# Universal Stabilization for Maximum Entropy Optimization in Reinforcement Learning

Xing Chen, Yewen Li, Xiaofeng Cao, Hechang Chen, Hengshuai Yao, Bo An, *Senior Member, IEEE*, and Yi Chang, *Senior Member, IEEE*

***Abstract*—In real-world decision-making tasks, it is critical for reinforcement learning (RL) methods to be both stable and robust. Maximum entropy RL methods typically generate a robust policy with entropy augmented reward. While incorporating entropy into the reward offers the benefit of exploration, it presents limited universal applicability and persistent convergence difficulties, such as suboptimal policy stabilization and unstable $Q$ value update. From optimization, we define these two issues as *tremulous policy* and *spiky Q-function*, investigating their underlying causes and relationships. Analysis with this, the maximum entropy principle leads to a spiky $Q$-function update, which ultimately results in a tremulous policy. We thus introduce a beta-symmetric Kullback–Leibler (KL) divergence objective to mitigate such issues under the maximum entropy framework. With this objective function, the tremulous nature of the policy could be controlled with a large beta value. The spiky $Q$-function could be avoided by annealing the entropy in the target $Q$ value, as the beta-symmetric KL divergence is an upper bound of the original reverse KL divergence. Theoretically, we prove that minimizing our new objective function results in a new policy that presents an improvement in the $Q$ value. Guaranteed by these results, we ultimately derive the optimal policy by iteratively updating the $Q$ value and policy, and we call this method max-entropy stable optimization (MeSO). Experimental results on the Mujoco and Roboschool platforms demonstrate that our algorithm maintains stability while offering better flexibility and overall performance.**

***Index Terms*—Maximum entropy, policy iteration, reinforcement learning (RL) algorithm, sample efficiency.**

Received 21 July 2024; revised 18 June 2025 and 4 October 2025; accepted 21 October 2025. Date of publication 20 November 2025; date of current version 6 April 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62476110, Grant 62476109, Grant 62206108, and Grant U2341229; in part by the National Key Research and Development Program of China under Grant 2021ZD0112500; in part by the Key Research and Development Project of Jilin Province under Grant 20240304200SF; and in part by the International Cooperation Project of Jilin Province under Grant 20220402009GH. The work of Xing Chen was supported by China Scholarship Council Program under Grant 202206170041. *(Corresponding authors: Xiaofeng Cao; Yi Chang.)*

Xing Chen, Xiaofeng Cao, and Hechang Chen are with the School of Artificial Intelligence, Jilin University, Changchun 130012, China, and also with the Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, Changchun 130012, China (e-mail: xingchen19@mails.jlu.edu.cn; xiaofengcao@jlu.edu.cn).

Yewen Li and Bo An are with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798.

Hengshuai Yao is with the Department of Computing Science, University of Alberta, Edmonton, AB T6G 2R3, Canada, and also with Sapient Intelligence, Singapore 069534.

Yi Chang is with the School of Artificial Intelligence and the International Center of Future Science, Jilin University, Changchun 130012, China, and also with the Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, Changchun 130012, China (e-mail: yichang@jlu.edu.cn).

Data is available on-line at https://github.com/raincchio/MeSO

Digital Object Identifier 10.1109/TNNLS.2025.3626050

## I. Introduction

In THE field of control decision-making, deep reinforcement learning (DRL) based on the maximum entropy principle has demonstrated remarkable capabilities compared to traditional algorithms, exemplified by the emergence of algorithms like soft $Q$-learning [1] and soft actor-critic [2]. Maximum entropy DRL algorithms encourage action exploration [3], which may achieve stochastic optimality [4] with noise and yield superior results. Furthermore, entropy, as a measure of uncertainty [5], endows strategies that incorporate entropy as a reward function with certain robustness [6], making them highly deployable in real-world scenarios. It is a valuable research direction [7], [8] to utilize the exploratory properties of entropy.

Entropy is closely related to policy optimization [9], [10], $Q$ value update [2], [11], and exploration [12], [13]. Generally, large entropy is beneficial for exploration but detrimental to policy optimization and $Q$ value update. Since reinforcement learning (RL) requires extensive exploration, the maximum entropy framework [3], [14] is regarded as a better choice compared to random exploration. We aim to minimize the negative effects of entropy on policy optimization and value update within the maximum entropy framework to achieve a stable policy.

However, previous methods [1], [2], [15] that focus on pursuing maximum entropy overlook the potential issues associated with it. We summarize these issues into two aspects. The first aspect concerns the impact on the policy, which we define as the *tremulous policy*. The max-entropy reward encourages action exploration, resulting in consecutive actions that can vary significantly, which directly causes tremulousness in continuous control. The second aspect concerns the impact on $Q$ value update, which we define as the *spiky Q-function*. The high variance in the estimation of entropy results in a high variance of the target $Q$ value, causing a spiky $Q$-function. This indirectly influences the policy gradient, which is derived by calculating the derivative of the $Q$-function with respect to the action. These two issues exist in the schema of the maximum entropy framework.

Inspired by the maximize entropy minimize cross-entropy principle [16], it is prudent to consider some stability with prior information rather than merely seeking to maximize entropy. Constrained max-entropy ensures the robustness of the policy while guaranteeing some stability. This consideration becomes particularly crucial when attempting to apply

algorithms to real robots [17], as opposed to merely simulation environments.

Specifically, we propose beta-symmetric Kullback–Leibler (KL) divergence, which is a variant of $f$-divergence, as a solution for controlling the impact of entropy with the prior information of $Q$ value. Intuitively, with beta-symmetric KL divergence, we not only consider maximizing the entropy of the current policy but also minimizing the difference between the entropy of the current policy and the entropy of the prior distribution. Theoretically, this objective guarantees policy improvement. It establishes a foundation for iteratively performing policy evaluation and improvement steps, ensuring the convergence of the policy. The impact of entropy on policy optimization can be controlled through a parameter beta. On the other hand, the policy derived by minimizing the beta-symmetric KL divergence can improve the $Q$ value not only under the max-entropy framework but also under the original RL framework. This implies that the impact of entropy on $Q$ value update can be eliminated by gradually reducing the proportion of entropy in the target $Q$ value.

