\pi_t(x_{t-1})}{\pi_{t-1}(x_{t-1})}, \quad (12)$$

and the expectation of these weights provides an unbiased estimator of the partition function ratio between  $\pi_T(x)$  and  $\pi_0(x)$ :

$$\frac{Z_T}{Z_0} \approx \frac{1}{N} \sum_{i=1}^N w^{(i)}, \quad (13)$$

where  $w^{(i)}$  is the importance weight computed for the  $i$ -th AIS trajectory, and  $N$  is the total number of trajectories. We compute the unbiased estimation of the log scale of the partition function for Manywell experiments by

$$\log \hat{Z}_T = \log \frac{1}{N} \sum_{i=1}^N w^{(i)}, \quad (14)$$

where  $\log Z_0 = 0$  because the initial distribution is Gaussian in our framework.

**Metropolis-Adjusted Langevin Algorithm (MALA).** The Metropolis-Adjusted Langevin Algorithm (MALA) [36] is an MCMC method that uses the gradient of the energy function to generate samples from a target distribution  $\pi(x)$ . MALA starts by sampling initial states  $x_0 \sim \pi_0(x_0)$ , where  $\pi_0(\cdot)$  is some proposed initial distribution (in most cases,  $\mathcal{N}(0, \sigma^2 I)$ ). It then iteratively proceeds transition from  $x_t$  to  $x_{t+1}$  by simulating the following Langevin dynamics:

$$dx_t = -\nabla \mathcal{E}(x_t) dt + \sqrt{2} dW_t, \quad (15)$$

Here,  $x_t$  is the current state at time  $t$ ,  $W_t$  denotes the standard Brownian motion, and  $\mathcal{E}(x)$  is the energy function of target distribution  $\pi(x)$ , i.e.  $-\nabla \mathcal{E}(x_t) = \nabla \log \pi(x_t)$ .

The proposed sample  $x_{t+1}$  is then accepted or rejected according to the Metropolis-Hastings acceptance probability:

$$\alpha = \min \left\{ 1, \frac{\pi(x_{t+1}) q(x_t | x_{t+1})}{\pi(x_t) q(x_{t+1} | x_t)} \right\}, \quad (16)$$

where  $q(\cdot | \cdot)$  denotes the Gaussian transition density induced by the Langevin proposal:

$$x_{t+1} = x_t - \nabla \mathcal{E}(x_t) \Delta t + \sqrt{2\Delta t} \cdot z, \quad z \sim \mathcal{N}(0, I). \quad (17)$$

The step size  $\Delta t$  is a key factor influencing the quality of sampling. For all tasks, we utilize the scheduling of step size, by comparison between the current acceptance rate and the target acceptance rate (57.4%). We use MALA as Searcher on 40GMM, LJ-13, and LJ-55.

Also, since a MALA trajectory forms a Markov chain, consecutive samples are still correlated and therefore  $\{x_i\}_{i=1}^N$  are not strictly i.i.d. To reduce the most severe correlations we discard the first  $M_{\text{burn-in}}$  iterations as burn-in and use all subsequent states directly. We then compute a rough estimate

$$\log \hat{Z} = \log \left[ \frac{1}{N} \sum_{i=1}^N \exp(-\mathcal{E}(x_i)) \right], \quad (18)$$

where this estimator is biased since  $x_i \sim \pi$  ideally and  $\mathbb{E}_\pi[\exp(-\mathcal{E}(x))] = Z \int \pi^2(x) dx < Z$ . Despite the bias, the estimation can provide a numerically reasonable heuristic value for the initialization of the Learner’s  $\log Z_\theta$ .

