S^{2}AC: ENERGY-BASED REINFORCEMENT LEARNING WITH STEIN SOFT ACTOR CRITIC

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning expressive stochastic policies instead of deterministic ones has been proposed to achieve better stability, sample complexity, and robustness. Notably, in Maximum Entropy Reinforcement Learning (MaxEnt RL), the policy is modeled as an expressive Energy-Based Model (EBM) over the Q-values. However, this formulation requires the estimation of the entropy of such EBMs, which is an open problem. To address this, previous MaxEnt RL methods either implicitly estimate the entropy, resulting in high computational complexity and variance (SQL), or follow a variational inference procedure that fits simplified actor distributions (*e.g.*, Gaussian) for tractability (SAC). We propose Stein Soft Actor-Critic (S^{2}AC), a MaxEnt RL algorithm that learns expressive policies without compromising efficiency. Specifically, S^{2}AC uses parameterized Stein Variational Gradient Descent (SVGD) as the underlying policy. We derive a closed-form expression of the entropy of such policies. Our formula is computationally efficient and only depends on first-order derivatives and vector products. Empirical results show that S^{2}AC yields more optimal solutions to the MaxEnt objective than SQL and SAC in the multi-goal environment, and outperforms SAC and SQL on the MuJoCo benchmark. Our code is available at: `https://anonymous.4open.science/r/Stein-Soft-Actor-Critic/`

1 INTRODUCTION

MaxEnt RL (Todorov, 2006; Ziebart, 2010; Haarnoja et al., 2017; Kappen, 2005; Toussaint, 2009; Theodorou et al., 2010; Abdolmaleki et al., 2018; Haarnoja et al., 2018a; Vieillard et al., 2020) has been proposed to address challenges hampering the deployment of RL to real-world applications, including stability, sample efficiency (Gu et al., 2017), and robustness (Eysenbach & Levine, 2021). Instead of learning a deterministic policy, as in classical RL (Sutton et al., 1999; Schulman et al., 2017; Silver et al., 2014; Lillicrap et al., 2015), MaxEnt RL learns a stochastic policy that captures the intricacies of the action space. This enables better exploration during training and eventually better robustness to environmental perturbations at test time, *i.e.*, the agent learns multimodal action space distributions which enables picking the next best action in case a perturbation prevents the execution of the optimal one. To achieve this, MaxEnt RL models the policy using the expressive family of EBMs (LeCun et al., 2006). This translates into learning policies that maximize the sum of expected future reward and expected future entropy. However, estimating the entropy of such complex distributions remains an open problem.

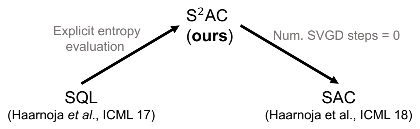

S²AC
(ours)
Explicit entropy evaluation
Num. SVGD steps = 0
SQL
(Haarnoja et al., ICML 17)
SAC
(Haarnoja et al., ICML 18)

Figure 1: Comparing S^{2}AC to SQL and SAC. S^{2}AC with a parameterized policy is reduced to SAC if the number of SVGD steps is 0. SQL becomes equivalent to S^{2}AC if the entropy is evaluated explicitly with our derived formula.

To address this, existing approaches either use tricks to go around the entropy computation or make limiting assumptions on the policy. This results in either poor scalability or convergence to suboptimal solutions. For example, SQL (Haarnoja et al., 2017) implicitly incorporates entropy in the Q-function computation. This requires using importance sampling, which results in high variability and hence poor training stability and limited scalability to high dimensional action spaces. SAC (Haarnoja et al., 2018a), on the other hand, follows a variational inference procedure by fitting a Gaussian distribution to the EBM policy. This enables a closed-form evaluation of the entropy but results in a suboptimal solution. For instance, SAC fails in environments characterized by multimodal

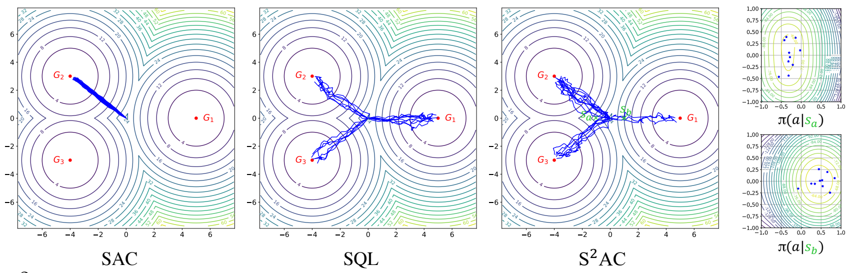


Figure 2: S^{2}AC learns a more optimal solution to the MaxEnt RL objective than SAC and SQL. We design a multigoal environment where an agent starts from the center of the 2-d map and tries to reach one of the three goals (G_1, G_2, and G_3). The maximum expected future reward (level curves) is the same for all the goals but the expected future entropy is different (higher on the path to G_2/G_3): the action distribution $\pi(a|s)$ is bi-modal on the path to the left (G_2 and G_3) and unimodal to the right (G_1). Hence, we expect the optimal policy for the MaxEnt RL objective to assign more weights to G_2 and G_3. We visualize trajectories (in blue) sampled from the policies learned using SAC, SQL, and S^{2}AC. SAC quickly commits to a single mode due to its actor being tied to a Gaussian policy. Though SQL also recovers the three modes, the trajectories are evenly distributed. S^{2}AC recovers all the modes and approaches the left two goals more frequently. This indicates that it successfully maximizes not only the expected future reward but also the expected future entropy.

action distributions. Similar to SAC, IAPO (Marino et al., 2021) models the policy as a uni-modal Gaussian. Instead of optimizing a MaxEnt objective, it achieves multimodal policies by learning a collection of parameter estimates (mean, variance) through different initializations for different policies. To improve the expressiveness of SAC, SSPG (Cetin & Celiktutan, 2022) and SAC-NF (Mazoure et al., 2020) model the policy as a Markov chain with Gaussian transition probabilities and as a normalizing flow (Rezende & Mohamed, 2015), respectively. However, due to training stability issues, the reported results in Cetin & Celiktutan (2022) show that though both models learn multi-modal policies, they fail to maximize the expected future entropy in positive rewards setups.

We propose a new algorithm, S^{2}AC, that yields a more optimal solution to the MaxEnt RL objective. To achieve *expressivity*, S^{2}AC models the policy as a Stein Variational Gradient Descent (SVGD) (Liu, 2017) sampler from an EBM over Q-values (target distribution). SVGD proceeds by first sampling a set of particles from an initial distribution, and then iteratively transforming these particles via a sequence of updates to fit the target distribution. To compute a *closed-form estimate of the entropy* of such policies, we use the change-of-variable formula for pdfs (Devore et al., 2012). We prove that this is only possible due to the invertibility of the SVGD update rule, which does not necessarily hold for other popular samplers (*e.g.*, Langevin Dynamics (Welling & Teh, 2011)). While normalizing flow models (Rezende & Mohamed, 2015) are also invertible, SVGD-based policy is more expressive as it encodes the inductive bias about the unnormalized density and incorporates a dispersion term to encourage multi-modality, whereas normalizing flows encode a restrictive class of invertible transformations (with easy-to-estimate Jacobian determinants). Moreover, our formula is computationally efficient and only requires evaluating first-order derivatives and vector products. To improve *scalability*, we model the initial distribution of the SVGD sampler as an isotropic Gaussian and learn its parameters, *i.e.*, mean and standard deviation, end-to-end. We show that this results in faster convergence to the target distribution, *i.e.*, fewer SVGD steps. Intuitively, the initial distribution learns to contour the high-density region of the target distribution while the SVGD updates result in better and faster convergence to the modes within that region. Hence, our approach is as parameter efficient as SAC, since the SVGD updates do not introduce additional trainable parameters.

Note that S^{2}AC can be reduced to SAC when the number of SVGD steps is zero. Also, SQL becomes equivalent to S^{2}AC if the entropy is computed explicitly using our formula (the policy in SQL is an amortized SVGD sampler). Beyond RL, the backbone of S^{2}AC is a new variational inference algorithm with a more expressive and scalable distribution characterized by a closed-form entropy estimate. We believe that this variational distribution can have a wider range of exciting applications.

We conduct extensive empirical evaluations of S^{2}AC from three aspects. We start with a sanity check on the merit of our derived SVGD-based entropy estimate on target distributions with known entropy values (*e.g.*, Gaussian) or log-likelihoods (*e.g.*, Gaussian Mixture Models) and assess its sensitivity to different SVGD parameters (kernel, initial distribution, number of steps and number of particles). We observe that its performance depends on the choice of the kernel and is robust to variations of the remaining parameters. In particular, we find out that the kernel should be chosen to guarantee inter-dependencies between the particles, which turns out to be essential for invertibility. Next, we assess the performance of S^{2}AC on a multi-goal environment (Haarnoja et al., 2017) where different goals are associated with the same positive (maximum) expected future reward but different (maximum) expected future entropy. We show that S^{2}AC learns multimodal policies and effectively maximizes the entropy, leading to better robustness to obstacles placed at test time. Finally, we test S^{2}AC on the MuJoCo benchmark (Duan et al., 2016). S^{2}AC yields better performances than the baselines on four out of the five environments. Moreover, S^{2}AC shows higher sample efficiency as it tends to converge with fewer training steps. These results were obtained from running SVGD for only three steps, which results in a small overhead compared to SAC during training. Furthermore, to maximize the run-time efficiency during testing, we train an amortized SVGD version of the policy to mimic the SVGD-based policy. Hence, this reduces inference to a forward pass through the policy network without compromising the performance.

2 PRELIMINARIES

2.1 SAMPLERS FOR ENERGY-BASED MODELS

In this work, we study three representative methods for sampling from EBMs: (1) Stochastic Gradient Langevin Dynamics (SGLD) & Deterministic Langevin Dynamics (DLD) (Welling & Teh, 2011), (2) Hamiltonian Monte Carlo (HMC) (Neal et al., 2011), and (3) Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016). We review SVGD here since it is the sampler we eventually use in S^{2}AC, and leave the rest to Appendix C.1. SVGD is a particle-based Bayesian inference algorithm. Compared to SGLD and HMC which have a single particle in their dynamics, SVGD operates on a set of particles. Specifically, SVGD samples a set of m particles $\{a_j\}_{j=1}^m$ from an initial distribution q^0 which it then transforms through a sequence of updates to fit the target distribution. Formally, at every iteration l, SVGD applies a form of functional gradient descent Δf that minimizes the KL-divergence between the target distribution p and the proposal distribution q^l induced by the particles, *i.e.*, the update rule for the i^{th} particles is: $a_i^{l+1} = a_i^l + \epsilon \Delta f(a_i^l)$ with

$$\Delta f(a_i^l) = \mathbb{E}_{a_j^l \sim q^l}\left[k(a_i^l, a_j^l)\nabla_{a_j^l} \log p(a_j^l) + \nabla_{a_j^l} k(a_i^l, a_j^l)\right]. \tag{1}$$

Here, ϵ is the step size and $k(\cdot,\cdot)$ is the kernel function, *e.g.*, the RBF kernel: $k(a_i, a_j) = \exp(||a_i - a_j||^2/2\sigma^2)$. The first term within the gradient drives the particles toward the high probability regions of p, while the second term serves as a repulsive force to encourage dispersion.

2.2 MAXIMUM-ENTROPY RL

We consider an infinite horizon Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space and $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, \infty]$ is the state transition probability modeling the density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$. Additionally, we assume that the environment emits a bounded reward function $r \in [r_{\text{min}}, r_{\text{max}}]$ at every iteration. We use $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ to denote the state and state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$. We consider the setup of continuous action spaces Lazaric et al. (2007); Lee et al. (2018); Zhou & Lu (2023). MaxEnt RL (Todorov, 2006; Ziebart, 2010; Rawlik et al., 2012) learns a policy $\pi^*(a_t|s_t)$, that instead of maximizing the expected future reward, maximizes the sum of the expected future reward and entropy:

$$\pi^* = \arg\max_\pi \sum_t \gamma^t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right], \tag{2}$$

where α is a temperature parameter controlling the stochasticity of the policy and $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy at state s_t. The conventional RL objective can be recovered for $\alpha = 0$. Note that the MaxEnt RL objective above is equivalent to approximating the policy, modeled as an EBM over Q-values, by a variational distribution $\pi(a_t|s_t)$ (see proof of equivalence in Appendix D), *i.e.*,

$$\pi^* = \arg\min_\pi \sum_t \mathbb{E}_{s_t\sim\rho_\pi}\left[D_{KL}\left(\pi(\cdot|s_t) \| \exp(Q(s_t,\cdot)/\alpha)/Z\right)\right], \tag{3}$$

where D_{KL} is the KL-divergence and Z is the normalizing constant. We now review two landmark MaxEnt RL algorithms: SAC (Haarnoja et al., 2018a) and SQL (Haarnoja et al., 2017).

SAC is an actor-critic algorithm that alternates between policy evaluation, *i.e.*, evaluating the Q-values for a policy $\pi_\theta(a_t|s_t)$:

$$Q_\phi(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma\, \mathbb{E}_{s_{t+1},a_{t+1}\sim\rho_{\pi_\theta}}\left[Q_\phi(s_{t+1}, a_{t+1}) + \alpha\mathcal{H}(\pi_\theta(\cdot|s_{t+1}))\right] \tag{4}$$

and policy improvement, *i.e.*, using the updated Q-values to compute a better policy:

$$\pi_\theta = \arg\max_\theta \sum_t \mathbb{E}_{s_t, a_t \sim \rho_{\pi_\theta}} \left[Q_\phi(a_t, s_t) + \alpha \mathcal{H}(\pi_\theta(\cdot|s_t)) \right]. \tag{5}$$

SAC models π_θ as an isotropic Gaussian, *i.e.*, $\pi_\theta(\cdot|s) = \mathcal{N}(\mu_\theta, \sigma_\theta I)$. While this enables computing a closed-form expression of the entropy, it incurs an over-simplification of the true action distribution, and thus cannot represent complex distributions, *e.g.*, multimodal distributions.