Overall, we propose a practical algorithm, named max-entropy stable optimization (MeSO), to better handle the impact of entropy on policy optimization and $Q$ value update under the maximum entropy RL framework.

The experiments conducted across multiple environments on Mujoco [18] and Roboschool [19] platforms, simulated with Python by Gym [20], indicate that our algorithm achieves higher sample efficiency and performs better in terms of final performance. Through experiments involving the manipulation of individual variables, it has been demonstrated that our approach yields superior results with flexible control on the impact of entropy. This experimental evidence validates that the MeSO algorithm achieves stable policy optimization and $Q$ value update, leading to improved performance.

*Contributions* Our main contributions are as follows.

1) Analyzing two issues of instability in max-entropy RL, which have been ignored by previous work.
2) Proposing the beta-symmetric KL divergence objective for policy optimization to achieve stable policy optimization in max-entropy RL.
3) Giving theoretical guarantees of convergence to optimal policy with this novel objective.
4) Experimentally validating that our method improves the stability and performance of the max-entropy-based RL algorithm.

## II. Related Work

Entropy in RL is consistently mentioned in various mainstream algorithms, but it is rarely studied as a primary subject. We try to focus on related works from the perspectives of stable $Q$ value update and policy optimization, which have remained consistent focal points in the research of entropy-related RL algorithms.

### A. Q Value Update

The methods for stable $Q$ value update are all based on improvements to $Q$-learning [21]. DQN [22] introduces an experience replay buffer to stabilize the learning of $Q$ value. Double DQN [23], [24] stabilizes the value update by a decoupled target network. TD3 [25] enables stable $Q$-function approximation with the minimum of two target $Q$ values. There are also some papers proposing separate estimators for $Q$ value and $V$ value to achieve better stability of value estimation, such as Dueling DQN [26]. Some methods attempt to introduce entropy into the calculation of $Q$ value to encourage exploration but introduce instability. For example, G-learning [27] proposes soft update with energy-based policy to reduce bias in the $Q$ value estimation. Then, soft $Q$-learning [1] integrates entropy reward to an energy-based policy under the maximum entropy framework. The SAC [28] algorithm further regards entropy as the expectation of negative log probability and introduces this log probability into the target $Q$ value. The improved SAC [29] algorithm stabilizes the $Q$ value update with on-policy samples. The maximum diffusion RL method [13] incorporates the entropy of state transition into the $Q$ value to encourage state exploration. These algorithms incorporate entropy into the calculation of $Q$ value, but they do not adequately address the stability issue, resulting in unstable $Q$ value updates.

### B. Policy Optimization

Policy gradient methods typically offer explicit theoretical guarantees for policy improvement, assuming small changes in both policy and entropy. This includes policy gradient algorithms based on approximate RL [30], as well as numerous subsequent methods. According to the theorem of monotonic policy improvement [31], in situations where the behavior policy and target policy are similar, a stable policy improvement can be ensured. The constrained policy optimization [32], [33] algorithm introduces additional constraints on policy change to guarantee policy improvement. The proximal policy optimization (PPO) [34] algorithm excludes data with significant policy differences during the gradient computation. Some works minimize the KL divergence between the current policy and a constructed $Q$ value distribution, such as relative entropy policy search [35], [36] and maximum a posteriori policy optimization (MPO) [37], [38]. These methods work well in discrete action spaces but struggle to handle continuous action spaces. The SAC [28] algorithm addresses the entropy issue by decaying the weight of stochastic policy gradients during the learning process. However, soft policy iteration introduces challenges in the $Q$ value update, where the introduction of log probability can lead to an unbounded $Q$ value. MPO [37] also considers stability through a KL divergence constraint and approaches the problem from a maximum likelihood perspective. The subsequent $V$-MPO algorithm [38] reformulates MPO into an online version, improving the stability of policy updates. The tail adaptive $f$-divergence [39] modifies the KL divergence used in the SAC to improve the stability of the optimization process. The dual-clip PPO [40] further removes data with tougher standards, while the soft-clip PPO [41] algorithm retains all data but reduces the weight of data from policies with large discrepancies, mitigating their impact on the policy gradient. In a continuous action space, it is hard to control the policy change.

## III. Preliminaries

Consider a discounted infinite-horizon Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r,$ and $\gamma)$, where the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are continuous, and the state transition probability $p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ represents the probability density of the next state. Given the state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$ at time step $t$, we can get the probability density of $\mathbf{s}_{t+1} \in \mathcal{S}$. The environment emits a bounded reward $r: \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ on for specific state and action pair. $\gamma$ is the discount factor, and its value is in the range $[0, 1)$, which makes the infinite accumulated reward finite in mathematics.

### A. Reinforcement Learning

The agent's goal is to use interactions with the environment to find the optimal policy $\pi$ such that the objective $J(\pi)$ is maximized

$$\max_{\pi} J(\pi) = \mathbb{E}_{s_0, a_0, \ldots, \sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$
$$\text{s.t. } s_0 \sim p_0,\ a_t \sim \pi(\cdot|s_t), \quad \mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)\ \ \forall t > 0. \tag{1}$$

For brevity, we abbreviate $\mathbb{E}_{s_0, a_0, \ldots, \sim\pi}[\cdot]$ as $\mathbb{E}_{\pi}[\cdot]$. Maximum entropy RL [42] attempts to encourage exploration by providing entropy as a reward that incentivizes the agent to be more optimistic when making decisions. Such an objective is always solved by RL, which learns state–action value (named $Q$ value) and state value (named $V$ value) by the Bellman equation evaluated on a fixed policy. For a fixed policy $\pi$, the $Q$ value can be computed iteratively, starting from any function $Q: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and repeatedly applying the Bellman backup operator $\mathcal{T}^{\pi}$ given by

$$\mathcal{T}^{\pi} Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\pi}\left[V(\mathbf{s}_{t+1})\right] \tag{2}$$

where

$$V(\mathbf{s}_t) = \mathbb{E}_{\pi}\left[Q(\mathbf{s}_t, \mathbf{a}_t)\right]. \tag{3}$$

In current popular reinforcement learning algorithms, state value or state–action value is involved in the objective function of these algorithms to derive a new policy.