**Underdamped Langevin dynamics.** For MCMC Searchers of a real-world molecule, Alanine Dipeptide, we adopt underdamped Langevin dynamics as our molecular dynamics (MD). This framework combines deterministic forces with stochastic fluctuations, which is essential for accurately capturing thermal motion and inertial effects of the molecules. The resulting dynamics are governed by the following system of stochastic differential equations:

$$\begin{aligned} dx_t &= v_t dt, \\ dv_t &= -M^{-1} \nabla \mathcal{E}(x_t) dt - \gamma v_t dt + \sqrt{2\gamma k_B T M^{-1}} dW_t. \end{aligned} \quad (19)$$

Here,  $x_t$  is the position at time  $t$ ,  $v_t$  is the velocity,  $M$  is the mass matrix (symmetric positive definite),  $\mathcal{E}(x)$  is the potential energy function, and  $\nabla \mathcal{E}(x_t)$  is its gradient with respect to position, i.e., the negative force. The parameter  $\gamma$  is the friction coefficient,  $k_B$  is the Boltzmann constant,  $T$  is the absolute temperature, and  $W_t$  denotes standard Brownian motion.

For ALDP, we use underdamped Langevin dynamics as MD with high temperature(600K). We use Euler-Maruyama integration to discretize the Langevin dynamics. As in MALA, we compute  $\log \hat{Z}$  for the initialization of  $\log Z_\theta$  in Learner, using Equation (18).

## A.2 Metrics

In this subsection, we formally define the evaluation metrics used to assess the Learner’s quality. All metrics are derived from the same importance-weight formulation based on the target partition function.

We begin with the exact log partition function  $\log Z$ , which can be written using forward-path importance sampling. Let  $\tau = (x_0, x_{\Delta t}, \dots, x_1)$  denote a sample trajectory drawn from the forward policy  $P_F(\tau)$ , and let  $R(x_1)$  be the reward associated with the final state  $x_1$ . Then, the partition function can be expressed as

$$\log Z = \log \mathbb{E}_{\tau \sim P_F(\tau)} \left( \frac{R(x_1) P_B(\tau | x_1)}{P_F(\tau)} \right), \quad (20)$$

where  $P_B(\tau | x_1)$  is the backward policy conditioned on the final state.

Since directly optimizing this quantity is intractable, we use two surrogate bounds. The first is the evidence lower Bound (ELBO), defined as

$$\text{ELBO} = \mathbb{E}_{\tau \sim P_F(\tau)} \left[ \log \frac{R(x_1) P_B(\tau | x_1)}{P_F(\tau)} \right]. \quad (21)$$

By Jensen’s inequality, ELBO is always a lower bound on the true  $\log Z$ . It is commonly used as a training objective and can reflect how well the forward policy  $P_F$  concentrates on high-reward trajectories. However, ELBO can be misleading in practice. A high ELBO does not necessarily imply

that all important modes are captured, as the forward policy may collapse to a small subset of modes while still achieving high reward [7].

To address this limitation, we also evaluate the evidence upper Bound (EUBO), which flips the sampling distribution:

$$\text{EUBO} = \mathbb{E}_{\tau \sim P_B(\tau)} \left[ \log \frac{R(x_1)P_B(\tau | x_1)}{P_F(\tau)} \right]. \quad (22)$$

Unlike ELBO, EUBO acts as a diagnostic metric. It is an upper bound of  $\log Z$  and penalizes missing probability mass. EUBO is driven to penalize missing probability mass and therefore exposes mode-collapse that ELBO may hide [7]. And then, true  $\log Z$  is consequently bounded by two bounds, i.e.,  $\text{ELBO} \leq \log Z \leq \text{EUBO}$ .

A smaller gap between the two bounds yields a tighter estimate of  $\log Z$ , making this gap a useful indicator of the Learner’s sampling quality.