SQL goes around the entropy computation, by defining a soft version of the value function $V_\phi = \alpha \log \big(\int_{\mathcal{A}} \exp \big(\frac{1}{\alpha} Q_\phi(s_t, a') \big) da' \big)$. This enables expressing the Q-value (Eq (4)) independently from the entropy, *i.e.*, $Q_\phi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_\phi(s_{t+1})]$. Hence, SQL follows a soft value iteration which alternates between the updates of the "soft" versions of Q and value functions:

$$Q_\phi(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_\phi(s_{t+1})], \; \forall (s_t, a_t) \tag{6}$$

$$V_\phi(s_t) \leftarrow \alpha \log \big(\int_{\mathcal{A}} \exp \big(\tfrac{1}{\alpha} Q_\phi(s_t, a') \big) da' \big), \; \forall s_t. \tag{7}$$

Once the Q_ϕ and V_ϕ functions converge, SQL uses amortized SVGD Wang & Liu (2016) to learn a stochastic sampling network $f_\theta(\xi, s_t)$ that maps noise samples ξ into the action samples from the EBM policy distribution $\pi^*(a_t|s_t) = \exp \big(\frac{1}{\alpha} (Q^*(s_t, a_t) - V^*(s_t)) \big)$. The parameters θ are obtained by minimizing the loss $J_\theta(s_t) = D_{KL}\big(\pi_\theta(\cdot|s_t) || \exp\big(\frac{1}{\alpha} (Q^*_\phi(s_t, \cdot) - V^*_\phi(s_t)) \big) \big)$ with respect to θ. Here, π_θ denotes the policy induced by f_θ. SVGD is designed to minimize such KL-divergence without explicitly computing π_θ. In particular, SVGD provides the most greedy direction as a functional $\Delta f_\theta(\cdot, s_t)$ (Eq (1)) which can be used to approximate the gradient $\partial J_\theta / \partial a_t$. Hence, the gradient of the loss J_θ with respect to θ is: $\partial J_\theta(s_t)/\partial \theta \propto \mathbb{E}_\xi \big[\Delta f_\theta(\xi, s_t) \partial f_\theta(\xi, s_t)/\partial \theta \big]$. Note that the integral in Eq (7) is approximated via importance sampling, which is known to result in high variance estimates and hence poor scalability to high dimensional action spaces. Moreover, amortized generation is usually unstable and prone to mode collapse, an issue similar to GANs. Therefore, SQL is outperformed by SAC Haarnoja et al. (2018a) on benchmark tasks like MuJoCo.

3 Approach

We introduce S^{2}AC, a new actor-critic MaxEnt RL algorithm that uses SVGD as the underlying actor to generate action samples from policies represented using EBMs. This choice is motivated by the expressivity of distributions that can be fitted via SVGD. Additionally, we show that we can derive a closed-form entropy estimate of the SVGD-induced distribution, thanks to the invertibility of the update rule, which does not necessarily hold for other EBM samplers. Besides, we propose a parameterized version of SVGD to enable scalability to high-dimensional action spaces and non-smooth Q-function landscapes. S^{2}AC is hence capable of learning a more optimal solution to the MaxEnt RL objective (Eq (2)) as illustrated in Figure 2.

3.1 Stein Soft Actor Critic

Like SAC, S^{2}AC performs soft policy iteration which alternates between policy evaluation and policy improvement. The difference is that we model the actor as a *parameterized sampler from an EBM*. Hence, the policy distribution corresponds to an expressive EBM as opposed to a Gaussian.

Critic. The critic's parameters ϕ are obtained by minimizing the Bellman loss as traditionally:

$$\phi^* = \arg\min_\phi \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi_\theta}} \left[(Q_\phi(s_t, a_t) - \hat{y})^2 \right], \tag{8}$$

with the target $\hat{y} = r_t(s_t, a_t) + \gamma \mathbb{E}_{(s_{t+1}, a_{t+1}) \sim \rho_\pi} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|s_{t+1})) \right]$. Here $\bar{\phi}$ is an exponentially moving average of the value network weights (Mnih et al., 2015).

Actor as an EBM sampler. The actor is modeled as a sampler from an EBM over the Q-values. To generate a set of valid actions, the actor first samples a set of particles $\{a^0\}$ from an initial distribution q^0 (*e.g.*, Gaussian). These particles are then updated over several iterations $l \in [1, L]$, *i.e.*, $\{a^{l+1}\} \leftarrow \{a^l\} + \epsilon h(\{a^l\}, s)$ following the sampler dynamics characterized by a transformation h (*e.g.*, for SVGD, $h = \Delta f$ in Eq (1)). If q^0 is tractable and h is invertible, it's possible to compute a closed-form expression of the distribution of the particles at the l^{th} iteration via the change of variable formula Devore et al. (2012): $q^l(a^l|s) = q^{l-1}(a^{l-1}|s) \left| \det(I + \epsilon \nabla_{a^l} h(a^l, s)) \right|^{-1}, \forall l \in [1, L]$. In this case, the policy is represented using the particle distribution at the final step L of the sampler dynamics, *i.e.*, $\pi(a|s) = q^L(a^L|s)$ and the entropy can be estimated by averaging $\log q^L(a^L|s)$ over a set of particles (Section 3.2). We study the invertibility of popular EBM samplers in Section 3.3.

Parameterized initialization. To reduce the number of steps required to converge to the target distribution (hence reducing computation cost), we further propose modeling the initial distribution as a parameterized isotropic Gaussian, *i.e.*, $a^0 \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$. The parameterization trick is then used to express a^0 as a function of θ. Intuitively, the actor would learn θ such that the initial distribution is close to the target distribution. Hence, fewer steps are required to converge, as illustrated in Figure 3. Note that if the number of steps $L = 0$, S^{2}AC is reduced to SAC. Besides, to deal with the non-smooth nature of deep Q-function landscapes which might lead to particle divergence in the sampling process, we bound the particle updates to be within a few standard deviations (t) from the mean of the learned initial distribution, *i.e.*, $-t\sigma_\theta \leq a_\theta^l \leq t\sigma_\theta$, $\forall l \in [1, L]$. Eventually, the initial distribution q_θ^0 learns to contour the high-density region of the target distribution and the following updates refine it by converging to the spanned modes. Formally, the parameters θ are computed by minimizing the expected KL-divergence between the policy q_θ^L induced by the particles from the sampler and the EBM of the Q-values:

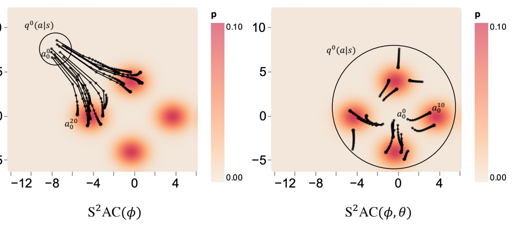


Figure 3: S^{2}AC(ϕ, θ) achieves faster convergence to the target distribution (in orange) than S^{2}AC(ϕ) by parameterizing the initial distribution $\mathcal{N}(\mu_\theta, \sigma_\theta)$ of the SVGD sampler.

$$\theta^* = \arg\max_\theta \mathbb{E}_{s_t \sim \mathcal{D}, a_\theta^L \sim \pi_\theta} \left[Q_\phi(s_t, a_\theta^L)\right] + \alpha \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathcal{H}(\pi_\theta(\cdot|s_t))\right]$$
$$\text{s.t. } -t\sigma_\theta \leq a_\theta^l \leq t\sigma_\theta, \quad \forall l \in [1, L]. \tag{9}$$

Here, $\mathcal{D}$ is the replay buffer. The derivation is in Appendix E. Note that the constraint does not truncate the particles as it is not an invertible transformation which then violates the assumptions of the change of variable formula. Instead, we sample more particles than we need and select the ones that stay within the range. We call S^{2}AC(ϕ, θ) and S^{2}AC(ϕ) as two versions of S^{2}AC with/without the parameterized initial distribution. The complete S^{2}AC algorithm is in Algorithm 1 of Appendix A.

3.2 A Closed-Form Expression of the Policy's Entropy

A critical challenge in MaxEnt RL is how to efficiently compute the entropy term $\mathcal{H}(\pi(\cdot|s_{t+1}))$ in Eq (2). We show that, if we model the policy as an iterative sampler from the EBM, under certain conditions, we can derive a closed-form estimate of the entropy at convergence.

Theorem 3.1. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible transformation of the form $F(a) = a + \epsilon h(a)$. We denote by $q^L(a^L)$ the distribution obtained from repeatedly applying F to a set of samples $\{a^0\}$ from an initial distribution $q^0(a^0)$ over L steps,* i.e., *$a^L = F \circ F \circ \cdots \circ F(a^0)$. Under the condition $\epsilon||\nabla_{a_i^l} h(a_i)||_\infty \ll 1$, $\forall l \in [1, L]$, the distribution of the particles at the L^{th} step is:*

$$\log q^L(a^L) \approx \log q^0(a^0) - \epsilon \sum_{l=0}^{L-1} \text{Tr}(\nabla_{a^l} h(a^l)) + \mathcal{O}(\epsilon^2 dL). \tag{10}$$

Here, d is the dimensionality of a, i.e., *$a \in \mathbb{R}^d$ and $\mathcal{O}(\epsilon^2 dL)$ is the order of approximation error.*
Proof Sketch: As F is invertible, we apply the change of variable formula (Appendix C.2) on the transformation $F \circ F \circ \cdots F$ and obtain: $\log q^L(a^L) = \log q^0(a^0) - \sum_{l=0}^{L-1} \log \left|\det(I + \epsilon \nabla_{a^l} h(a^l))\right|$. Under the assumption $\epsilon||\nabla_{a_i} h(a_i)||_\infty \ll 1$, we apply the corollary of Jacobi's formula (Appendix C.3) and get Eq. (10). The detailed proof is in Appendix F. Note that the condition $\epsilon||\nabla_{a_i} h(a_i)||_\infty \ll 1$ can always be satisfied when we choose a sufficiently small step size ϵ, *or* the gradient of $h(a)$ is small, *i.e.*, $h(a)$ is Lipschitz continuous with a sufficiently small constant.
It follows from the theorem above, that the entropy of a policy modeled as an EBM sampler (Eq (9)) can be expressed analytically as:

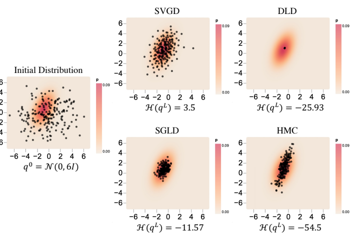
$$\mathcal{H}(\pi_\theta(\cdot|s)) = -\mathbb{E}_{a_\theta^0 \sim q_\theta^0}\left[\log q_\theta^L(a_\theta^L|s)\right] \approx -\mathbb{E}_{a_\theta^0 \sim q_\theta^0}\left[\log q_\theta^0(a^0|s) - \epsilon \sum_{l=0}^{L-1} \text{Tr}\left(\nabla_{a_\theta^l} h(a_\theta^l, s)\right)\right]. \tag{11}$$

In the following, we drop the dependency of the action on θ for simplicity of the notation.

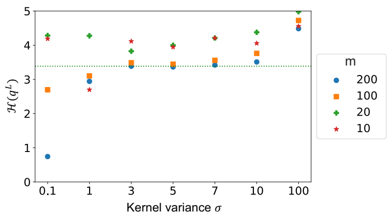
3.3 Invertible Policies

Next, we study the invertibility of three popular EBM samplers: SVGD, SGLD, and HMC as well as the efficiency of computing the trace, *i.e.*, $\text{Tr}(\nabla_{a^l} h(a^l, s))$ in Eq (10) for the ones that are invertible.

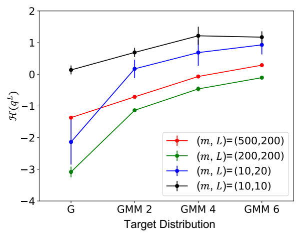
Proposition 3.2 (SVGD invertibility)**.** *Given the SVGD learning rate ϵ and RBF kernel $k(\cdot, \cdot)$ with variance σ, if $\epsilon \ll \sigma$, the update rule of SVGD dynamics defined in Eq (1) is invertible.*



(a) Recovering the GT entropy



(b) Effect of σ on $\mathcal{H}(q^L)$



(c) Effect of m and L on $\mathcal{H}(q^L)$

Figure 4: Entropy evaluation results.

Proof Sketch: We use the explicit function theorem to show that the Jacobian $\nabla_a F(a, s)$ of the update rule $F(a, s)$ is diagonally dominated and hence invertible. This yields invertibility of $F(a, s)$. See detailed proof in Appendix G.3.

Theorem 3.3. *The closed-form estimate of* $\log q^L(a^L|s)$ *for the SVGD based sampler with an RBF kernel* $k(\cdot,\cdot)$ *is*

$$\log q^L(a^L|s) \approx \log q^0(a^0|s) + \frac{\epsilon}{m\sigma^2} \sum_{l=0}^{L-1} \sum_{j=1, a^l \neq a_j^l}^{m} k(a_j^l, a^l)\Big((a^l - a_j^l)^\top \nabla_{a_j^l} Q(s, a_j^l) + \frac{\alpha}{\sigma^2}\|a^l - a_j^l\|^2 - d\alpha\Big).$$

Here, $(\cdot)^\top$ denotes the transpose of a matrix/vector. Note that the entropy does not depend on any matrix computation, but only on vector dot products and first-order vector derivatives. The proof is in Appendix H.1. Intuitively, the derived likelihood is proportional to (1) the concavity of the curvature of the Q-landscape, captured by a weighted average of the neighboring particles' Q-value gradients and (2) pairwise-distances between the neighboring particles ($\sim \|a_i^l - a_j^l\|^2 \cdot \exp(\|a_i^l - a_j^l\|^2)$), *i.e.*, the larger the distance the higher is the entropy. We elaborate on the connection between this formula and non-parametric entropy estimators in Appendix B.

Proposition 3.4 (SGLD, HMC)**.** *The SGLD and HMC updates are not invertible w.r.t.* a.

Proof Sketch: SGLD is stochastic (noise term) and thus not injective. HMC is only invertible if conditioned on the velocity v. Detailed proofs are in Appendices G.1-G.2.
From the above theoretic analysis, we can see that SGLD update is not invertible and hence is not suitable as a sampler for S^{2}AC. While the HMC update is invertible, its derived closed-form entropy involves calculating Hessian and hence computationally more expensive. Due to these considerations, we choose to use SVGD with an RBF kernel as the underlying sampler of S^{2}AC.

4 Results

We first evaluate the correctness of our proposed closed-form entropy formula. Then we present the results of different RL algorithms on multigoal and MuJoCo environments.