### B. Maximum Entropy RL

Considering the entropy of policy $\mathcal{H}(\cdot)$, the maximum entropy objective becomes

$$\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t))\right)\right] \tag{4}$$

where $\alpha$ denotes the entropy reward scaling coefficient. This objective is solved by the Bellman operator with a modified state value function given by

$$V(\mathbf{s}_t) = \mathbb{E}_{\pi}\left[Q(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log(\pi(\mathbf{s}_t, \mathbf{a}_t))\right]. \tag{5}$$

State-of-the-art max-entropy-based methods [1], [2] often optimize the objective in (4) to get a policy.

## IV. Methodology

We propose minimizing the beta-symmetric KL divergence to optimize policy in the maximum entropy framework. Our core contribution lies in addressing the challenge of stabilizing maximum entropy optimization. We propose a solution based on the beta-KL divergence, which, under mild assumptions, guarantees monotonic policy improvement. In Section IV-A, we discuss two issues of maximum entropy RL and provide an intuitive illustration. In Section IV-B, we give the definition of our new objective function and discuss related properties of it. In Section IV-C, we prove that this novel objective guarantees policy convergence in a tabular setting. In Section IV-D, considering parameterized $Q$-function with a neural network, we compute the policy gradient with the new objective function and analyze the impact of optimizing for entropy change. In Section IV-E, we present a practical DRL algorithm.

### A. Insight

The maximum entropy method incorporates policy entropy as a part of the reward, encouraging action-level exploration but relatively neglecting the stability of consecutive actions. As shown in Fig. 1(a), encouraging action-level exploration results in unsmooth trajectories. We aim to achieve stable trajectories while maintaining diversity, as depicted in Fig. 1(b). The key to achieving smoothness lies in the control of policy, where the entropy seems to be the biggest factor in optimization. We first discuss the sources of instability within the maximum entropy framework. Entropy's impact on the policy mainly exists in two aspects: the direct influence on the policy and the indirect influence through affecting the $Q$ value.

*1) Tremulous Policy:* From the perspective of reward shaping, refer to (2) and (4), entropy can be regarded as a part of reward, and we can rewrite the right part of (2) as

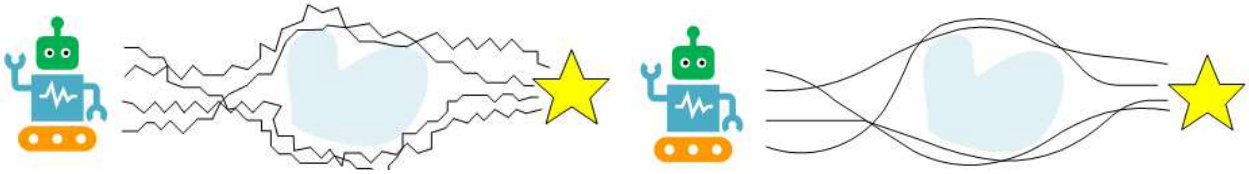
$$\underbrace{r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\alpha\mathcal{H}\left(\pi\left(\cdot\,|\mathbf{s}_{t+1}\right)\right)}_{\text{augmented reward}} + \gamma\mathbb{E}\left[Q\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right)\right]. \tag{6}$$

Compared to maximizing the original reward function $r(\mathbf{s}_t, \mathbf{a}_t)$, maximizing the augmented reward can easily deviate from the original objective and lead to excessive pursuit of entropy, resulting in unstable or tremulous control. Maximizing entropy may accompany excessive exploration of actions, whereas what we desire is trajectory-level exploration rather than action-level exploration. Overly encouraging action-level exploration makes continuous control for the robot unstable, thereby impacting policy optimization.

*2) Spiky Q-Function:* The entropy augmented reward can also be further decomposed and integrated into the $Q$ value as follows:

$$r(\mathbf{s}_t, \mathbf{a}_t) + \underbrace{\gamma\mathbb{E}\left[Q\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right) - \alpha \log \pi\left(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}\right)\right]}_{\text{Q value}}. \tag{7}$$

In standard RL, we typically need to compute the next-step $Q$ value, $\mathbb{E}_{\mathbf{a}_{t+1}\sim\pi}[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})]$. Due to computational considerations, this expectation is often approximated by randomly sampling an action to estimate the next-step $Q$ value. However, if we follow (7) to compute the target, low-probability actions



(a) Max-Entropy Reinforcement Learning (b) Max-Entropy with Stable Control

Fig. 1. Agent needs to explore to avoid the pond and reach the position of star. (a) Max-entropy RL methods encourage exploration to bypass the pond. (b) On the basis of maximizing entropy, additional stability is considered to ensure smoother actions. While encouraging exploration, one should consider the coherence of actions to achieve a more elegant long-term continuous control. The ideal target is that long-term entropy over the trajectory is encouraged, while short-term entropy over action is discouraged. By following this smooth and shortened trajectory, the agent can reach the target rapidly while consuming less energy.

can yield large negative values of $\log \pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})$. From a statistical perspective, this can lead to the $Q$-function exhibiting many sharp cusps, which are detrimental to both the iterative update of $Q$ values and gradient-based optimization of the $Q$-function.

The core issue lies in the difference between (6) and (7). The entropy term $\mathcal{H}(\pi(\cdot|\mathbf{s}_{t+1}))$ is bounded, whereas the log-probability term $\log \pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})$ is unbounded.