Table 4: Searcher configurations of SGDS

Benchmark	40GMM	Manywell 32	Manywell 64	Manywell 128	LJ-13	LJ-55	ALDP
Type	MALA	AIS	AIS	AIS	MALA	MALA	MD
# of Chains	300	60K	60K	60K	16	1	4
Chain length	4K	100	100	100	4K	10K	110K
Burn-in	2K	-	-	-	2K	4K	10K
init. step size	1e-3	1e-3	1e-3	1e-3	1e-5	1e-5	0.5fs

Table 5: Learner configurations of SGDS

Benchmark	40GMM	Manywell 32	Manywell 64	Manywell 128	LJ-13	LJ-55	ALDP
Brownian bridge std ( $\sigma$ )	10.0	1.0	1.0	1.0	0.2	0.2	0.2
Buffer size	600k	60k	60k	60k	50K	10K	800K
Batch size	300	300	300	300	32	4	16
Architecture	MLP	MLP	MLP	MLP	EGNN	EGNN	EGNN
hidden dim	256	256	256	256	64	64	128
# of layers	2	2	2	2	5	5	5
RND weight	100	100	100	100	10	1	10

### A.3 Experimental setup

For the diffusion-based neural samplers, we follow the setup of [38].

**Gaussian mixture model with 40 modes (40GMM).** Training proceeds in one or two rounds. Our framework achieves competitive performance against baselines even with only a single round, and shows marginal improvement with a second round. We use MALA as the Searcher, running 300 parallel chains of length 4K, discarding the first 2K steps as burn-in. We maintain a target acceptance rate of 57.4% through step size scheduling, resulting in a total of 2.4M energy evaluations. We use the Gaussian prior with a standard deviation of 21.0 for MALA.

All methods adopt the PIS architecture [45, 38], with a joint network consisting of a two-layer MLP with 256 hidden dimensions. The RND network consists of three layers in the predictor network and the target network, with 256 hidden dimensions. We adopt Brownian bridges as the backward process, with a Brownian motion coefficient of 10.0. We run 25K epochs in both the first round and the second round.

**Manywell distributions.** We proceed with one or two rounds for training on Manywell distributions. We use AIS as the Searcher, running 60K parallel chains (3K chains \* 20 iterations) of length 100, only taking the final step samples. We use the Gaussian prior with a standard deviation of 1.0.

All methods adopt the PIS architecture [45, 38], with a joint network consisting of a two-layer MLP with 256 hidden dimensions. The RND network consists of three layers in the predictor network and the target network, with 256 hidden dimensions. We adopt Brownian bridges as the backward process, with a Brownian motion coefficient of 1.0. We run 25K epochs in the first round and 30K in the second round.

**Lennard-Jones (LJ) potentials.** Training proceeds in two rounds. We use MALA as the Searcher for two rounds: in LJ-13 we run 16 parallel chains of length 4K corresponding to 64K energy evaluations, discarding the first 2K steps as burn-in and retaining 57.4% accepted samples among remaining 32K samples; in LJ-55 we run a single chain of length 10K corresponding to 10K energy evaluations, discarding the first 4K steps and retaining 57.4% accepted samples among remaining 6K samples. We use the Gaussian prior with a standard deviation of 1.75 for MALA.

All methods utilize five EGNN layers with 64 hidden dimensions. Following [19, 22], we design an  $E(3)$ -equivariant generative model initialized from a Dirac delta at the origin, using a mean-free forward transition kernel in inference. The RND network comprises three layers in the predictor network and two in the target network. We adopt Brownian bridges as the backward process for diffusion-based neural samplers, with a Brownian motion coefficient of 0.2. For LJ-13, we run 5K epochs in the first round and 10K in the second round; for LJ-55, 10K and then 20K epochs.

Specifically, we note that the reported performance of the iDEM on Table 2 differs from the original paper [1] due to adjustments, except  $\sigma_{\max}$  and  $\sigma_{\min}$  of the noise scheduling, made to avoid significant discrepancies in energy call usage compared to our method. We reduce the EGNN hidden dimension to 64 and the batch size to 8, and limit the total number of training epochs, including both inner and outer loops, to 15K accordingly. And while the latest iDEM codebase employs 10 steps of Langevin dynamics refinement before evaluation, particularly for LJ-55, we omit this step for fair comparison and instead set the number of samples for MC estimation to 1K. While iDEM reports a lower bound of  $\log Z$  computed via importance sampling with its learned proposal density  $q(x)$  given by OT-CFM model, we omit this result in our tables. We compute the lower bound based on trajectory-level estimators without training auxiliary models, i.e., CFM. Thus, our reported values are not directly comparable to those from iDEM.