4.1 Entropy Evaluation

This experiment tests the correctness of our entropy formula. We compare the estimated entropy for distributions (with known ground truth entropy or log-likelihoods) using different samplers and study the sensitivity of the formula to different samplers' parameters. **(1) Recovering the ground truth entropy.** In Figure 4a, we plot samples (black dots) obtained by SVGD, SGLD, DLD and HMC at convergence to a Gaussian with ground truth entropy $\mathcal{H}(p) = 3.41$, starting from the same initial distribution (leftmost sub-figure). We also report the entropy values computed via Eq.(11). Unlike SGLD, DLD, and HMC, SVGD recovers the ground truth entropy. This empirically supports Proposition 3.4 that SGLD, DLD, and HMC are not invertible. **(2) Effect of the kernel variance.** Figure 4b shows the effect of different SVGD kernel variances σ, where we use the same initial Gaussian from Figure 4a. We also visualize the particle distributions after L SVGD steps for the different configurations in Figure 9 of Appendix I. We can see that when the kernel variance is too small (*e.g.*, $\sigma = 0.1$), the invertibility is violated, and thus the estimated entropy is wrong even at convergence. On the other extreme when the kernel variance is too large (*e.g.*, $\sigma = 100$), *i.e.*, when the particles are too scattered initially, the particles do not converge to the target Gaussian due to noisy gradients in the first term of Eq.(1). The best configurations hence lie somewhere in between (*e.g.*, $\sigma \in \{3, 5, 7\}$). **(3) Effect of SVGD steps and particles.** Figure 4c and Figure 10b (Appendix. I) show the behavior of our entropy formula under different configurations of the number of SVGD steps and particles, on two settings: (i) GMM M with an increasing number of components M, and (ii) distributions with increasing ground truth entropy values, *i.e.*, Gaussians with increasing variances σ.

Results show that our entropy consistently grows with an increasing M (Figure 4c) and increasing σ (Figure 10b), even when a small number of SVGD steps and particles is used (*e.g.*, $L = 10, m = 10$).

4.2 Multi-goal Experiments

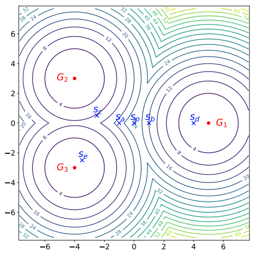


Figure 5: Multigoal env.

To check if S^{2}AC learns a better solution to the max-entropy objective (Eq (2)), we design a new multi-goal environment as shown in Figure 5. The agent is a 2D point mass at the origin trying to reach one of the goals (in red). Q-landscapes are depicted by level curves. Actions are bounded in $[-1, 1]$ along both axes. Critical states for the analysis are marked with blue crosses. It is built on the multi-goal environment in Haarnoja et al. (2017) with modifications such that all the goals have (i) the same maximum expected future reward (positive) but (ii) different maximum expected future entropy. This is achieved by asymmetrically placing the goals (two goals on the left side and one on the right, leading to a higher expected future entropy on the left side) while assigning the same final rewards to all the goals. The problem setup and hyperparameters are detailed in Appendix J. **(1) Multi-modality.** Figure 6 visualizes trajectories (blue lines) collected from 20 episodes of S^{2}AC(ϕ, θ), S^{2}AC(ϕ), SAC, SQL and SAC-NF (SAC with a normalizing flow policy, Mazoure et al. (2020)) agents (rows) at test time for increasing entropy weights α (columns). S^{2}AC and SQL consistently cover all the modes for all α values, while this is only achieved by SAC and SAC-NF for large α values. Note that, in the case of SAC, this comes at the expense of accuracy. Although normalizing flows are expressive enough in theory, they are known to quickly collapse to local optima in practice Kobyzev et al. (2020). The dispersion term in S^{2}AC encodes an inductive bias to mitigate this issue. **(2) Maximizing the expected future entropy.** We also see that with increasing α, more S^{2}AC and SAC-NF trajectories converge to the left goals (G_2/G_3). This shows both models learn to maximize the expected future entropy. This is not the case for SQL whose trajectory distribution remains uniform across the goals. SAC results do not show a consistent trend. This validates the hypothesis that the entropy term in SAC only helps exploration but does not lead to maximizing future entropy. The quantified distribution over reached goals is in Figure 12 of Appendix J. **(3) Robustness/adaptability.** To assess the robustness of the learned policies, we place an obstacle (red bar in Figure 7) on the path to G_2. We show the test time trajectories of 20 episodes using S^{2}AC, SAC, SQL and SAC-NF agents trained with different α's. We observe that, for S^{2}AC and SAC-NF, with increasing α, more trajectories reach the goal after hitting the obstacles. This is not the case for SAC, where many trajectories hit the obstacle without reaching the goal. SQL does not manage to escape the barrier even with higher α. Additional results on the **(4) effect of parameterization of** q^0, and the **(5) entropy's effect on the learned Q-landscapes** are respectively reported in Figure 11 and Figure 14 of Appendix J.

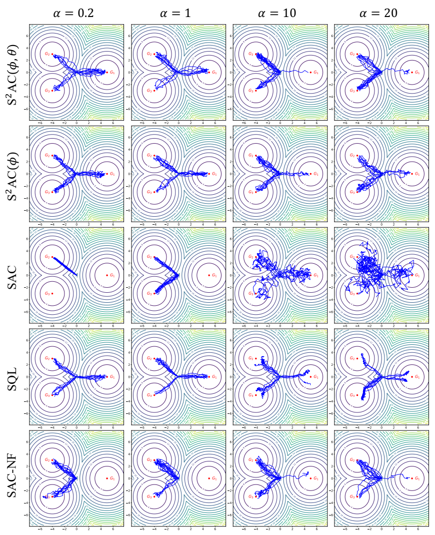


Figure 6: S^{2}AC and SAC-NF learn to maximize the expected future entropy (biased towards G_2/G_3) while SAC and SQL do not. S^{2}AC consistently recovers all modes, while SAC-NF with smaller α's does not, indicating its instability.

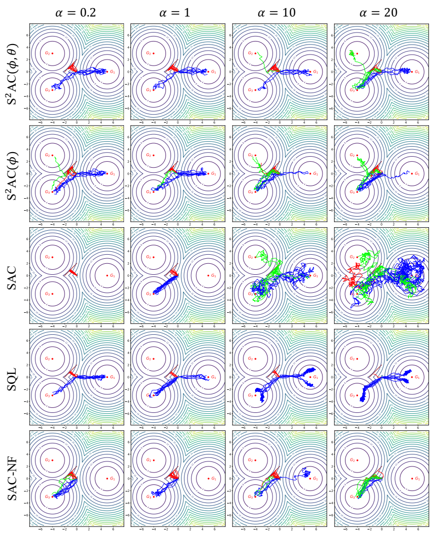


Figure 7: S^{2}AC and SAC-NF are more robust to perturbations. Obstacle O is placed diagonally at $[-1, 1]$. Trajectories that did and did not reach the goal after hitting O are in green and red, respectively.

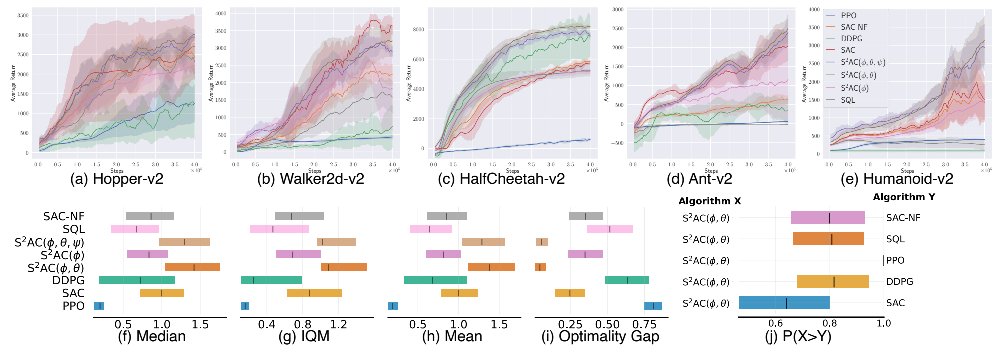


Figure 8: **(a)-(e)**: Performance curves on the MuJoCo benchmark (training). S^{2}AC outperforms SQL and SAC-NF on all environments and SAC on 4 out of 5 environments. **(f)-(i)**: Comparison of Median, IQM, Mean, and Optimality Gap between S^{2}AC and baseline algorithms. **(j)**: The probabilities of S^{2}AC outperforming baseline algorithms.

4.3 MuJoCo Experiments

We evaluate S^{2}AC on five environments from MuJoCo (Brockman et al., 2016): Hopper-v2, Walker2d-v2, HalfCheetah-v2, Ant-v2, and Humanoid-v2. As baselines, we use (1) DDPG (Gu et al., 2017), (2) PPO (Schulman et al., 2015), (3) SQL (Haarnoja et al., 2017), (4) SAC-NF (Mazoure et al., 2020), and (5) SAC (Haarnoja et al., 2018a). Hyperparameters are in Appendix K.

(1) Performance and sample efficiency. We train five different instances of each algorithm with different random seeds, with each performing 100 evaluation rollouts every 1000 environment steps. Performance results are in Figure 8(a)-(e). The solid curves correspond to the mean returns over the five trials and the shaded region represents the minimum and maximum. S^{2}AC(ϕ, θ) is consistently better than SQL and SAC-NF across all the environments and has superior performance than SAC in four out of five environments. Results also show that the initial parameterization was key to ensuring the scalability (S^{2}AC(ϕ) has poor performance compared to S^{2}AC(ϕ, θ)). Figure 8(f)-(j) demonstrate the statistical significance of these gains by leveraging statistics from the rliable library (Agarwal et al., 2021) which we detail in Appendix K.

(2) Run-time. We report the run-time of action selection of SAC, SQL, and S^{2}AC algorithms in Table 1. S^{2}AC(ϕ, θ) run-time increases linearly with the action space. To improve the scalability, we train an amortized version that we deploy at test-time, following (Haarnoja et al., 2017). Specifically, we train a feed-forward deepnet $f_\psi(s, z)$ to mimic the SVGD dynamics during testing, where z is a random vector that allows mapping the same state to different particles. Note that we cannot use $f_\psi(s, z)$ during training as we need to estimate the entropy in Eq (11), which depends on the unrolled SVGD dynamics (details in Appendix K). The amortized version S^{2}AC(ϕ, θ, ψ) has a similar run-time to SAC and SQL with a slight tradeoff in performance (Figure 8).

	Hopper	Walker2d	HalfCheetah	Ant
Action dim	3	6	6	8
State dim	11	17	17	111
SAC	0.723	0.714	0.731	0.708
SQL	0.839	0.828	0.815	0.836
S^{2}AC(ϕ, θ)	3.267	4.622	4.583	5.917
S^{2}AC(ϕ, θ, ψ)	0.850	0.817	0.830	0.837

Table 1: Action selection run-time on MuJoCo.

5 Related Work

MaxEnt RL (Todorov, 2006; Ziebart, 2010; Rawlik et al., 2012) aims to learn a policy that gets high rewards while acting as randomly as possible. To achieve this, it maximizes the sum of expected future reward and expected future entropy. It is different from entropy regularization (Schulman et al., 2015; O'Donoghue et al., 2016; Schulman et al., 2017) which maximizes entropy at the current time step. It is also different from multi-modal RL approaches (Tang & Agrawal, 2018) which recover different modes with equal frequencies without considering their future entropy. MaxEnt RL has been broadly incorporated in various RL domains, including inverse RL (Ziebart et al., 2008; Finn et al., 2016), stochastic control (Rawlik et al., 2012; Toussaint, 2009), guided policy search (Levine & Koltun, 2013), and off-policy learning (Haarnoja et al., 2018a;b). MaxEnt RL is shown to maximize a lower bound of the robust RL objective (Eysenbach & Levine, 2022) and is hence less sensitive to perturbations in state and reward functions. From the variational inference lens, MaxEnt RL aims to find the policy distribution that minimizes the *KL*-divergence to an EBM over Q-function. The desired family of variational distributions is (1) expressive enough to capture the intricacies of the Q-value landscape (*e.g.*, multimodality) and (2) has a tractable entropy estimate. These two requirements are hard to satisfy. SAC (Haarnoja et al., 2018a) uses a Gaussian policy. Despite having a tractable entropy, it fails to capture arbitrary Q-value landscapes. SAC-GMM (Haarnoja, 2018) extends SAC by modeling the policy as a Gaussian Mixture Model, but it requires an impractical grid search over the number of components. Other extensions include IAPO (Marino et al., 2021) which also models the policy as a uni-modal Gaussian but learns a collection of parameter estimates (mean, variance) through different initializations. While this yields multi-modality, it does not optimize a MaxEnt objective. SSPG (Cetin & Celiktutan, 2022) and SAC-NF (Mazoure et al., 2020) respectively improve the policy expressivity by modeling the policy as a Markov chain with Gaussian transition probabilities and as a normalizing flow. Due to training instability, the reported multi-goal experiments in (Cetin & Celiktutan, 2022) show that, though both models capture multimodality, they fail to maximize the expected future entropy in positive reward setups. SQL (Haarnoja et al., 2017), on the other hand, bypasses the explicit entropy computation altogether via a soft version of value iteration. It then trains an amortized SVGD (Wang & Liu, 2016) sampler from the EBM over the learned Q-values. However, estimating soft value functions requires approximating integrals via importance sampling which is known to have high variance and poor scalability. We propose a new family of variational distributions induced by a parameterized SVGD sampler from the EBM over Q-values. Our policy is expressive and captures multi-modal distributions while being characterized by a tractable entropy estimate.

EBMs (LeCun et al., 2006; Wu et al., 2018) are represented as Gibbs densities $p(x) = \exp E(x)/Z$, where $E(x) \in \mathbb{R}$ is an energy function describing inter-variable dependencies and $Z = \int \exp E(x)$ is the partition function. Despite their expressiveness, EBMs are not tractable as the partition function requires integrating over an exponential number of configurations. Markov Chain Monte Carlo (MCMC) methods (Van Ravenzwaaij et al., 2018) (*e.g.*, HMC (Hoffman & Gelman, 2014), SGLD (Welling & Teh, 2011)) are frequently used to approximate the partition function via sampling. There have been recent efforts to parameterize these samplers via deepnets (Levy et al., 2017; Gong et al., 2018; Feng et al., 2017) to improve scalability. Similarly to these methods, we propose a parameterized variant of SVGD (Liu & Wang, 2016) as an EBM sampler to enable scalability to high-dimensional action spaces. Beyond sampling, we derive a closed-form expression of the sampling distribution as an estimate of the EBM. This yields a tractable estimate of the entropy. This is opposed to previous methods for estimating EBM entropy which mostly rely on heuristic approximation, lower bounds Dai et al. (2017; 2019a), or neural estimators of mutual information (Kumar et al., 2019). The idea of approximating the entropy of EBMs via MCMC sampling by leveraging the change of variable formula was first proposed in Dai et al. (2019b). The authors apply the formula to HMC and LD, which, as we show previously, violate the invertibility assumption. To go around this, they augment the EBM family with the noise or velocity variable for LD and HMC respectively. But the derived log-likelihood of the sampling distribution turns out to be –counter-intuitively– independent of the sampler's dynamics and equal to the initial distribution, which is then parameterized using a flow model (details in Appendix B.2). We show that SVGD is invertible, and hence we sample from the original EBM, so that our derived entropy is more intuitive as it depends on the SVGD dynamics.