### B. Beta-Symmetric Kl Divergence

In RL, the $Q$ value provides key information about how to learn an optimal policy. Previous paper [1] has defined a target distribution with $Q$ value, which we name it as $Q$ distribution. For a given state $\mathbf{s}_t$, $Q$ distribution is defined as

$$\pi_Q(\cdot\,|\mathbf{s}_t) = \exp\left(\alpha^{-1} Q(\mathbf{s}_t, \cdot) - \log Z(\mathbf{s}_t)\right) \tag{8}$$

where $Z(\mathbf{s}_t)$ normalizes the $Q$ value and can be derived by summarizing the $Q$ value over all actions.

Inspired by the symmetric cross-entropy objective for robust classification [43], we give the definition of the beta-symmetric KL divergence.

*Definition 1 (Beta-Symmetric KL Divergence):* Given a state, with policy $\pi$ and $\pi_Q$, the beta-symmetric KL divergence is defined as

$$\mathrm{D}^{\beta}_{\mathrm{SKL}}\left(\pi \,\|\, \pi_Q\right) = \underbrace{\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q\right)}_{\text{reverse}} + \underbrace{\beta \mathrm{D}_{\mathrm{KL}}\left(\pi_Q \,\|\, \pi\right)}_{\text{forward}} \tag{9}$$

where $\mathrm{D}_{\mathrm{KL}}\left(q \,\|\, p\right) = \sum_x q(x)(\log q(x) - \log p(x))$ is the KL divergence and $\beta \geq 0$ is a temperature coefficient. The symmetry is reflected in the use of the KL function. When $\beta = 1$, it is completely symmetric in form and has a specific name "Jeffreys divergence." Since $\pi$ is the posterior distribution we aim to learn, the first term in the formula represents the reverse KL divergence, while the second term represents the forward KL divergence. Compared to the original Jeffreys divergence, using the beta-KL divergence as the objective introduces a tunable parameter $\beta$, which to some extent balances exploration and exploitation—thanks to the differing emphases of the reverse KL and forward KL divergences. While the maximum entropy regularization based on reverse KL divergence encourages moving as quickly as possible in a single step—allowing large changes—the beta-symmetric KL divergence additionally considers the coherence of movements. It limits large action changes, thereby improving stability.

Compared to the reverse KL divergence, the beta-symmetric KL divergence incorporates forward KL divergence, which has the mass-covering property [39].

*Remark 1 (Mass-Covering Property):* The beta-symmetric KL divergence aims to cover all modes of the prior distribution to minimize the divergence. In contrast, reverse KL divergence tends to focus on a single mode, potentially getting trapped there. This can lead to neglecting other significant beneficial modes of the true distribution.

The beta-symmetric KL divergence is a variant of $f$-divergence; it naturally has the properties of $f$-divergence, such as convexity.

*Proposition 1 (Convexity of the Beta-Symmetric KL Divergence):* The beta-symmetric KL divergence is convex in the pair of probability distributions $(p, q)$, i.e.,

$$\begin{aligned}&\mathrm{D}^{\beta}_{\mathrm{SKL}}\left(\gamma p_1 + (1-\gamma) p_2 \,\|\, \gamma q_1 + (1-\gamma) q_2\right)\\ &\leq \gamma \mathrm{D}^{\beta}_{\mathrm{SKL}}\left(p_1 \,\|\, q_1\right) + (1-\gamma)\mathrm{D}^{\beta}_{\mathrm{SKL}}\left(p_2 \,\|\, q_2\right)\end{aligned} \tag{10}$$

where $(p_1, q_1)$ and $(p_2, q_2)$ are two pairs of probability distributions and $0 \leq \gamma \leq 1$.

*Proof:* An intuitive understanding is that the KL divergence is convex, and then, the convex combination of two convex functions is still a convex function. To prove the convexity of the beta-symmetric KL divergence in the pair of probability distributions $(p, q)$, we will use the convexity of the KL divergence. More details are available in the Appendix.

Another point that needs to be noted is that one of these two KL divergences decreases, and the other one will also decrease.

*Proposition 2 (Consistent Minimization Direction):* Given two Gaussian distributions, learnable distribution $P$ and target distribution $Q$, initialize $P$ with a sufficiently small variance. When the reverse KL divergence decreases, the forward KL divergence will also decrease and vice versa.

*Proof:* The relationship between reverse KL divergence and forward KL divergence in optimization has been ignored in the previous work. We demonstrate that assuming that distribution $P$ initially has low variance, as is typical in neural networks, a decrease in one of these two KL divergences also results in a decrease in the other, indicating consistent minimization direction. For more details, please see the Appendix.

Aside from these properties, the beta-symmetric KL divergence should have a faster constant convergence rate than the reverse KL divergence in constant order when $\beta > 0$. This can be verified if we assume that the two distributions have consistent variances.[1] In this article, we will not focus too much on this aspect.

### C. Policy Improvement With Beta-Symmetric Kl Divergence

Let $\pi_{\text{old}} \in \Pi$ and $Q^{\pi_{\text{old}}}$ and $V^{\pi_{\text{old}}}$ be the corresponding soft state–action value and soft state value, and the policy objective function can be defined as

$$J_\pi^{\text{SKL}}(\pi(\cdot|\mathbf{s}_t)) = \text{D}_{\text{SKL}}^{\beta}\left(\pi(\cdot|\mathbf{s}_t) \parallel \pi_Q(\cdot|\mathbf{s}_t)\right).$$

We want to discuss the policy evaluation and policy improvement with this objective. Policy evaluation has been discussed in the soft policy iteration theorem [2].

*Lemma 1 (Policy Evaluation):* Consider the Bellman backup operator $\mathcal{T}^\pi$ in (2) with modified state value function in (5) and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. Then, the sequence $Q^k$ will converge to the soft $Q$ value of $\pi$ as $k \to \infty$.

*Proof:* Define the entropy augmented reward as

$$r_\pi(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\alpha\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[\mathcal{H}\left(\pi(\cdot|\mathbf{s}_{t+1})\right)\right] \tag{11}$$

and rewrite the update rule as

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_\pi(\mathbf{s}_t, \mathbf{a}_t) + \gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p, \mathbf{a}_{t+1}\sim\pi}\left[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})\right] \tag{12}$$

and apply the standard convergence results for policy evaluation [44]. The assumption $|\mathcal{A}| < \infty$ is required to guarantee that the entropy and cross-entropy augmented reward is bounded.