Additionally, in LJ-55, we maximize the log-likelihood of the forward path distribution under the backward process for the first 5K epochs of each round, discretizing backward paths from Brownian bridges initialized with empirical samples collected by Searchers. We also use randomized time scheduling introduced in [6] for our method. We train PIS at a learning rate of  $1e-4$ , TB at a learning rate of  $2e-4$ , and SGDS at a learning rate of  $5e-4$ . We use 4 and 32 batch sizes for all methods except PIS in LJ-13 and LJ-55, respectively. For PIS, we halve these sizes due to the memory limitation required by the forward SDE computational graph.

**Alanine Dipeptide.** We perform two rounds of search using under-damped Langevin dynamics. In each round, we run four parallel simulations of 55 ps each, with a time step of 0.5 fs, requiring 440K energy evaluations. We discard the first 5 ps of each trajectory as burn-in, then collect 400K samples. Each simulation starts from the same initial position drawn from a Dirac delta distribution, with all initial velocities set to zero. We integrate equations of motion using the Euler-Maruyama integrator, set the friction coefficient  $\gamma = 1$ , and use temperature  $T = 600K$  for the first round Searcher and  $T = 300K$  for the second round Searcher.

Similar to LJ potentials, all models utilize five EGNN layers with 128 hidden dimensions. We use a Dirac delta prior distribution at the origin and a mean-free forward transition kernel to guarantee  $E(3)$ -equivariance of the marginal density in inference. The Learner network comprises five EGNN layers, while the predictor network and target network in the RND framework contain three and two layers, respectively. As in LJ potentials, we use the Brownian motion coefficient of 0.2. We run 10K epochs in the first round and 20K epochs in the second. As in LJ-55, we maximize the log-likelihood for the first 5K epochs each round. We also utilize randomized time scheduling for our method. We train PIS at a learning rate of  $1e-4$  and all other methods at  $5e-4$ . We use a 16 batch size for all methods except PIS, which uses an 8 batch size due to the memory limitation required by the forward SDE computational graph.

In inference time, we follow [22]. We first align the topology of generated samples with the target bond graph since the architecture and machine learning potential have a degree of freedom in atom ordering. We first match the bond graphs of generated samples with a given bond graph of interest and then correct the chirality of the generated sample to fit the target molecular configuration. The generated sample is rejected if the bond graph is not isomorphic to the target bond graph.

#### A.4 Task details

**40-Component Gaussian Mixture Model (40GMM).** The 40-component Gaussian Mixture Model (GMM) consists of a mixture distribution of 40 Gaussian components, each characterized by a distinct

Table 6: ELBO, EUBO, their gap, and Energy calls on 40GMM and Manywell-32.