SVGD-augmented RL (Liu & Wang, 2016) has been explored under other RL contexts. Liu et al. (2017) use SVGD to learn a distribution over policy parameters. While this leads to learning diverse policies, it is fundamentally different from our approach as we are interested in learning a single multi-modal policy with a closed-form entropy formula. Castanet et al. (2023); Chen et al. (2021) use SVGD to sample from multimodal distributions over goals/tasks. We go beyond sampling and use SVGD to derive a closed-form entropy formula of an expressive variational distribution.

6 Conclusion

We propose S^{2}AC, an actor-critic algorithm that yields a more optimal solution to the MaxEnt RL objective than previously proposed approaches. S^{2}AC achieves this by leveraging a new family of variational distributions characterized by SVGD dynamics. The proposed distribution has high expressivity, *i.e.*, it is flexible enough to capture multimodal policies in high dimensional spaces, and a tractable entropy estimate. Empirical results show that S^{2}AC learns expressive and robust policies while having superior performance than other MaxEnt RL algorithms. For future work, we plan to study the application of the proposed variational distribution to other domains and develop benchmarks to evaluate the robustness of RL agents.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS*, 2021.

I. Ahmad and Pi-Erh Lin. A nonparametric estimation of the entropy for absolutely continuous distributions (corresp.). *IEEE Trans. Inf. Theory*, 1976.

Jan Beirlant and M.C.A van Zuijlen. The empirical distribution function and strong laws for functions of order statistics of uniform spacings. *J. Multivar. Anal.*, 1985.

Jan Beirlant, Edward J Dudewicz, László Györfi, Edward C Van der Meulen, et al. Nonparametric entropy estimation: An overview. *IJSRMSS*, 1997.

Laura T Bernhofen, Edward J Dudewicz, Janos Levendovszky, and Edward C van der Meulen. Ranking of the best random number generators via entropy-uniformity theory. *AJMMS*, 1996.

Peter J. Bickel and Leo Breiman. Sums of Functions of Nearest Neighbor Distances, Moment Bounds, Limit Theorems and a Goodness of Fit Test. *Ann. Probab.*, 1983.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 2017.

Nicolas Castanet, Olivier Sigaud, et al. Stein variational goal generation for adaptive exploration in multi-goal reinforcement learning. 2023.

Edoardo Cetin and Oya Celiktutan. Policy gradient with serial markov chain reasoning. *NeurIPS*, 2022.

Jiayu Chen, Yuanxin Zhang, Yuanfan Xu, Huimin Ma, Huazhong Yang, Jiaming Song, Yu Wang, and Yi Wu. Variational automatic curriculum learning for sparse-reward cooperative multi-agent problems. *NeurIPS*, 2021.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Noel Cressie. Power results for tests based on high-order gaps. *Biometrika*, 1978.

Bo Dai, Hanjun Dai, Arthur Gretton, Le Song, Dale Schuurmans, and Niao He. Kernel exponential family estimation via doubly dual embedding. In *AISTAT*, 2019a.

Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. *NeurIPS*, 2019b.

Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017.

Jay L Devore, Kenneth N Berk, Matthew A Carlton, et al. *Modern mathematical statistics with applications*. Springer, 2012.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

Edward J Dudewicz and Edward C Van Der Meulen. Entropy-based tests of uniformity. *JASA*, 1981.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. In *ICLR*, 2022.

Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. *arXiv preprint arXiv:1707.06626*, 2017.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, 2016.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.

Xun Gao and Lu-Ming Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature communications*, 2017.

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-learning for stochastic gradient mcmc. *arXiv preprint arXiv:1806.04522*, 2018.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *ICLR*, 2017.

László Györfi and Edward C. van der Meulen. Density-free convergence properties of various estimators of entropy. *CSDA*, 1987.

Tuomas Haarnoja. *Acquiring diverse robot skills via maximum entropy deep reinforcement learning (Ph.D. thesis)*. University of California, Berkeley, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Peter Hall. On powerful distributional tests based on sample spacings. *JMVA*, 1986.

Hideitsu Hino and Noboru Murata. A conditional entropy minimization criterion for dimensionality reduction and multiple kernel learning. *Neural Comput.*, 2010.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *JMLR*, 2014.

Aleksandr Vasil'evich Ivanov and MN Rozhkova. On properties of the statistical estimate of the entropy of a random vector with a probability density. *Problemy Peredachi Informatsii*, 1981.

Harry Joe. Estimation of entropy and other functionals of a multivariate density. *Ann. Inst. Stat. Math.*, 1989.

Hilbert J Kappen. Path integrals and symmetry breaking for optimal control theory. *JSTAT*, 2005.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *PAMI*, 2020.

Lyudmyla F Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 1987.

Rithesh Kumar, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Engan: Latent space mcmc and maximum entropy generators for energy-based models. *ICLR*, 2019.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. *NeurIPS*, 2007.

Erik G Learned-Miller and John W Fisher III. Ica using spacings estimates of entropy. *JMLR*, 2003.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 2006.

Kyowoon Lee, Sol-A Kim, Jaesik Choi, and Seong-Whan Lee. Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In *ICML*, 2018.

Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013.

Daniel Levy, Matthew D Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. *arXiv preprint arXiv:1711.09268*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Haozhe Liu, Bing Li, Haoqian Wu, Hanbang Liang, Yawen Huang, Yuexiang Li, Bernard Ghanem, and Yefeng Zheng. Combating mode collapse in gans via manifold entropy estimation. *AAAI*, 2022.

Qiang Liu. Stein variational gradient descent as gradient flow. *NeurIPS*, 2017.

Qiang Liu and Dilin Wang. Stein variational gradient descent: a general purpose bayesian inference algorithm. In *NeurIPS*, 2016.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *UAI*, 2017.

Jan R Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

Shie Mannor, Dori Peleg, and Reuven Rubinstein. The cross entropy method for classification. In *ICML*, 2005.

Joseph Marino, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue. Iterative amortized policy optimization. *NeurIPS*, 2021.

Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *CoRL*, 2020.

Roger G Melko, Giuseppe Carleo, Juan Carrasquilla, and J Ignacio Cirac. Restricted boltzmann machines in quantum physics. *Nature Physics*, 2019.

Safa Messaoud. *Toward More Scalable Structured Models*. PhD thesis, University of Illinois Urbana-Champaign, 2021.

Safa Messaoud, David Forsyth, and Alexander G Schwing. Structural consistency and controllability for diverse colorization. In *ECCV*, pp. 596–612, 2018.

Safa Messaoud, Maghav Kumar, and Alexander G Schwing. Can we learn heuristics for graphical model inference using reinforcement learning? In *CVPR Workshops*, pp. 766–767, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2011.

Brendan O'Donoghue, Remi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *CVPR*, pp. 11814–11824, 2021.

Liam Paninski. Estimation of entropy and mutual information. *Neural Comput.*, 2003.

Fernando Pérez-Cruz. Estimation of information theoretic measures for continuous random variables. In *NeurIPS*, 2008.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*. PMLR, 2015.

Édgar Roldán, Jérémie Barral, Pascal Martin, Juan MR Parrondo, and Frank Jülicher. Quantifying entropy production in active fluctuations of the hair-cell bundle from time irreversibility and uncertainty relations. *New J. Phys.*, 2021.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag, 2004.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, 2007.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE*, 2001.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *NeurIPS*, 1999.

Yunhao Tang and Shipra Agrawal. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint arXiv:1809.10326*, 2018.

F.P. Tarasenko. On the evaluation of an unknown probability density function, the direct estimation of the entropy from independent observations of a continuous random variable, and the distribution-free entropy test of goodness-of-fit. *Proceedings of the IEEE*, 1968.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *JMLR*, 2010.

Emanuel Todorov. Linearly-solvable markov decision problems. In *NeurIPS*, 2006.

Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 2018.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *ICML*, 2009.

Don Van Ravenzwaaij, Pete Cassey, and Scott D Brown. A simple introduction to markov chain monte–carlo sampling. *Psychon. Bull. Rev.*, 2018.

Oldrich Vasicek. *A Test for Normality Based on Sample Entropy*. JSTOR, 2015.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. In *NeurIPS*, 2020.

Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

Ying Nian Wu, Jianwen Xie, Yang Lu, and Song-Chun Zhu. Sparse and deep generalizations of the frame model. *Annals of Mathematical Sciences and Applications*, 2018.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative voxelnet: learning energy-based models for 3d shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5), 2020.

Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Learning cycle-consistent cooperative networks via alternating mcmc teaching for unsupervised cross-domain translation. In *AAAI*, volume 35, pp. 10430–10440, 2021.

Yang Zhao, Jianwen Xie, and Ping Li. Learning energy-based generative models via coarse-to-fine expanding and sampling. In *ICLR*, 2021.

Zilong Zheng, Jianwen Xie, and Ping Li. Patchwise generative convnet: Training energy-based models from a single natural image for internal learning. In *CVPR*, pp. 2961–2970, June 2021.

Mo Zhou and Jianfeng Lu. Single timescale actor-critic method to solve the linear quadratic regulator with convergence guarantees. *JMLR*, 2023.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

Supplementary Material

A Summary

In this paper, we propose a new variational distribution that we use to model the actor in the context of actor-critic MaxEntr RL algorithms. Our distribution is induced by an SVGD sampler with a parametrized initial distribution (isotropic Gaussian). It enables fitting multi-modal distribution (*e.g.*, EBM) and is characterized by a closed-form entropy estimate. Hence, it addresses the major bottleneck in classical MaxEntr RL algorithms. Our derivation is based on the unique invertibility property of the SVGD sampler, which is not satisfied for other popular samplers (*e.g.*, SGLD, HMC). The key to achieving scalability was to learn the initial Gaussian distribution such that it contours the high-density region of the target distribution, by limiting particles' updates to be always within few standard deviations of the mean of this Gaussian. This resulted in better and faster exploration of the relevant regions of the target distribution. Our proposed approach S^{2}AC is summarized in Algorithm 1.The rest of the supplementary is organized as follows:

- Appendix B provides additional related work on the entropy estimation.
- Appendix C introduces additional preliminaries on EBM samplers, the change of variable formula and the Jacobi formula.
- Appendix D provides the derivation of the optimal policy for the MaxEntr RL objective.
- Appendix E provides the derivation of the actor objective.
- Appendices F-H provide proofs for theorems related to (1) a generic closed-form expression of log-likelihood of inverible samplers, (2) discussion of samplers invertibility and (3) closed-form likelihood derivation for SVGD.
- Appendices I-K provide additional results for the (1) entropy evaluation, (2) multigoal environment, and (3) MuJoCo environments.

Algorithm 1 Stein Soft Actor Critic (S^{2}AC)

```
1:  Initialize parameters $\phi$, $\theta$, hyperparameter $\alpha$, and replay buffer $\mathcal{D} \leftarrow \emptyset$
2:  for each iteration do
3:      for each environment step $t$ do
4:          Sample action particles $\{a\}$ from $\pi_\theta(\cdot|s_t)$
5:          Select $a_t \in \{a\}$ using exploration strategy
6:          Sample next state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
7:          Update replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r_t, s_{t+1})$
8:      for each gradient step do
9:          Critic update:
10:             Sample particles $\{a\}$ from an EMB sampler $\pi_\theta(\cdot|s_{t+1})$
11:             Compute entropy $\mathcal{H}(\pi_\theta(\cdot|s_{t+1}))$ using Eq.(11)
12:             Update $\phi$ using Eq.(8)
13:         Actor update:
14:             Update $\theta$ using Eq.(9)
```

B Additional Related-work

B.1 Entropy

The differential entropy Cover (1999); Shannon (2001) of a p-dimensional random variable X with a probability density function $p(x)$ is defined by: $H(p) = -\int p(x) \ln p(x) dx$. The differential entropy plays a central role in information and communication theory, statistics Tarasenko (1968), signal processing Vasicek (2015); Learned-Miller & III (2003), machine learning and pattern recognition Mannor et al. (2005); Rubinstein & Kroese (2004); Hino & Murata (2010); Liu et al. (2022); Wulfmeier et al. (2015). For example, Max-Entropy RL Wulfmeier et al. (2015); Haarnoja et al. (2017; 2018a) methods augment the expected reward objective with an entropy maximization term which results in learning multi-modal policies and more robustness. Recently, Liu *et. al* Liu et al. (2022) propose maximizing the entropy of the discriminator distribution to combat mode collapse. In statistical mechanics entropy appears as the negative of the rate function to quantify the fluctuations around thermodynamic equilibrium Roldán et al. (2021). Estimating the differential entropy for expressive distributions is a challenging problem as it requires computing a closed-form expression of the probability density function. Several non-parametric approaches Beirlant et al. (1997); Györfi & van der Meulen (1987); Paninski (2003); Pérez-Cruz (2008) based on approximating the entropy using samples $\mathcal{D} = \{x_i\}_{i=1}^{|\mathcal{D}|}$ from $p(x)$, have been proposed in the literature. These methods can be classified into (1) plug-in estimates Ahmad & Lin (1976); Ivanov & Rozhkova (1981); Joe (1989) which approximate $p(x)$ via a kernel density estimate, (2) samples spacing Beirlant & van Zuijlen (1985); Cressie (1978); Dudewicz & Van Der Meulen (1981); Hall (1986) and (3) nearest-neighbor distances based estimates Bernhofen et al. (1996); Bickel & Breiman (1983); Kozachenko & Leonenko (1987) which express the entropy in terms of pairwise distances between the samples (larger distances imply higher entropy). Next, we review the work on entropy estimation of Energy-Based-Models (EBMs).