In the policy improvement step, for each state, we update the policy according to

$$\pi' = \arg\min_{\pi\in\Pi} J_\pi^{\text{SKL}}(\pi(\cdot|\mathbf{s}_t)). \tag{13}$$

In the policy improvement step, for each state, we update the policy according to (13); then, we can drive a better policy. Policy improvement consists of two steps: first, ensuring the improvement of $Q$ value and then making the policy similar to the $Q$ distribution. If the $Q$ value converges, the $Q$ distribution will converge and the policy will also converge. The process of policy improvement is formalized in Theorem 1.

*Theorem 1 (Policy Improvement With Beta-Symmetric KL Divergence):* Let $\pi_{\text{old}} \in \Pi$ and let $\pi_{\text{new}}$ be the optimizer of the minimization problem defined in (13). Then, $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.

*Proof:* We change the objective function of soft policy iteration, from the reverse KL divergence to the beta-symmetric KL divergence. According to Propositions 1 and 2, minimizing the beta-symmetric KL divergence can lead to a decrease in reverse KL divergence. We thus derive a new policy, which guarantees that

$$\mathbb{E}_{\mathbf{a}_t\sim\pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t) - \alpha\log\pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t)\right] \geq V^{\pi_{\text{old}}}(\mathbf{s}_t). \tag{14}$$

[1]A detailed proof could be seen here: https://stats.stackexchange.com/questions/357798/rate-of-converge-of-kl-divergence-to-posterior

Policy improvement is guaranteed by minimizing the beta-symmetric KL divergence, and more details are available in the Appendix.

In theory, the policy iteration alternates between the soft policy evaluation and the soft policy improvement steps, and it will provably converge to the optimal policy.

### D. Policy Optimization and Q Value Update

We show the practical computation of policy gradient and target $Q$ value, which influences the policy optimization and value update.

*1) Policy Optimization:* Consider the neural network parameterized function, such as $\theta$ parameterized $Q_\theta$ function and $\phi$ parameterized policy $\pi_\phi$, and new policy is derived by minimizing the $J_\pi^{\text{SKL}}$ objective

$$\begin{aligned}
J_\pi(\phi) &= \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[\text{D}_{\text{SKL}}^{\beta}\left(\pi_\phi(\cdot|\mathbf{s}_t) \parallel \pi_Q(\cdot|\mathbf{s}_t, \pi)\right)\right] \\
&= \mathbb{E}_{\mathbf{s}_t}\left[\text{D}_{\text{SKL}}^{\beta}\left(\pi_\phi(\cdot|\mathbf{s}_t) \parallel \exp\left(\alpha^{-1}Q_\theta(\mathbf{s}_t, \cdot) - \log Z(\mathbf{s}_t)\right)\right)\right] \\
&= \mathbb{E}_{\mathbf{s}_t}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi_\phi(\cdot|\mathbf{s}_t)}\left[\log\pi_\phi(\mathbf{a}_t|\mathbf{s}_t) - \alpha^{-1}Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + Z(\mathbf{s}_t)\right]\right] \\
&+\beta\mathbb{E}_{\mathbf{s}_t}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi_Q(\cdot|\mathbf{s}_t)}\left[-\log\pi_\phi(\mathbf{a}_t|\mathbf{s}_t) + \alpha^{-1}Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - Z(\mathbf{s}_t)\right]\right]
\end{aligned}$$

where $Z(\mathbf{s}_t) = \sum_{a\in\mathcal{A}}\exp Q_\theta(\mathbf{s}_t, a)$ and $\mathcal{D}$ is a replay buffer. $\log Z(\mathbf{s}_t)$ is an additive constant and has no relation to parameter $\phi$, so, in fact, we also do not need to include it for gradient computation.

We can sample actions from $\pi_\phi$ to get an unbiased estimation of $\nabla_\phi J_\pi(\phi)$. Here, the reparameterization trick is used to ensure a lower variance estimation. It is that $\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$, $\epsilon_t \sim \mathcal{N}(0, 1)$; the detailed form of the function $f_\phi$ is $f_\phi(\epsilon_t; \mathbf{s}_t) = \mu_\phi(\mathbf{s}_t) + \epsilon\sigma_\phi(\mathbf{s}_t)$, where $\mu_\phi(\mathbf{s}_t)$ and $\sigma_\phi(\mathbf{s}_t)$ are the outputs of the policy network.

An unbiased policy gradient estimation can be derived by

$$\begin{aligned}
\hat{\nabla}_\phi J_\pi(\phi) &= \nabla_\phi\left(\log\pi_\phi(a|\mathbf{s}_t) - \alpha^{-1}Q_\theta(\mathbf{s}_t, a)\right) \\
&\quad - \beta\frac{\pi_Q(\mathbf{a}_t|\mathbf{s}_t)}{\pi(\mathbf{a}_t|\mathbf{s}_t)}\nabla_\phi\log\pi_\phi(a|\mathbf{s}_t) \qquad (15) \\
&= \nabla_\phi\log\pi_\phi(a|\mathbf{s}_t) + \nabla_a\log\pi_\phi(a|\mathbf{s}_t)\nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \\
&\quad - \alpha^{-1}\nabla_a Q_\theta(\mathbf{s}_t, a)\nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \\
&\quad - \beta\frac{\pi_Q(\mathbf{a}_t|\mathbf{s}_t)}{\pi(\mathbf{a}_t|\mathbf{s}_t)}\nabla_\phi\log\pi_\phi(a|\mathbf{s}_t) \qquad (16)
\end{aligned}$$

where $\mathbf{s}_t$ is sampled from the replay buffer $\mathcal{D}$ and $\pi_Q(\mathbf{a}_t|\mathbf{s}_t)$ can be approximated by $\exp(\alpha^{-1}Q(\mathbf{s}_t, \mathbf{a}_t) - \log V(\mathbf{s}_t))$.