Method	40GMM ( $d = 2$ )				Manywell ( $d = 32$ )			
	ELBO $\uparrow$	EUBO $\downarrow$	Gap $\downarrow$	Energy calls	ELBO $\uparrow$	EUBO $\downarrow$	Gap $\downarrow$	Energy calls
PIS+LP	$-1.32 \pm 0.07$	$2.42 \pm 0.20$	$3.75 \pm 0.22$	300M	$160.83 \pm 0.41$	$180.49 \pm 4.76$	$19.66 \pm 4.78$	300M
TB+LP	$-0.35 \pm 0.03$	$0.53 \pm 0.04$	$0.87 \pm 0.03$	160M	$161.42 \pm 0.40$	$195.89 \pm 8.14$	$34.37 \pm 8.15$	300M
FL-SubTB+LP	$-0.36 \pm 0.01$	$0.58 \pm 0.08$	$0.94 \pm 0.07$	260M	$160.74 \pm 0.15$	$215.93 \pm 4.52$	$55.19 \pm 4.52$	330M
TB+LS+LP	$-0.38 \pm 0.03$	$0.32 \pm 0.02$	$0.69 \pm 0.02$	320M	$162.95 \pm 0.08$	$166.30 \pm 0.11$	$3.35 \pm 0.14$	320M
TB+Expl+LP	$-0.37 \pm 0.01$	$0.32 \pm 0.02$	$0.69 \pm 0.02$	300M	$160.76 \pm 0.13$	$215.92 \pm 14.90$	$55.16 \pm 14.90$	300M
TB+Expl+LS+LP	$-0.37 \pm 0.01$	$0.34 \pm 0.02$	$0.71 \pm 0.02$	320M	$162.97 \pm 0.06$	$166.25 \pm 0.10$	$3.28 \pm 0.12$	320M
PIS	$-2.03 \pm 0.22$	$55.48 \pm 10.71$	$57.50 \pm 9.02$	100M	$159.71 \pm 1.70$	$333.79 \pm 3.98$	$174.08 \pm 4.33$	100M
TB	$-1.35 \pm 0.04$	$99.04 \pm 6.01$	$100.40 \pm 5.67$	100M	$160.58 \pm 0.87$	$439.28 \pm 166.52$	$278.70 \pm 166.49$	100M
TB+LS	$-0.38 \pm 0.03$	$0.83 \pm 0.46$	$1.21 \pm 0.38$	290M	$163.12 \pm 0.10$	$166.05 \pm 0.12$	$2.93 \pm 0.16$	290M
TB+Expl+LS	$-0.38 \pm 0.05$	$0.58 \pm 0.34$	$0.96 \pm 0.34$	290M	$160.87 \pm 3.31$	$168.27 \pm 1.49$	$7.40 \pm 3.63$	290M
GAFN	*	*	*	N/A	$161.02 \pm 0.05$	$282.40 \pm 2.02$	$121.38 \pm 2.02$	100M
iDEM	$-2.14 \pm 0.45$	$12.75 \pm 3.67$	$14.89 \pm 3.70$	300M	$142.23 \pm 0.40$	$211.56 \pm 2.53$	$69.33 \pm 2.56$	300M
<b>Ours (round 1)</b>	$-0.40 \pm 0.01$	$0.33 \pm 0.02$	$0.73 \pm 0.02$	<b>6M</b>	$162.49 \pm 0.05$	$166.60 \pm 0.01$	$4.11 \pm 0.05$	<b>9M</b>
<b>Ours (round 2)</b>	$-0.40 \pm 0.03$	$0.33 \pm 0.05$	$0.73 \pm 0.05$	<b>12M</b>	$162.63 \pm 0.01$	$166.48 \pm 0.03$	$3.85 \pm 0.03$	<b>20M</b>

mean vector  $\mu_i$ . The energy function for the GMM is defined as:

$$\mathcal{E}(x) = -\log \left( \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; \mu_i, \sigma^2 I) \right),$$

where  $n = 40$ , the weight of each  $i$ -th Gaussian component is the same, and  $\mathcal{N}(x; \mu_i, \sigma^2 I)$  is the probability density function of the multivariate Gaussian distribution.

**ManyWell distributions.** The Manywell potential describes a high-dimensional energy landscape containing multiple wells (local minima), each representing stable states with distinct energy levels. The energy function of Manywell distribution is given by:

$$\mathcal{E}(x) = \sum_{k=1}^n (x_{2k-1}^4 - 6x_{2k-1}^2 - \frac{1}{2}x_{2k-1} + \frac{1}{2}x_{2k}^2) + C,$$

where  $n = d/2$  is the number of wells, and  $d$  is the dimensionality of the landscape. Adjusting the dimensionality  $d = 2n$  allows varying the number of wells and complexity, creating tasks like Manywell-32, Manywell-64, and Manywell-128.