B.2 Entropy of EBMs

In this work, we are interested in computing entropy estimates for the class of EBMs LeCun et al. (2006) represented as Gibbs densities $p(x) = \frac{\exp E(x)}{Z}$, where $E(x) \in \mathbb{R}$ is an energy function describing inter-variable dependencies and $Z = \int \exp E(x)$ is an intractable partition function. EBMs provide a unified framework for many probabilistic and non-probabilistic approaches, particularly for learning and inference in structured models and are widely used in computer science (*e.g.*, semantic segmentation, colorization, image generation, collaborative filtering) Salakhutdinov et al. (2007); Messaoud et al. (2018); Zhao et al. (2021); Gao et al. (2020); Xie et al. (2020); Zheng et al. (2021); Carleo & Troyer (2017); Messaoud et al. (2020); Xie et al. (2021); Pang et al. (2021); Messaoud (2021) and physics Carleo & Troyer (2017); Gao & Duan (2017); Torlai et al. (2018); Melko et al. (2019) (*e.g.*, to model the wavefunctions of quantum systems). To estimate the entropy of EBMs, previous methods mostly rely on heuristic approximation, lower bounds Dai et al. (2017; 2019a), or neural estimators of mutual information to approximate the entropy Kumar et al. (2019). The idea of approximating the entropy of EBMs via the one from an MCMC sampler by leveraging the change of variable formula was first proposed by Dai et al. (2019b). Specifically, the authors apply the formula to HMC and LD which, as we show in Appendix. G, violate the invertibility assumption. To go around this, the authors propose augmenting the EBM family with the noise or velocity variable for, respectively, LD and HMC, *i.e.*, sampling from $p(x)$ is replaced with sampling from $p(x, v)$ or $p(x, \xi)$. The authors assume that the sampler update rule is invertible with respect to the augmented samples (x, v) and (x, ξ). However, computing the determinant of the update rule with respect to the augmented variable is always equal to 1 in this case. Hence, the resulting log-likelihood of the sampling distribution is, counter-intuitively, independent of the sampler's dynamics and equal to the initial distribution, *i.e.*, $\log q^L(a^L) = \log q^0(a^0)$, which the author model using a flow model. Differently, we show that SVGD is invertible, our entropy depends on the dynamics of SVGD, we still sample from the original EBM $p(x)$ and our initial distribution is a simple Gaussian. Similarly to the non-parametrized entropy estimates described above, our formula leverages pairwise distances between the neighboring samples. Differently, our formula is also based on the curvature of the energy function $E(x)$ (measured by a weighted average of neighboring particle gradients $\nabla_x E(x)$). Hence maximizing our derived entropy results in the intuitive effect of learning smoother energy landscapes.

C Additional Preliminaries

In the following, we review (1) additional samplers for EBMs, (2) the change of variable formula and (3) the corollary of the Jacobi's formula.

C.1 Additional Samplers for EBMs

SGLD (Welling & Teh, 2011) is a popular Markov chain Monte Carlo (MCMC) method for sampling from a distribution. It initializes a sample a^0 from a random distribution, and then in each step $l+1$ it adds the gradient of the current proposal distribution $p(a)$ to the previous sample a^l, together with a Brownian motion $\xi \sim N(0, I)$. We denote the step size as ϵ. The iterative update for SGLD is:

$$a^{l+1} = a^l + \epsilon \nabla_{a^l} \log p(a^l) + \sqrt{2\epsilon}\xi. \tag{12}$$

DLD are equivalent to SGLD without the noise term, *i.e.*,

$$a^{l+1} = a^l + \epsilon \nabla_{a^l} \log p(a^l). \tag{13}$$

HMC is another popular variant of MCMC samplers. The most commonly used discretized Hamilton's equations are the leapfrog method (Neal et al., 2011). The three (half) steps of leapfrog updates in HMC are:

$$\begin{aligned} v^{l+1/2} &= v^l + (\epsilon/2)\nabla_a \log p(a^{l+1}) \\ a^{l+1} &= a^l + \epsilon v^{l+1/2} \\ v^{l+1} &= a^{l+1} + (\epsilon/2)\nabla_a \log p(a^{l+1}) \end{aligned} \tag{14}$$

Here v^l is interpreted the velocity at iteration l (assuming unit mass) and a^l is the "location" of a sample in a distribution.

C.2 Change of Variable Formula

We first introduce the concept of an invertibile function.

Definition C.1 (Invertibile transformation)**.** Transformation $F : Z \to X$ is invertible iff $F(\cdot)$ is bijective, *i.e.*, simultaneously injective and surjective: (i) $F(\cdot)$ is injective iff for any $z, z' \in Z$, $F(z) = F(z') \Rightarrow z = z'$; (ii) $F(\cdot)$ is surjective iff for every $x \in X$, there exists some $z \in Z$ such that $F(z) = x$.

According to change of variable formula, the following holds when $F : Z \to X$ is an invertible function:

$$p_X(x) = p_Z(z)\Big|\det\frac{\partial F^{-1}(x)}{\partial x}\Big| = p_Z(z)\Big|\det\frac{\partial F(z)}{\partial z}\Big|^{-1}$$

C.3 A Corollary of Jacobi's Formula

An important corollary of Jacobi's Formula (Magnus & Neudecker, 2019) states that, given an invertible matrix A, the following equality holds:

$$\log(\det A) = \mathrm{Tr}\,(\log A) = \mathrm{Tr}\,\Big(\sum_{k=1}^{\infty}(-1)^{k+1}\frac{(A-I)^k}{k}\Big).$$

The second equation is obtained by taking the power series of $\log A$. Hence, under the assumption $\|A - I\|_\infty \ll 1$, we obtain:

$$\log(\det A) \approx \mathrm{tr}(A - I).$$

D Derivation of the MaxEnt RL optimal policy

In this section, we prove that the solution π^* of the MaxEnt RL objective

$$\max_\pi J(\pi) \equiv \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\Big[\gamma^t\Big(r(a_t,s_t) - \alpha\log\pi(\cdot|s_t)\Big)\Big] \tag{15}$$

is $\pi^* = \frac{\exp(\frac{1}{\alpha}Q(s,a))}{Z}$. Here, $Q(s,a)$ is the soft Q-function defined as

$$\begin{aligned} Q(s,a) &= \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\Big[\sum_t \gamma^t\Big(r(a_t,s_t) - \alpha\log p(\pi(\cdot|s_t))\Big)|s_0=s,a_0=a\Big] \\ &= r(a,s) + \alpha\mathcal{H}(\pi(\cdot|s)) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}\Big[Q(s',a')\Big]. \end{aligned} \tag{16}$$

Consequently, we deduce that π^* is also the solution of the expected KL divergence:

$$\pi^* = \arg\min_\pi \sum_t \mathbb{E}_{s_t\sim\rho_\pi}\Big[D_{KL}\big(\pi(\cdot|s_t)\|\exp(Q(s_t,\cdot)/\alpha)/Z\big)\Big]. \tag{17}$$

Proof. We express the MaxEnt loss as a function of $Q(s,a)$, *i.e.*, $J(\pi) = \mathbb{E}_{(s,a)\sim\rho_\pi}\Big[Q(s,a)\Big]$. To find $\pi^* = \arg\max_\pi J(\pi)$ under the constraint $\int_a \pi(a|s)da = 1$, we evaluate the Lagrangian (with $\lambda\in\mathbb{R}$ being the Lagrange multiplier):

$$\mathcal{L}(\pi,\lambda) = \mathbb{E}_{(s,a)\sim\rho_\pi}\Big[Q(s,a)\Big] + \lambda\Big(\int_a \pi(a|s)da - 1\Big), \tag{18}$$

and compute $\frac{\partial\mathcal{L}(\pi,\lambda)}{\partial\pi(a|s)}$:

$$\begin{aligned} \frac{\partial\mathcal{L}(\pi,\lambda)}{\partial\pi(a|s)} &= \frac{\partial}{\partial\pi(a|s)}\Big(\int_s\int_a \pi(a|s)\rho_\pi(s)Q(s,a)da\,ds + \lambda\Big(\int_a\pi(a|s)da - 1\Big)\Big) \\ &= \frac{\partial}{\partial\pi(a|s)}\Big(\pi(a|s)\rho_\pi(s)\Big(r(a,s) - \alpha\log\pi(a|s) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}\Big[Q(s',a')\Big]\Big)\Big) + \lambda \\ &= \rho_\pi(s)\Big(r(a,s) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}[Q(s',a')]\Big) - \alpha\rho_\pi(s)\frac{\partial}{\partial\pi(a|s)}\Big(\pi(a|s)\log\pi(a|s)\Big) + \lambda \\ &= \rho_\pi(s)\Big(r(a,s) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}[Q(s',a')]\Big) - \alpha\rho_\pi(s)\Big(\log\pi(a|s) + 1\Big) + \lambda. \end{aligned}$$

Setting $\frac{\partial\mathcal{L}(\pi,\lambda)}{\partial\pi(a|s)}$ to 0:

$$\begin{aligned} \frac{\partial\mathcal{L}(\pi,\lambda)}{\partial\pi(a|s)} = 0 \iff & \Big(r(a,s) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}[Q^\pi(s',a')]\Big) - \alpha + \frac{\lambda}{\rho_\pi(s)} = \alpha\log\pi(a|s) \\ \iff & \frac{1}{\alpha}\Big(r(a,s) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}[Q(s',a')]\Big) - 1 + \frac{\lambda}{\alpha\rho_\pi(s)} = \log\pi(a|s) \\ \iff & \pi(a|s) = \frac{\exp\Big(\frac{1}{\alpha}\Big(r(a,s) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}[Q(s',a')]\Big)\Big)}{\exp\Big(1 - \frac{\lambda}{\alpha\rho_\pi(s)}\Big)} \\ \iff & \pi(a|s) = \frac{\exp\Big(\frac{1}{\alpha}\Big(r(a,s) + \mathcal{H}(\pi(\cdot|s)) + \mathbb{E}_{\pi(a'|s)\rho_\pi(s')}[Q(s',a')]\Big)\Big)}{\exp\Big(1 - \frac{\lambda}{\alpha\rho_\pi(s)}\Big)} \\ \iff & \pi(a|s) = \frac{\exp\Big(\frac{1}{\alpha}Q(s,a)\Big)}{\exp\Big(\frac{\mathcal{H}(\pi(\cdot|s))}{\alpha} + 1 - \frac{\lambda}{\alpha\rho_\pi(s)}\Big)} \end{aligned} \tag{19}$$

We choose λ such that $\int_a \pi(a|s)da = 1$, *i.e.*,

$$\int_a \frac{\exp\Big(\frac{1}{\alpha}Q(s,a)\Big)}{\exp\Big(\frac{\mathcal{H}(\pi(\cdot|s))}{\alpha}+1-\frac{\lambda}{\alpha\rho_\pi(s)}\Big)} da = 1 \Longleftrightarrow \lambda = -\alpha\rho_\pi(s)\Big(\log\int_a \exp\Big(\frac{1}{\alpha}Q(s,a)\Big)da - \frac{\mathcal{H}(\pi(\cdot|s))}{\alpha} - 1\Big). \tag{20}$$

Hence, $\pi^*(a|s) = \exp(\frac{1}{\alpha}Q(s,a))/\int_a \exp(\frac{1}{\alpha}Q(s,a))$. A similar proof follows for any state and action pairs. Trivially, π^* is also the global minimum of Eq.(17). □

E Derivation of the actor objective (Eq.(9))

In the following, we prove that the objective

$$\arg\min_\theta \mathbb{E}_{s_t\sim\mathcal{D}}\Big[D_{KL}\Big(\pi_\theta(\cdot|s_t)\Big|\Big|\exp\Big(\frac{1}{\alpha}Q_\phi(s_t,\cdot)\Big)/Z(\phi)\Big)\Big]$$

is equivalent to

$$\arg\max_\theta \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta(a_t|s_t)}\Big[Q_\phi(s_t,a_t)\Big] + \mathbb{E}_{s_t}\Big[\alpha\mathcal{H}(\pi_\theta(a_t|s_t))\Big],$$

with $\mathcal{D}$ being a replay buffer.

Proof.

$$\begin{aligned}
\theta^* &= \arg\min_\theta \mathbb{E}_{s_t\sim\mathcal{D}}\Big[D_{KL}\Big(\pi_\theta(\cdot|s_t)\Big|\Big|\exp\Big(\frac{1}{\alpha}Q_\phi(s_t,\cdot)\Big)/Z(\phi)\Big)\Big]\\
&= \arg\min_\theta \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta(a_t|s_t)}\Big[\log(\pi_\theta(a_t|s_t)) - \Big(\frac{1}{\alpha}Q_\phi(s_t,a_t) - \log Z(\phi)\Big)\Big]\\
&= \arg\min_\theta \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta(a_t|s_t)}\Big[\log(\pi_\theta(a_t|s_t)) - \frac{1}{\alpha}Q_\phi(s_t,a_t)\Big]\\
&= \arg\max_\theta \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta(a_t|s_t)}\Big[-\log(\pi_\theta(a_t|s_t)) + \frac{1}{\alpha}Q_\phi(s_t,a_t)\Big]\\
&= \arg\max_\theta \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta(a_t|s_t)}\Big[\frac{1}{\alpha}Q_\phi(s_t,a_t) + \mathcal{H}(\pi_\theta(a_t|s_t))\Big]\\
&= \arg\max_\theta \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta(a_t|s_t)}\Big[Q_\phi(s_t,a_t)\Big] + \mathbb{E}_{s_t\sim\mathcal{D}}\Big[\alpha\mathcal{H}(\pi_\theta(a_t|s_t))\Big]
\end{aligned}$$

□

F PROOF OF THEOREM 3.1

Theorem. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible transformation of the form $F(a) = a + \epsilon h(a)$. We denote by $q^L(a^L)$ the distribution obtained from repeatedly (L times) applying F to a set of action samples (called "particles") $\{a^0\}$ from an initial distribution $q^0(a^0)$,* i.e.*, $a^L = F \circ F \circ \cdots \circ F(a^0)$. Under the condition $\epsilon ||\nabla_{a_i} h(a_i)||_\infty \ll 1$, the closed-form expression of $\log q^L(a^L)$ is:*

$$\log q^L(a^L) = \log q^0(a^0) - \epsilon \sum_{l=0}^{L-1} \operatorname{Tr}(\nabla_{a^l} h(a^l)). \tag{21}$$

Proof. Based on the change of variable formula (Appendix C.2), when for every iteration $l \in [1, L]$, the transformation $a^l = L(a^{l-1})$ (of the action sampler in our paper) is invertible, we have:

$$q^l(a^l) = q^{l-1}(a^{l-1}) \left| \det \frac{da^l}{da^{l-1}} \right|^{-1}, \forall l \in [1, L].$$

By induction, we derive the probability distribution of sample a^L:

$$q^L(a^L) = q^0(a^0) \prod_{l=1}^{L} \left| \det \frac{da^l}{da^{l-1}} \right|^{-1} = q^0(a^0) \prod_{l=0}^{L-1} \left| \det \left(I + \epsilon \nabla_{a^l} h(a^l) \right) \right|^{-1}$$

By taking the $\log$ for both sides, we obtain:

$$\log q^L(a^L) = \log q^0(a^0) - \sum_{l=0}^{L-1} \log \left| \det \left(I + \epsilon \nabla_{a^l} h(a^l) \right) \right|.$$

Let $A = I + \epsilon \nabla_{a^l} h(a^l)$, under the assumption $\epsilon ||\nabla_{a_i} h(a_i)||_\infty \ll 1$, *i.e.*, $||A - I||_\infty \ll 1$, we apply the corollary of Jacobi's formula (Appendix C.3) and get

$$\begin{aligned}
\log q^L(a^L) &\approx \log q^0(a^0) - \sum_{l=0}^{L-1} \operatorname{Tr} \left((I + \epsilon \nabla_{a^l} h(a^l)) - I) \right) + \mathcal{O}(\epsilon^2 dL) \\
&\approx \log q^0(a^0) - \epsilon \sum_{l=0}^{L-1} \operatorname{Tr} \left(\nabla_{a^l} h(a^l) \right) + \mathcal{O}(\epsilon^2 dL).
\end{aligned}$$

Here, d is the action space dimension. □

G Samplers Invertibility Proofs

We start by state the implicit function theorem which we will be using in the following proofs.