*Remark 2 (Stable Policy Optimization):* We introduce $-\nabla_\phi\log\pi_\phi(a|\mathbf{s}_t)$ into the policy gradient by beta-symmetric KL divergence, which encourages the negative log probability to be smaller, achieving policy optimization in a smooth manner

Ideally, $\pi_Q(\mathbf{a}_t|\mathbf{s}_t) = \pi(\mathbf{a}_t|\mathbf{s}_t)$, to prohibit large value influences the value of $\pi_Q(\mathbf{a}_t|\mathbf{s}_t)/\pi(\mathbf{a}_t|\mathbf{s}_t)$, and the output is clipped to $[0.8, 1.2]$.

*2) Q Value Update:* In practical computation, since the argmin operator is only theoretical, we consider minimizing the beta-symmetric KL divergence objective to obtain a new policy. This change actually affects the iteration of $Q$ value. We propose surrogate function optimization, altering the calculation of the $Q$-function target and decoupling policy optimization from $Q$ value update. This modification allows

the influence of entropy on policy optimization and $Q$ value update to be controlled separately. Specifically, we use $\pi_Q^\alpha$ as the target of policy optimization. By rewriting (8), we get

$$\pi_Q^\alpha(\cdot\,|\mathbf{s}_t) = \exp\left(\alpha^{-1}Q(\mathbf{s}_t,\cdot) - \log Z(\mathbf{s}_t)\right).$$

There exists a distribution $\pi_Q^{\alpha'}$, and $\alpha \le \alpha'$. When $Q(s,a) > 0$, we have

$$\begin{aligned}&\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^\alpha\right) \ge \mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^{\alpha'}\right)\\ &\text{where } \pi_Q(\cdot\,|\mathbf{s}_t) = \exp\left(\alpha'^{-1}Q(\mathbf{s}_t,\cdot) - \log Z(\mathbf{s}_t)\right).\end{aligned}$$

This inequality can be derived by expanding the KL divergence. The condition of $Q(s,a) > 0$ is easy to be satisfied. Thus, $\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^\alpha\right)$ is the upper bound of $\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^{\alpha'}\right)$, and $\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^{\alpha'}\right)$ decreases when $\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^\alpha\right)$ is minimized. Referring to the proof of policy improvement, if $\mathrm{D}_{\mathrm{KL}}\left(\pi \,\|\, \pi_Q^{\alpha'}\right)$ decreases, we can guarantee the improvement of modified $Q$ value. Referring to (14), we have

$$\mathbb{E}_{\mathbf{a}_t\sim\pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(\mathbf{s}_t,\mathbf{a}_t) - \alpha'\log\pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t)\right] \ge V^{\pi_{\text{old}}}(\mathbf{s}_t).$$

Similarly, referring to (12), a modified policy evaluation is derived as follows:

$$\begin{aligned}&Q(\mathbf{s}_t,\mathbf{a}_t) \leftarrow r(\mathbf{s}_t,\mathbf{a}_t)\\ &+\gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p,\mathbf{a}_{t+1}\sim\pi}\left[Q(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \alpha'\log\pi(\mathbf{a}_{t+1},\mathbf{s}_{t+1})\right].\end{aligned}$$

*Remark 3 (Smooth Q Value):* We introduce $\alpha'$ to decouple the $Q$ value update and max-entropy reward, which were connected with $\alpha$ in the previous method [15]. As we can adjust the value of $\alpha'$, then a large $\alpha$ will not directly influence the update of $Q$ value. Thus, the update of $Q$ value can be flexible and smooth with a small $\alpha'$.

In practice, the target $Q$ value is estimated with the sampled next state and action

$$\begin{aligned}\hat{Q}(\mathbf{s}_t,\mathbf{a}_t) &= r(\mathbf{s}_t,\mathbf{a}_t)\\ &+\gamma Q(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \gamma\alpha'\log\pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}).\end{aligned} \tag{17}$$

To learn the $Q$-function, we can minimize the mse loss between the parameterized $Q$ value and the estimated target $Q$ value

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \hat{Q}(\mathbf{s}_t,\mathbf{a}_t)\right)^2\right]. \tag{18}$$

The gradient of the $Q$ value with respect to $\theta$ is

$$\begin{aligned}\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t,\mathbf{s}_t)\big(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - r(\mathbf{s}_t,\mathbf{a}_t)\\ -\gamma Q(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) + \gamma\alpha'\log\pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\big).\end{aligned} \tag{19}$$

Up to this point, we have completed the derivation and calculation of the policy optimization and $Q$ value update. Next, we will connect these key conclusions to form a concrete method.

**Algorithm 1** MeSO

**Require:** Initial parameters $\theta_1, \theta_2$ of the $Q$ value function and $\phi$ of the target policy $\pi_T$.

1 Set the values of $\alpha$, $\alpha'$, and $\beta$. $\alpha$ and $\alpha'$ are automatically tuned, while $\beta$ remains fixed.
2 $\breve{\theta}_1 \leftarrow \theta_1$, $\breve{\theta}_2 \leftarrow \theta_2, \mathcal{D} \leftarrow \emptyset$ {Initialize target network weights and replay buffer}
3 **for** each iteration **do**
4  **for** each environment step **do**
5  $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$ {Sample action from policy}
6  $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$ {Sample state from the environment}
7  $\mathcal{D} \leftarrow \mathcal{D}\cup\{(\mathbf{s}_t,\mathbf{a}_t,R(\mathbf{s}_t,\mathbf{a}_t),\mathbf{s}_{t+1})\}$ {Store the transition in the replay buffer}
8  **end for**
9  **for** each training step **do**
10  sample batch transition $(\mathbf{s}_t,\mathbf{a}_t,R(\mathbf{s}_t,\mathbf{a}_t),\mathbf{s}_{t+1})$ from the buffer
11  update $\phi$ with $\hat{\nabla}_\phi J_\pi(\phi)$ {Policy parameter update with Eq. (15)}
12  compute target $\hat{Q}_i(\mathbf{s}_t,\mathbf{a}_t)$ for $i \in 1,2$ {Target $Q$ value update with Eq. (17)}
13  update $\theta_i$ with $\hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in 1,2$ {Q parameter update with Eq. (19)}
14  Automatically tune $\alpha$ according to the setting in paper [15], and set $\alpha'$ = (1−training step/total step)$\cdot$ $\alpha'\cdot\alpha$.
15  $\breve{\theta}_1 \leftarrow \tau\theta_1 + (1-\tau)\breve{\theta}_1, \breve{\theta}_2 \leftarrow \tau\theta_2 + (1-\tau)\breve{\theta}_2$ {Update target networks}
16  **end for**
17 **end for**