**Lennard-Jones (LJ) potentials.** The Lennard-Jones potential models the interactions between particles. The energy function is defined as:

$$\mathcal{E}(x) = 2\kappa \sum_{1 \leq i < j \leq N} \epsilon \left[ \left( \frac{\sigma}{r_{ij}} \right)^{12} - 2 \left( \frac{\sigma}{r_{ij}} \right)^6 \right] + \frac{\lambda}{2} \sum_{i=1}^N \|r_i - r_{cm}\|^2, \quad (23)$$

where  $\epsilon$  and  $\kappa$  are parameters defining the depth of the potential well and the energy factor, respectively.  $r_{ij} = \|x_i - x_j\|$  represents the Euclidean distance between particles  $i$  and  $j$ .  $\sigma$  is the characteristic distance at which the potential between two particles vanishes, often interpreted as the van der Waals radius. In our experiments, we set all parameters to 1.0, i.e.,  $\kappa = \epsilon = \sigma = \lambda = 1.0$ . Adjusting the number of particles creates tasks such as LJ-13 and LJ-55, increasing the complexity of the particle interactions and resulting in a rugged energy landscape.

**TorchANI potential for Alanine Dipeptide.** We leverage TorchANI [16], a PyTorch implementation of ANI deep-learning potentials trained on quantum-mechanical reference data, to accurately calculate molecular energies. It provides transferable machine learning potential trained on organic molecules for efficient energy and force evaluations with accuracy comparable to density-functional theory (DFT). In particular, TorchANI excels at modeling small to medium-sized organic molecules such as alanine dipeptide.

## B Additional experimental results

### B.1 Low-dimensional standard benchmarks

**Baselines and settings.** We benchmark our framework on two standard low-dimensional tasks: 40GMM and Manywell-32. Consistent with the high-dimensional experiments, we report ELBO,

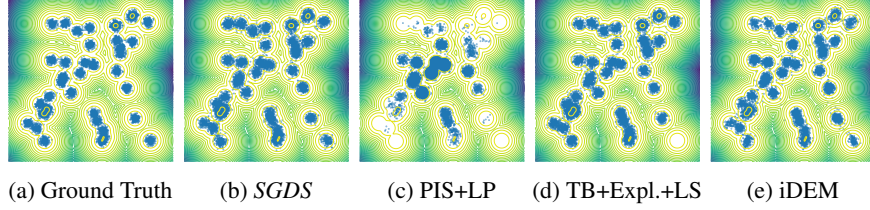


Figure 5: Mode coverage comparison on 40GMM.