Theorem G.1 (**Implicit function theorem**). *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable on some open set containing a, and suppose $\det\left(Jf(a)\right) = \det\left(\nabla_a f(a)\right) \neq 0$. Then, there is some open set V containing a and an open W containing $f(a)$ such that $f : V \to W$ has a continuous inverse $f^{-1} : W \to V$ which is differentiable $\forall y \in W$.*

G.1 Stochastic Gradient Langevin Dynamics

Proposition (SGLD). *The SGLD update in Eq.*(12) *is not invertible.*

Proof. We show that SGLD are not invertible using two different methods: (1) We show that SGLD is not a bijective transformation, (2) Using the implicit function theorem, we show that the Jacobian of the dynamics is not invertible.

G.1.1 Method1: SGLD is not invertible $\Longleftrightarrow$ SGLD is not a bijection

The update rule $F(\cdot)$ for SGLD and DGLD are given by Eq.(12) and Eq.(13), respectively. In the following, we drop the dependency on the time step for ease of notation.

Injectivity is equivalent to checking: $F(a_1) = F(a_2) \Longrightarrow a_1 = a_2$. This, however, doesn't hold in case of SGLD as the noise terms ξ_1 and ξ_2 can be chosen such that the equality

$$a_1 + \epsilon\nabla_{a_1} \log p\left(a_1\right) + \sqrt{2\epsilon}\xi_1 \; = \; a_2 + \epsilon\nabla_{a_2} \log p\left(a_2\right) + \sqrt{2\epsilon}\xi_2$$

holds with $a_1 \neq a_2$. Therefore, SGLD is not injective. The same holds for DGLD, where the equality $a_1 + \nabla_{a_1} \log p\left(a_1\right) = a_2 + \nabla_{a_2} \log p\left(a_2\right)$ can be valid for $a_1 \neq a_2$. A counter-example: $a_1 = a_2 + \eta$ and $\eta + \nabla_{a_1} \log p\left(a_1\right) = \nabla_{a_2} \log p\left(a_2\right)$, with η being an arbitrary constant.

Surjectivity is equivalent to checking: $\forall a^{l+1} \in \mathbb{R}^d, \exists a^l \in \mathbb{R}^d$ s.t. $a^{l+1} = F(a^l)$. Assume that $a^{l+1} = a^l + \epsilon\nabla_{a^l} \log p\left(a^l\right)$, and $a^l \in \mathbb{R}^d$, we can always choose an adaptive learning rate ϵ such that $\epsilon\nabla_{a^l} \log p\left(a^l\right) = a^{l+1} - a^l$.

G.1.2 Method2: Implicit function theorem

We compute the derivative of the update rule in Eq 12 with respect to a: $J_F = I + \epsilon\nabla_a^2 \log p(a)$. It's possible for J_F not to be invertible, *e.g.*, in case $I = -\epsilon\nabla_a^2 \log p(a)$. Hence, in general $F(a)$ is not guaranteed to be a bijection. ☐

G.2 Hamiltonian Monte Carlo (HMC)

Proposition (HMC). *The HMC update in Eq.*(14) *is not invertible w.r.t. a.*

Neal et al. (2011) show that HMC update rule is only invertible with respect to the (a, v), *i.e.*, when conditioning on v. Since v is sampled from a random distribution, it has the effect of the noise variable in SGLD. Hence, a similar proof applies.

G.3 Stein Variational Gradient Descent

Proposition (SVGD). *Under the assumption that $\epsilon \ll \sigma$, the update rule of SVGD dynamics defined in Eq.*(1) *with an RBF kernel is invertible.*

G.3.1 Method2: SVGD is invertible $\Longleftrightarrow$ SVGD is a bijection

Injectivity. The equality $F(a_1) = F(a_2)$:

$$a_1 + \frac{\epsilon}{m}\sum_{j=1}^{m}\left[k\left(a_j, a_1\right)\nabla_{a_j} \log p\left(a_j\right) - \nabla_{a_j} k\left(a_j, a_1\right)\right] \; = \; a_2 + \frac{\epsilon}{m}\sum_{l=1}^{m}\left[k\left(a_l, a_2\right)\nabla_{a_l} g\left(a_l\right) - \nabla_{a_l} k\left(a_l, a_2\right)\right]$$

is too complex to hold for a solution other than $a_1 = a_2$ given the sum over multiple particles on both sides and the dependency on the kernel.

$\Longrightarrow$ not obvious (depends on the Kernel)

Surjectivity. Similarly, to Langevin dynamics, surjectivity can be achieved by choosing a suitable learning rate.

G.3.2 Method 2: Implicit function theorem

We start by proving the proposition above for the 1-Dimensional case, *i.e.*, $a \in \mathbb{R}$. Then, we extend the proof to the multi-dimensional case, *i.e.*, $a \in \mathbb{R}^d$.
1-Dimensional Case. We prove that F is invertible by showing that F is bijective, which is equivalent to showing that F is strictly monotonic, *i.e.*, $\nabla_{a_i} F(a_i) > 0$ or $\nabla_{a_i} F(a_i) < 0, \quad \forall a_i$.
Computing the derivative of the SVGD update (Eq. 1) rule w.r.t a_i results in:

$$\nabla_{a_i} F(a_i) = 1 + \frac{\varepsilon}{m} \sum_{i=1}^{m} \nabla_{a_i} k\left(a_i, a_j\right) \nabla_{a_j} g\left(a_j\right) + \nabla_{a_i} \nabla_{a_j} k\left(a_i, a_j\right).$$

For $k(a_i, a_j) = e^{-\frac{\|a_i - a_j\|^2}{2\sigma^2}}$, we have: $\begin{cases} \nabla_{a_j} k(a_i, a_j) = \frac{-2(a_i - a_j)}{2\sigma^2} k(a_i, a_j) = \frac{-(a_i - a_j)}{\sigma^2} k(a_i, a_j) \\ \nabla_{a_j} k(a_i, a_j) = \frac{(a_i - a_j)}{\sigma^2} k(a_i, a_j) \\ \nabla_{a_i} \nabla_{a_j} = \frac{1}{\sigma^2} k(a_i, a_j) \left(1 - \frac{1}{\sigma^2} \|a_i - a_j\|^2\right) \end{cases}$

Hence,

$$\nabla_{a_i} F(a_i) = 1 + \frac{\epsilon}{m} \sum_{i=1}^{m} \frac{k(a_i, a_j)}{\sigma^2} \left(-(a_i - a_j) \nabla_{a_j} \log p(a_j) + 1 - \frac{\|a_i - a_j\|}{\sigma^2}\right).$$

Next, under the condition $\epsilon < \sigma$, we show that $\nabla_{a_i} F(a_i) > 0, \forall a_i$. This is equivalent to showing that $\nabla_{a_i} F(a_i) > -1$.

$$\nabla_{a_i} F(a_i) > -1 \iff \frac{\epsilon}{m\sigma^2} \sum_{j=1}^{m} k(a_i, a_j) \left(-(a_i - a_j) \nabla_{a_j} \log p_{a_j}(a_j) + 1 - \frac{\|a_i - a_j\|^2}{\sigma^2}\right) > -1$$

We compute a lower bound on the LHS and investigate when it's strictly larger than -1. We can safely assume that $-3\sigma \leq k(a_i, a_j)(a_i - a_j) \leq 3\sigma$ and $-3\sigma \leq k(a_i, a_j)\|a_i - a_j\|^2 \leq 3\sigma$. We compute the lower bound as: $\sum_{j=1}^{m} \frac{\epsilon\alpha}{m\sigma^2} \left(-3\sigma \|\nabla_{x_j} \log p(x_j)\| + 1 - \frac{(3\sigma)^2}{\sigma^2}\right) = \frac{\epsilon\alpha}{m\sigma^2} \left(-3\sigma \sum_{j=1}^{m} \|\nabla_{x_j} \log p(x_j)\| - 8m\right)$. This results in:

$$\nabla_{a_i} F(a_i) > \frac{\epsilon\alpha}{m\sigma^2} \left(-3\sigma \sum_{j=1}^{m} \|\nabla_{a_j} \log p(a_j)\| - 8m\right) > -1 \iff \sum_{j=1}^{m} \|\nabla_{a_j} \log p(a_j)\| < \frac{m\sigma}{3\epsilon\alpha} - \frac{8m}{3\sigma} \tag{22}$$

Hence, $\max_{a_j} \|\nabla_{a_j} \log p(a_j)\| < \frac{\sigma}{3\epsilon\alpha} - \frac{8}{3\sigma}$. The LHS is guaranteed to be a large positive number when $\epsilon \ll \sigma$.

Multi-Dimensional Case. We assume that $\log p(a_j)$ is continuously differentiable. Note that in practice, this can be easily satisfied by choosing the activation function to be Elu instead of Relu. We can easily show that:

$$\nabla_{a_i} F(a_i) = I + \frac{\epsilon}{m\sigma^2} \sum_{i=1}^{m} k(a_i, a_j) \left(-\nabla_{a_j} \log p(a_j)(a_i - a_j)^\top - \frac{1}{\sigma^2}(a_i - a_j)(a_i - a_j)^\top + I\right)$$

Next, we will show that $\nabla_{a_i} F(a_i)$ is diagonally dominated and is, hence, invertible, *i.e.*, $\det(\nabla_{a_i} h(a_i)) \neq 0$. For this, we show that $\nabla_{a_i} h(a_i)|_{kl} < 1, \forall k, l \in [1, d]$.

$$\nabla_{a_i} h(a_i)|_{kl} = \frac{1}{m} \sum_{i=1}^{m} k(a_i, a_j) \left(-\partial_{a_j^{(k)}} \log p(a_j)(a_i^{(l)} - a_j^{(l)}) - (a_i^{(k)} - a_j^{(k)})(a_i^{(l)} - a_j^{(l)}) + 1\right)$$

Following the proof in Section G.3.2 for the 1-Dimensional case, we show that $\nabla_{a_i} h(a_i)|_{kl} \ll 1$ if $\sigma \ll \epsilon$.

H Derivation of Closed-Form Likelihood for Samplers

H.1 Proof of Theorem 3.3

Theorem. *The closed-form estimate of the log-likelihood* $\log q^L(a^L|s)$ *for the SVGD-based sampler with an RBF kernel* $k(\cdot,\cdot)$ *is*

$$\log q^L(a^L|s) \approx \log q^0(a^0|s) - \frac{\epsilon}{m\sigma^2}\sum_{l=0}^{L-1}\sum_{\substack{j=1\\ a^l\neq a^l_j}}^{m} k(a^l_j,a^l)\left(-(a^l-a^l_j)^\top\nabla_{a^l_j}Q(s,a^l_j) - \frac{\alpha}{\sigma^2}\|a^l-a^l_j\|^2 + d\alpha\right),$$

where d *is the feature space dimension.*

Proof. We generate a chain of samples using SVGD starting from $a^0\sim q^0$, and following the update rule $a^{l+1}_i \leftarrow a^l_i + \epsilon\, h(a^l_i,s)$, where $h(a^l_i,s)=\mathbb{E}_{a^l_j\sim q^l}\Big[k(a^l_i,a^l_j)\nabla_{a^l_j}Q(s,a^l_j)+\nabla_{a^l_j}k(a^l_i,a^l_j)\Big]$ and $k(a^l_i,a^l_j)=\exp(-\frac{\|a^l_i-a^l_j\|^2}{2\sigma^2})$. This update rule is the optimal direction in the reproducing kernel Hilbert space of $k(\cdot,\cdot)$ for minimizing the KL divergence objective (actor loss):

$$\pi^* = \arg\min_\pi \sum_t \mathbb{E}_{s_t\sim\rho_\pi}\Big[D_{KL}\big(\pi(\cdot|s_t)\|\exp(Q(s_t,\cdot)/\alpha)/Z\big)\Big]. \tag{23}$$

According to Proposition 3.2, the iteration step (Eq.(1)) is invertible. Hence, following Theorem 3.1 and substituting $h(a^l_i,s)$ with the above formula for SVGD, for each action particle a^L_i we obtain:

$$\log q^L(a^L_i) \approx \log q^0(a^0_i) - \frac{1}{m}\sum_{l=0}^{L-1}\sum_{\substack{j=1\\ a^l_i\neq a^l_j}}^{m}\Big[\underbrace{\mathrm{Tr}\Big(\nabla_{a^l_i}(k(a^l_i,a^l_j)\nabla_{a^l_j}Q(s,a^l_j))\Big)}_{①} + \underbrace{\mathrm{Tr}\,\alpha\Big(\nabla_{a^l_i}\nabla_{a^l_j}k(a^l_i,a^l_j)\Big)}_{②}\Big].$$

Note that we empirically approximate the expectation in $h(a^l_i,s)$ by an empirical mean over particles that are different from a^l_i, in order to avoid computing Hessians in the derivation below. Next we compute simplifications for terms ① and ② respectively. In the following, we denote by $(\cdot)^{(k)}$ the k-th dimension of the vector.