**Ensure:** $\theta_1$, $\theta_2$, $\phi$ {Optimized parameters}

### E. Practical Algorithm

In comparison to SAC, MeSO (refer to Algorithm 1) alters the policy objective for the computation of policy gradient (line 11) and the target $Q$ value for value update (line 12).

The overall flow of the algorithm is roughly as follows.

1) In parameter initialization, Line 1, we initialize the network weight for policy and $Q$-function.
2) In environment interaction, Lines 3–8, we use policy to interact with environment.
3) In parameter update, Lines 9–14, we update the policy parameter by minimizing the beta-symmetric KL divergence objective function and update the parameter of $Q$-function by minimizing the prediction error of $Q$ value. In Line 10, we sample transitions uniformly from the buffer. Line 14, the temperature coefficient $\alpha$ is updated by minimizing the following objective:

$$\mathcal{L}_\alpha = -\mathbb{E}_{s_t,a_t\sim\mathcal{D}}\left[\alpha\log\pi_\theta(a_t|s_t) - \alpha H_{\text{target}}\right].$$

For continuous action spaces, the target entropy $H_{\text{target}}$ is the number of dimensions in the action space.

## V. Experiments

To demonstrate the effectiveness of our algorithm, we initially compared the results based on average episode returns

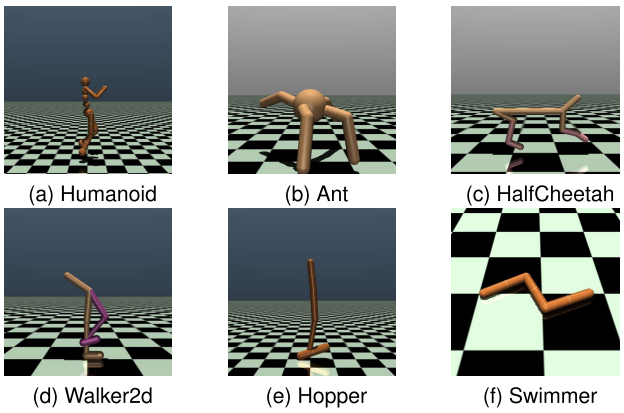



Fig. 2. Six environments of Mujoco are demonstrated. In these tasks, we need to control objects to move forward with different poses, such as (a), (b), and (d) walking, (c) running, (e) jumping, and (f) swimming.

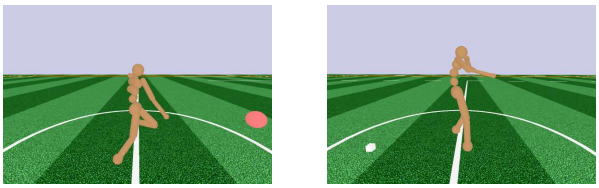



Fig. 3. Two environments of Roboschool are demonstrated. (a) Task for the agent is to chase the red flag. (b) Agent will be attacked by the white cube when chasing the red flag.

TABLE I
DETAILS OF ENVIRONMENTS USED IN THIS ARTICLE

| Mujoco environment | state dim | action dim | episode length |
|---|---|---|---|
| Humanoid-v2 | 376 | 17 | 1000 |
| Ant-v2 | 111 | 8 | 1000 |
| HalfCheetah-v2 | 17 | 6 | 1000 |
| Walker2d-v2 | 17 | 6 | 1000 |
| Hopper-v2 | 11 | 3 | 1000 |
| Swimmer-v2 | 8 | 2 | 1000 |
| **Roboschool environment** | **state dim** | **action Dim** | **episode length** |
| Flagrun-v1 | 44 | 17 | 1000 |
| FlagrunHarder-v1 | 44 | 17 | 1000 |

with other baseline algorithms. Since our algorithm involves not only stable update of $Q$ value but also stable update of the policy, we conduct ablation experiments concerning our algorithm itself. We conduct experiments on Mujoco and Roboschool physical simulations. The algorithms used in the comparative experiments share the same neural network architecture. The results are averaged over six random seeds, and the time steps for all environments are set to one million steps.

### A. Experiment Setting and Baseline Algorithms

*1) Environments:* We do experiments on the Mujoco platform and Roboschool platform. The detail information of environments is provided in Table I. The task of each environment is demonstrated in Figs. 2 and 3. The hyperparameters are in the Appendix.

*2) Baselines:* We compared the four most well-known baseline algorithms in RL, which have had a significant impact on the field.

1) PPO [34] has been popular used in many discrete control fields. It ensures policy improvement with monotonic policy improvement theorem [31] under small policy change. In continuous control tasks, encouraging entropy hurts its performance due to large policy change.
2) P3O [41] allows more policy change than the PPO algorithm by considering more off-policy data but still suffers from the entropy control issue.
3) TD3 [25] aims to address the issue of overestimation of $Q$ value in continuous action spaces. By improving policy learning through better $Q$ value estimation, it has achieved excellent experimental results.
4) SAC [2] updates the theoretical framework of the policy gradient algorithm and proposes the soft policy iteration theorem, achieving excellent experimental results while ensuring policy improvement.
5) DSAC-v2 [45] updates DSAC [46] with three refinements: expected value substitution, twin value distribution learning, and variance-based critic gradient adjustment.