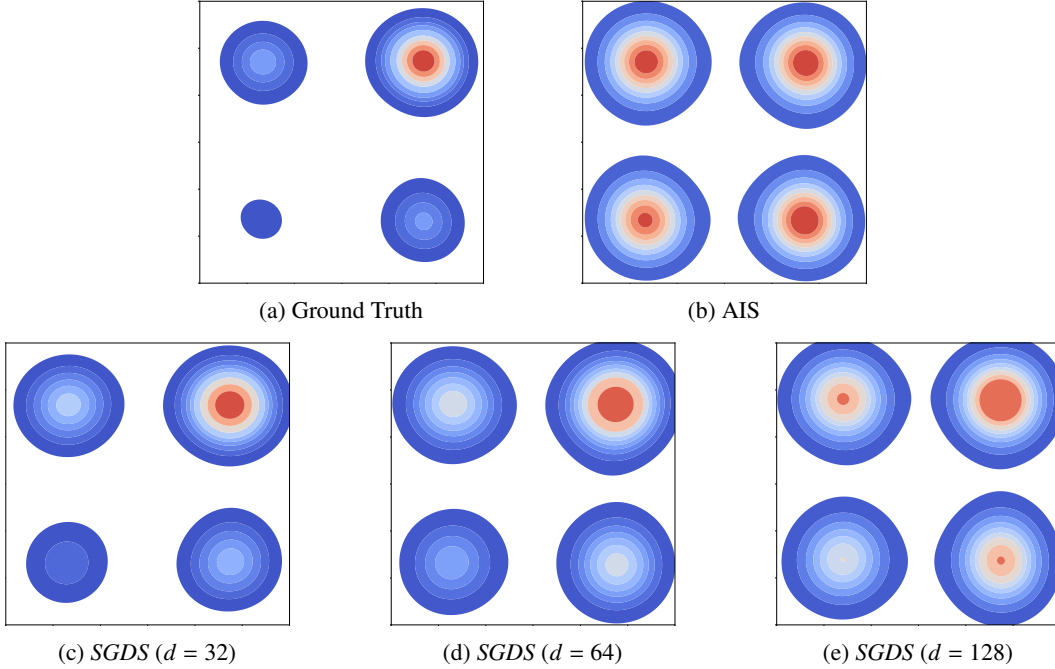


Figure 6: KDE figures of AIS( $T = 100$ ), ours, and true samples on Manywell-32/64/128.

EUBO, their gap, and the number of energy calls required during training. We employ the same baseline methods and trajectory configurations (including trajectory length and training objectives) as in the high-dimensional settings. We provide detailed configurations, including diffusion scales for each task, in [Appendix A](#).

**Results.** As demonstrated in [Table 6](#), our method achieves competitive performance on lower-dimensional standard tasks, producing EUBO and ELBO metrics comparable to the strongest baselines, while using significantly fewer energy calls. On the 40GMM task, despite some baselines reporting strong ELBO and EUBO scores, they notably fail to capture the mode located at the bottom-right corner (see [Figure 5](#)). In contrast, our framework reliably identifies all modes without sacrificing performance metrics. We report both the first-round and second-round performances of our method, showing that our method attains robust performance on low-dimensional tasks even in the first round, with a slight but consistent improvement observed in the second round.

## B.2 Debiasing of Learner from MCMC Searcher

To address potential biases inherent in MCMC sampling due to finite-length chains, our framework incorporates both off-policy TB training using samples from the Searcher and on-policy TB training. This design choice aims to mitigate biases arising from the Searcher samples alone by enabling the Learner model to adjust toward the target distribution.

To evaluate whether the Learner effectively debiases the samples collected by the Searcher, we compare kernel density estimations (KDE) of samples obtained by the AIS Searcher with those



generated by the on/off-policy TB Learner on Manywell distributions. [Figure 6](#) illustrates these KDE comparisons across dimensions 32, 64, and 128.

Due to the varying mode masses assigned in Manywell distributions, even when AIS successfully covers all modes with limited budgets, it struggles to precisely capture the relative mode masses. In contrast, the KDE of the samples generated by the Learner aligns more closely with the true density, effectively reflecting the relative importance of different modes. This result highlights the effectiveness of combining on- and off-policy training to achieve better density approximation than relying solely on finite-budget AIS samples.

## C Limitations

While our framework demonstrates strong empirical performance, several limitations remain.

First, the effectiveness of intrinsic rewards from RND depends on careful tuning of the novelty scale parameter  $\alpha$ . Poorly calibrated  $\alpha$  can overly emphasize exploration, producing noisy or irrelevant samples, or conversely yield overly conservative exploration. This could be mitigated by employing adaptive strategies that dynamically adjust  $\alpha$  during sampling based on diversity metrics or exploration progress signals.

Additionally, the quality of samples provided by the Searcher sets a fundamental exploration limit. If the Searcher fails to adequately explore challenging modes, the Learner will inevitably inherit these limitations, particularly in high-barrier energy landscapes. Introducing enhanced exploration strategies, such as parallel tempering or more advanced proposal schemes like HMC, could improve coverage of hard-to-sample modes.