Term ①:

$$\begin{aligned}
\mathrm{Tr}\Big(\nabla_{a^l_i}(k(a^l_j,a^l_j)\nabla_{a^l_j}Q(s,a^l_j))\Big) &= \mathrm{Tr}\Big(\nabla_{a^l_i}k(a^l_j,a^l_j)(\nabla_{a^l_j}Q(s,a^l_j))^\top + k(a^l_j,a^l_j)\nabla_{a^l_i}\nabla_{a^l_j}Q(s,a^l_j))\Big)\\
&= \sum_{t=1}^{d}\frac{\partial k(a^l_j,a^l_j)}{\partial(a^l_i)^{(t)}}\frac{\partial Q(s,a^l_j)}{\partial(a^l_i)^{(t)}} + 0\\
&= (\nabla_{a^l_i}k(a^l_j,a^l_j))^\top\nabla_{a^l_j}Q(s,a^l_j)\\
&= -\frac{\alpha}{\sigma^2}k(a^l_j,a^l_j)(a^l_i-a^l_j)^\top\nabla_{a^l_j}Q(s,a^l_j)
\end{aligned}$$

Term ②:

$$\begin{aligned}
\mathrm{Tr}\Big(\nabla_{a^l_i}\nabla_{a^l_j}k(a^l_i,a^l_j)\Big) &= \alpha\,\mathrm{Tr}\Big(\nabla_{a^l_i}\Big(\frac{1}{\sigma^2}k(a^l_i,a^l_j)(a^l_i-a^l_j)\Big)\Big)\\
&= \frac{\alpha}{\sigma^2}\mathrm{Tr}\Big(\nabla_{a^l_i}k(a^l_i,a^l_j)(a^l_i-a^l_j)^\top + k(a^l_i,a^l_j)\cdot I\Big)\\
&= \frac{\alpha}{\sigma^2}\mathrm{Tr}\Big(-\frac{1}{\sigma^2}k(a^l_i,a^l_j)(a^l_i-a^l_j)(a^l_i-a^l_j)^\top + k(a^l_i,a^l_j)\cdot I\Big)\\
&= \frac{\alpha}{\sigma^2}\sum_{t=1}^{d}\Big(-\frac{1}{\sigma^2}k(a^l_i,a^l_j)(a^l_i-a^l_j)^{(t)}(a^l_i-a^l_j)^{(t)} + k(a^l_i,a^l_j)\Big)\\
&= -\frac{\alpha}{\sigma^4}\times k(a^l_i,a^l_j)\|a^l_i-a^l_j\|^2 + \frac{\alpha}{\sigma^2}\times d\times k(a^l_i,a^l_j)\\
&= k(a^l_i,a^l_j)\Big(-\frac{\alpha}{\sigma^4}\|a^l_i-a^l_j\|^2 + \frac{d\alpha}{\sigma^2}\Big)
\end{aligned}$$

By combining **Terms ①** and **②**, we obtain:

$$\log q^L(a_i^L) \approx \log p^0(a_i^0) - \frac{\epsilon}{m\sigma^2} \sum_{l=0}^{L-1} \sum_{j=1}^{m} k(a_j^l, a_j^l) \left(-(a_i^l - a_j^l)^\top \nabla_{a_j^l} Q(s, a_j^l) - \frac{\alpha}{\sigma^2} \|a_i^l - a_j^l\|^2 + d\alpha \right)$$

Proof done if we take a generic action particle a_i in place of a. $\square$

I Additional Results: Entropy Evaluation

The SVGD hyperparameters for this set of experiments are summarized in Table 2. We include additional figures for (1) the effect of (2) the kernel variance (Figure 9) and (2) number of SVGD steps and particles (Figure 10).

Table 2: Parameters

	Parameter	Value
Figure 4a-4b	Target distribution Initial distribution	$p = \mathcal{N}([-0.69, 0.8], [[1.13, 0.82], [0.82, 3.39]])$ $q^0 = \mathcal{N}([0,0], 6\boldsymbol{I})$
Figure 4c	Target distribution Initial distribution	$p_{\text{GMM}_M} = \sum_{i=1}^{M} \mathcal{N}([0,0], 0.1\boldsymbol{I})/M$ $q^0 = \mathcal{N}([0,0], 6\boldsymbol{I})$
Default SVGD parameters	Learning rate Number of steps Number of particles Kernel variance	$\epsilon = 0.5$ $L = 200$ $m = 200$ $\sigma = 5$

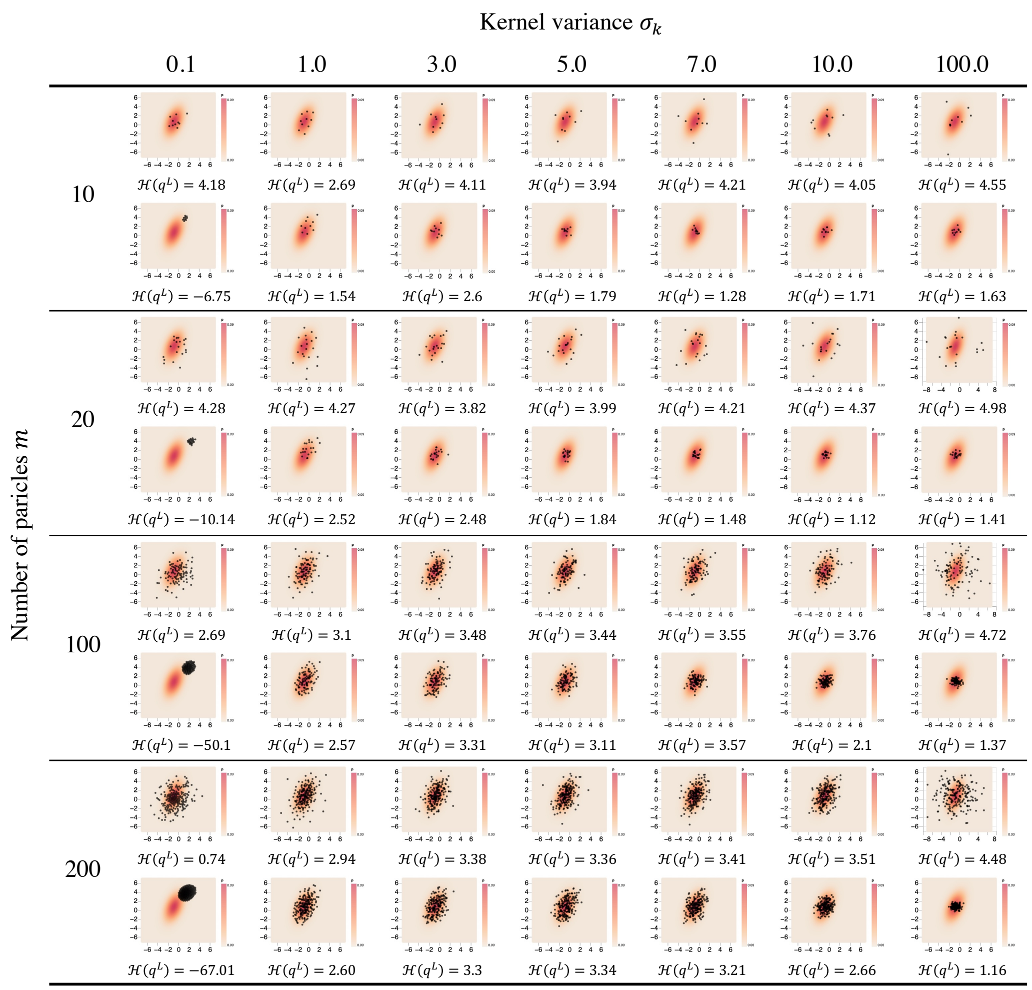
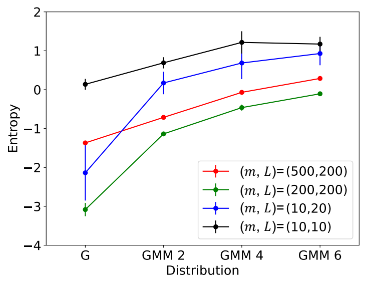
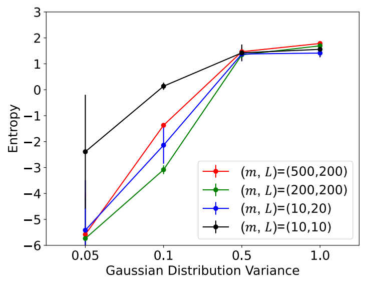


Figure 9: Visualization of the particles after L steps of SVGD for the different configurations of kernel variance σ and number of particles m in Figure 4b.



(a) GMM distributions



(b) Gaussian distributions

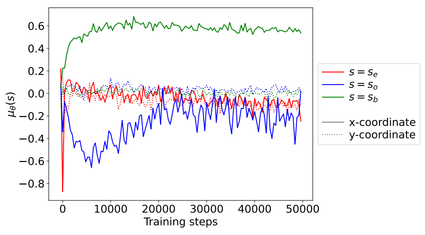
Figure 10: Sensitivity of our entropy formula to the number of SVGD steps (L) and particles (m). Our entropy consistently increases with increasing σ and increasing number of GMM components, even when a small number of SVGD steps and particles is used *e.g.*, $L = 10, m = 10$.

J Additional Results: Multi-goal Results

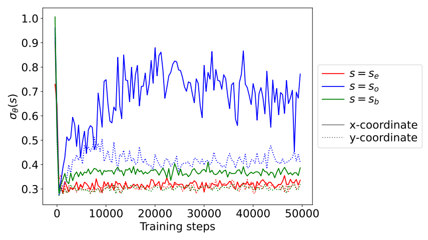
Hyperparameters are reported in Table 3. Additionally, we include results for (1) the effect of the parametrization of the initial distribution (Figure 14), (2) the entropy heatmap (Figure 13), (3) the effect of the entropy on the learned Q-landscapes (Figure 14 and Figure 15), (4) the robustness/adaptability of the learned policies (Figure 16) and (5) Amortized S^{2}AC results (Figure 17).

Table 3: Hyperparameters for multi-goal environment.

	Hyperparameter	Value
Training	Optimizer	Adam
	Learning rate	$3 \cdot 10^{-4}$
	Batch size	100
Deepnet	Number of hidden layers (all networks)	2
	Number of hidden units per layer	256
	Nonlinearity	ReLU
RL	Discount factor γ	0.8
	Replay buffer size $\lvert\mathcal{D}\rvert$	10^6
	Target smoothing coefficient	0.005
	Target update interval	1
SVGD	initial distribution q^0	$\mathcal{N}(\mathbf{0}, 0.3\boldsymbol{I})$
	Learning rate ϵ	0.01
	Number of steps L	10
	Number of particles m	10
	Particles range (num. std) t	3
	Kernel variance	$\sigma = \frac{\sum_{i,j} \lVert a_i - a_j \rVert^2}{4(2\log m+1)}$



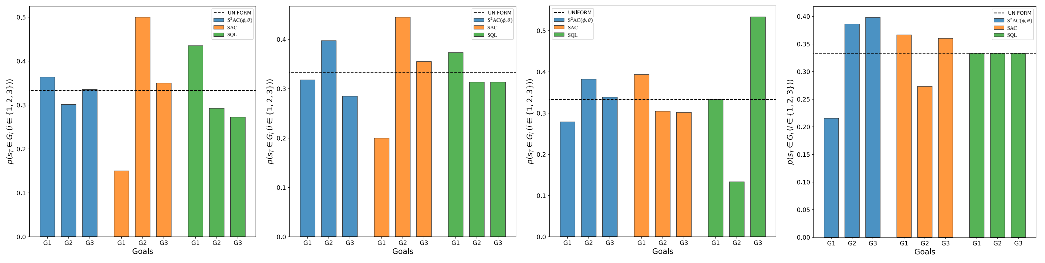
(a) Mean $\mu_\theta(s)$



(b) Standard deviation $\sigma_\theta(s)$

Figure 11: Trends of x and y coordinates for the mean and standard deviation of the parameterized initial distribution for some critical states, during training.

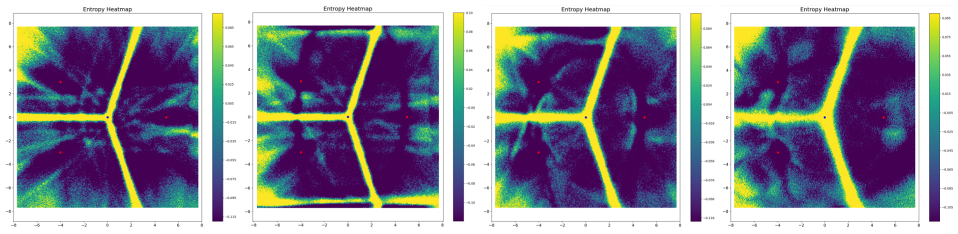
Distribution of reached goals for the multi-goal environment. Figure 12 shows the distribution of reached goals for S^{2}AC/SAC for the agents in Figure 6. Trajectories are collected from 20 episodes of 20 different agents trained with 20 different seeds for each algorithm. We observe that with higher α's, more agent trajectories converge to the left two goals (G2 and G3), which is not the case for SAC and SQL. This shows that S^{2}AC learns a more optimal solution to the MaxEnt objective in Eq.(2).



(a) $\alpha = 0.2$ (b) $\alpha = 1$ (c) $\alpha = 10$ (d) $\alpha = 20$

Figure 12: Distribution of reached goals for S^{2}AC, SAC and SQL with different α's. The x-axis denotes different goals. The y-axis represents the ratio of trajectories that reach the goal.

Entropy heatmap of S^{2}AC in the multi-goal environment. Figure 13 shows the entropy heatmap of S^{2}AC with different α's. A brighter color corresponds to higher entropy. For S^{2}AC, the higher α, the higher the entropy on the left quadrant compared to the right one, i.e., the more contrast between the left and the right quadrants. For instance, In Figure 13d (S^{2}AC, $\alpha = 20$), notice a clear green/yellow patch spanning the left side, while the right side is mostly dark blue except for the edges.



(a) S^{2}AC, $\alpha = 0.2$ (b) S^{2}AC, $\alpha = 1$ (c) S^{2}AC, $\alpha = 10$ (d) S^{2}AC, $\alpha = 20$

Figure 13: The entropy heatmap of S^{2}AC in the multi-goal environment for different α

Smoothness of the Q-landscapes. To assess the effect of the entropy, we visualize the Q-landscapes corresponding to six typical states $s \in \{s_o, s_a, s_b, s_c, s_d, s_e\}$ (marked in blue on the upper left of Figure 14) across different trajectories to the goal and report their associated entropy $\mathcal{H}(\cdot|s)$ (bottom left of Figure 14). The blue dots correspond to 10 SVGD particles at convergence. We observe that the Q-landscape becomes smoother with increasing α. For instance, notice how the modes for state s_c become more connected. Quantified measurements of smoothness are in Figure 15. We use two metrics M_1 and M_2 to measure the smoothness of the learned Q-landscape: (1) M_1: the average over the L1-norm of the gradient of the Q-value with respect to the actions across trajectories, *i.e.*, $\mathbb{E}_{\tau\sim\pi(a|s)}\left[\mathbb{E}_{(s_t,a_t)\in\tau}\left[\frac{||\nabla_{a_t}Q(s_t,a_t)||_1}{d}\right]\right]$. (2) M_2: The average over the L1-norm of the Hessian of the Q-value with respect to the actions across trajectories, *i.e.*, $\mathbb{E}_{\tau\sim\pi(a|s)}\left[\mathbb{E}_{(s_t,a_t)\in\tau}\left[\frac{1}{d^2}\sum_{i,j}|\nabla^2_{a_t}Q(s_t,a_t)|_{i,j}\right]\right]$. Figure 15 shows that increasing α leads to consistently smaller gradients (Figure 15a) and less curvature (Figure 15b). Hence, the entropy results in a smoother landscape that helps the sampling convergence.

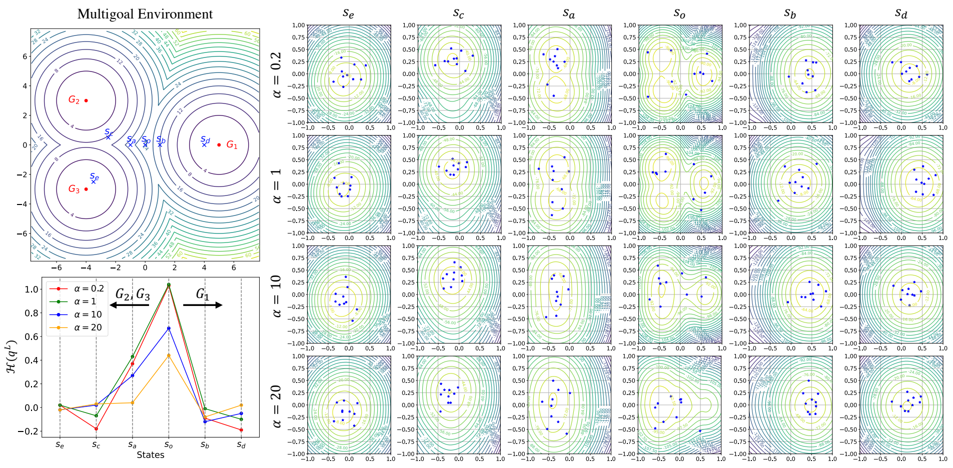
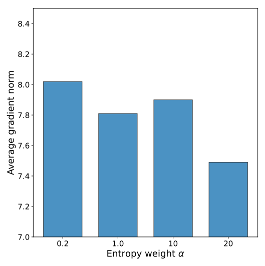
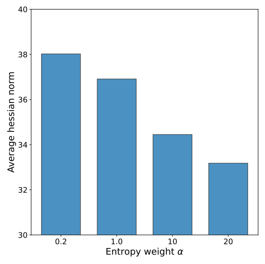


Figure 14: Results on the multi-goal environment. Increasing α yields smoother landscapes (*e.g.*, s_o). Notice how the modes become more connected (*e.g.*, for $s = s_a$ with increasing α). The entropy at the different states is reported in the lower left figure.



(a) Average gradient across trajectories



(b) Average Hessian across trajectories

Figure 15: Quantitative evaluation of the smoothness the Q-landscape of S^{2}AC for different α's.

Parametrization of q^0. In Figure 11, we visualizes the coordinates of the mean $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$ of q^0_θ at different states $s \in \{s_o, s_e, s_b\}$ in the multigoal environment. As training goes on, $\mu_\theta(s)$ shifts closer to the nearest goals. For example, $\mu_\theta(s_b)$ becomes more positive during the training as it is shifting to G_1. Additionally, the model learns a high variance $\sigma_\theta(s)$ for the multimodal state s_o and becomes more deterministic for the unimodal ones (*e.g.*, s_e and s_b). As a result, in Figure 7, we observe that S^{2}AC(ϕ, θ) requires a smaller number of steps to convergence than S^{2}AC(ϕ).

Entropy estimation. Figure 14 shows that the entropy is higher for states on the left side due to the presence of two goals, as opposed to a single goal on the right side (*e.g.*, $\mathcal{H}(\pi_\theta(\cdot|s_a)) < \mathcal{H}(\pi_\theta(\cdot|s_o))$). Also, the entropy decreases when approaching the goals (*e.g.*, $\mathcal{H}(\pi_\theta(\cdot|s_d)) < \mathcal{H}(\pi_\theta(\cdot|s_b)) < \mathcal{H}(\pi_\theta(\cdot|s_o))$). The same is valid along the paths to goal G_1.

Robustness/Adaptability. In Figure 16, we report the distribution of reached goals after hitting an obstacle for S^{2}AC, SAC and SQL for different α's. Notice that S^{2}AC robustness, measured by the probability of reaching the goal for S^{2}AC is consistently increasing with increasing α. Intuitively, exploration is better with large values of α, leading to better learning of the Q-landscape. In other words, from a given state, the agent is more likely to have explored more sub-optimal ways to reach the goal. So, when the optimal path is blocked with the barrier, the agents trained with S^{2}AC are more likely to have learned several other ways to go around it. This is different from SAC, when the policy is uni-modal (Gaussian) and the agents are only able to escape the barrier and get to the goal for large α's ($\alpha \in 10, 20$). However, robustness in the case of SAC trained with large α's come at the expense of performance, *i.e.*, increased number of steps (See row 3 in Figure 7). Besides, note that the number of SAC agents reaching the goals for $\alpha = 20$ is less than the one for $\alpha = 10$. This is due to the fact that higher α's lead to higher stochasticity and less structured exploration (the standard

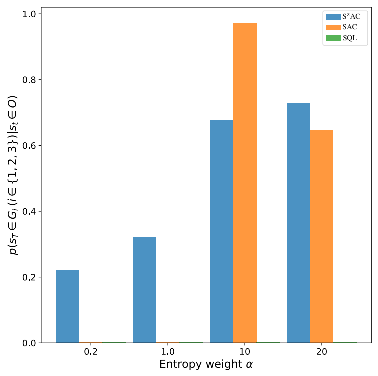
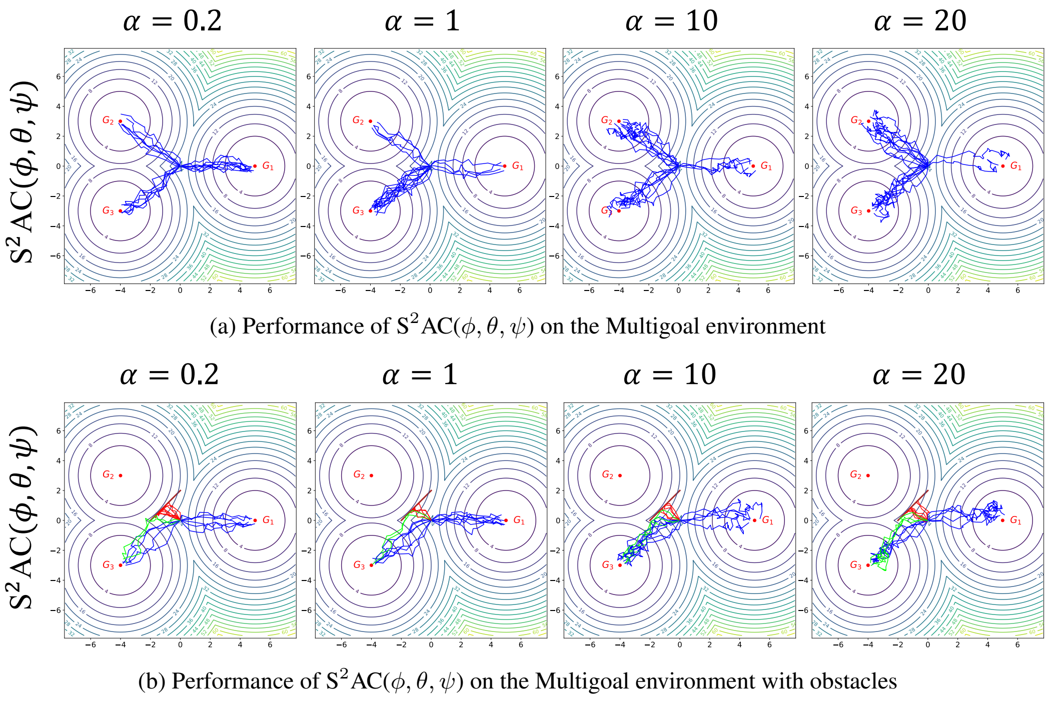


Figure 16: Distribution of reached goals after hitting an obstacle for S^2AC, SAC and SQL.

deviation of the Gaussian becomes very large). SQL fails to reach the goals once the obstacle is added. This shows that the implicit entropy in SQL is not as efficient as the explicit entropy in SAC and S^2AC.

Amortized S^2AC. In Figure 17, we report results of the amortized version of S^2AC, *i.e.*, S^2AC(ϕ, θ, ψ) on the multigoal environment. Performance and robustness are comparable with the non-amortized version S^2AC(ϕ, θ) while having a faster inference (feedforward pass through $f_\psi(s, z)$).



(a) Performance of S^2AC(ϕ, θ, ψ) on the Multigoal environment

(b) Performance of S^2AC(ϕ, θ, ψ) on the Multigoal environment with obstacles

Figure 17: Performance of Amortized S^2AC on the Multigoal environment

K ADDITIONAL RESULTS: MUJOCO

Table 4 lists the S^{2}AC hyper-parameters used in our experiments. Additionally, we give details on accelerating S^{2}AC.

Table 4: Hyperparameters

	Hyperparameter	Value
Training	Optimizer	Adam
	Learning rate	$3 \cdot 10^{-4}$
	Batch size	100
Deepnet	Number of hidden layers (all networks)	2
	Number of hidden units per layer	256
	Number of samples per minibatch	256
	Nonlinearity	ReLU
RL	Target smoothing coefficient	0.005
	Discount γ	0.99
	Target update interval	1
	Entropy weight α	1.0 for all environments, 0.2 Ant
	Replay buffer size $\lvert\mathcal{D}\rvert$	10^6
SVGD	initial distribution q_0	$\mathcal{N}(\mathbf{0}, 0.5\boldsymbol{I})$
	Learning rate ϵ	0.1
	Number of steps L (S^{2}AC(ϕ))	20
	Number of steps L (S^{2}AC(ϕ, θ))	3
	Number of particles m	10
	Particles range (num. std) t	3
	Kernel variance	$\sigma = \frac{\sum_{i,j} \lVert a_i - a_j \rVert^2}{4(2\log m + 1)}$

Computational Efficiency. Compared to SAC, running SVGD for L steps requires L additional back-propagation passes through the Q-network and a factor of m (number of particles) increase in the memory complexity. In order to improve the efficiency of S^{2}AC, we limit the number of particles m to 10/20 and the number of SVGD steps L to 10/20.

Additionally, we experiment with the following amortized version of S^{2}AC. Specifically, we train a deepnet $f_\psi(s, z)$ to mimic the SVGD dynamics during testing, where z is a random vector that allows mapping the same state to different particles. Note that we cannot use this deepnet during training as we need to estimate the closed-form entropy which depends on the SVGD dynamics. One way to train $f_\psi(s, z)$ is to run SVGD to convergence and train $f_\psi(s, z)$ to fit SVGD outputs. This however requires collecting a large training set of state action pairs by repeatedly deploying the policy. This might be slow and result in low coverage of the states that are rarely visited by the learned policy and hence result in poor robustness in case of test time perturbations. We instead propose an incremental approach in which ψ is iteratively adjusted so that the network output $a = f_\psi(s, z)$ changes along the Stein variational gradient direction that decreases the KL divergence between the policy and the EBM distribution, *i.e.*,

$$\Delta f_\psi(z, s) = \frac{1}{m} \sum_{i=1}^{m} k(a_i, f_\psi(s, z)) \nabla_{a_i} Q(s, a_i) + \alpha \nabla_{a_i} k(a_i, f_\psi(s, z)) \tag{24}$$

Note that Δf_ψ is the optimal direction in the reproducing kernel Hilbert space, and is thus not strictly the gradient of Eq.(5), but it still serves a good approximation, *i.e.*, $\frac{\partial J}{\partial a_t} \propto \Delta f_\psi$, as explained by Wang & Liu (2016). Thus, we can use the chain rule and backpropagate the Stein variational gradient into the policy network according to

$$\frac{\partial J(s)}{\partial \psi} \propto \mathbb{E}_z \left[\Delta f_\psi(s, z) \frac{\partial f_\psi(z, s)}{\partial \psi} \right]. \tag{25}$$

to learn the optimal sampling network parameters ψ^*. Note that the amortized network takes advantage of a Q-value that estimates the expected future entropy which we compute via unrolling the SVGD steps using Eq (3.3).

The modified S^{2}AC algorithm is described in Algorithm 2.

Algorithm 2 Stein Soft Actor Critic (S^{2}AC) with Amortized policy (test-time)

```
 1: Initialize parameters $\phi$, $\theta$, $\psi$, hyperparameter $\alpha$, and replay buffer $\mathcal{D} \leftarrow \emptyset$
 2: for each iteration do
 3:     for each environment step $t$ do
 4:         Sample action particles $\{a\}$ from $\pi_\theta(\cdot|s_t)$
 5:         Select $a_t \in \{a\}$ using exploration strategy
 6:         Sample next state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
 7:         Update replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r_t, s_{t+1})$
 8:     for each gradient step do
 9:         Critic update:
10:             Sample particles $\{a\}$ from an EMB sampler $\pi_\theta(\cdot|s_{t+1})$
11:             Compute entropy $\mathcal{H}(\pi_\theta(\cdot|s_{t+1}))$ using Eq.(11)
12:             Update $\phi$ using Eq.(8)
13:         Actor update:
14:             Update $\theta$ using Eq.(9)
15:             Update $\psi$ using Eq.(25)
```

Evaluation with the Rliable Library. Performances curves in Figure 8 are averaged over 5 random seeds and then smoothed using Savitzky-Golay filtering with window size 10. Additionally, we report metrics from the Rliable Library (Agarwal et al., 2021) in Fig. 8, including

- **Median**: Confidence interval of the median performance of each algorithm across different seeds, averaged over different MuJoCo environments.
- **Mean**: Confidence interval of the average performance of each algorithm across different seeds and environments.
- **IQM (Interquantile means)**: Instead of computing the average performance on all trials, IQM shows the mean of the middle 50 percent of performance across different seeds.
- **Optimality Gap**: The area between results curve of baseline algorithms and the horizontal line at the average performance of S^{2}AC (ϕ, θ).
- **Probability of improvement over baselines**: The average probability that S^{2}AC (ϕ, θ) can make performance improvements over baseline algorithms.

The parameterized version of S^{2}AC has the best performance among baselines in all the considered metrics. It has a probability of $\sim$65% in outperforming SAC-NF and $\sim$80% in outperforming IAF.