### B. Performance of Policy Optimization on Mujoco

We consider the mainstream stochastic policy gradient algorithms, namely, the PPO algorithm and SAC algorithm, as baseline algorithms, also include the deterministic policy algorithm TD3, and also include a more recent algorithm, P3O. We believe that the success of an algorithm lies in the stable control of entropy, where reasonable entropy affects both exploration and policy improvement—both crucial aspects of policy learning. We also include DSAC-v2 as a baseline to provide readers with a clear understanding of the latest advances.

As shown in Fig. 4, in the Humanoid environment, our method can achieve a reward of 5000 around 0.4 million time steps, which is approximately twice as fast as the SAC algorithm. DSAC-v2 suffers from a little instability.

The TD3 algorithm performs moderately in the Humanoid environment due to a lack of exploration. The PPO algorithm struggles to achieve good results as it uses a fixed entropy. In the HalfCheetah environment, DSAC-v2 achieves the best performance. The advantages of our method are also evident, as it achieves better results while being less sensitive to random seeds. Moreover, Meso is not in conflict with the DSAC-v2 algorithm, and combining the two could potentially lead to further performance improvements. In the Swimmer environment, our method gradually gains an advantage as learning continues because the reward in the Swimmer environment is delayed. It is a long-term reward; more interaction data are required for the algorithm to adequately learn the $Q$ value function. In the remaining environments, our algorithm performs slightly better. In general, it can be observed that our method achieves more stable results while achieving better performance.

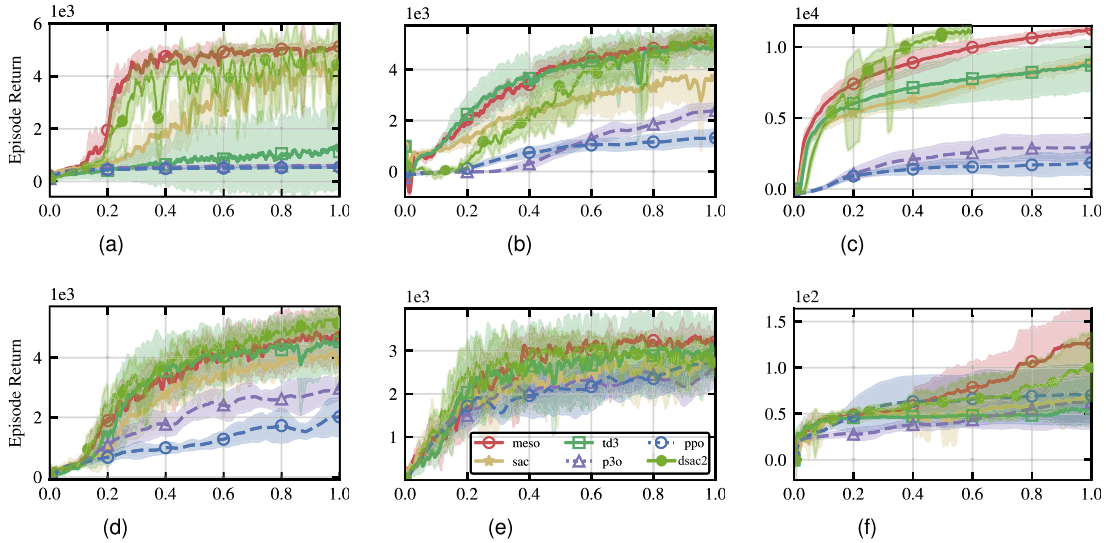



Fig. 4. General result compared with SAC and other baseline algorithms on Mujoco. The $x$-axis represents time steps, and the $y$-axis represents episode return. (a) Humanoid-v2. (b) Ant-v2. (c) HalfCheetah-v2. (d) Walker2d-v2. (e) Hopper-v2. (f) Swimmer-v2.

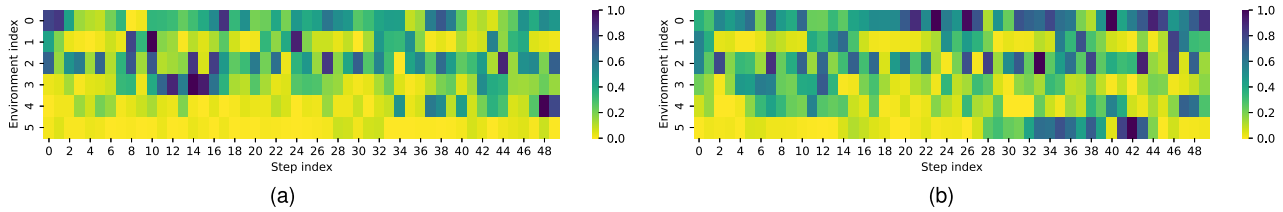



Fig. 5. Normalized heatmap of consecutive action difference. The environment indices 0–5 represent the sequence of environments: Humanoid, Ant, HalfCheetah, Walker2d, Hopper, and Swimmer. The $x$-axis represents the consecutive time steps of 50, and the color intensity indicates the magnitude of the action difference, with 1 indicating the largest difference. The differences are first normalized within each environment setting. (a) MeSO. (b) SAC.

### C. Policy Stability Comparison Between MeSO and SAC

We also compare the differences in consecutive actions to verify whether the motion of our method is smooth. This comparison allows us to explore whether the algorithm encourages action-level exploration.

As shown in Fig. 5, we compare the action differences of 50× steps in six environments. Some results are easy to distinguish, such as comparing indices 5 of Fig. 5(a) and (b), which correspond to the Swimmer environment in Fig. 4. Some comparisons of the results might not be very obvious and require careful discernment, such as the comparison of the results in the HalfCheetah environment. Overall, the results of our method are relatively lighter in color, indicating a smooth change in consecutive actions. This suggests that considering the stability of the strategy has had a positive impact on the coherence of the actions and that our algorithm better controls exploration at the action level.

We also evaluated the final policies trained by SAC and MeSO, recording the control reward separately in the Mujoco environments to provide a standard quantitative metric. Each algorithm generated six policies, and each policy was run ten times; the control reward is the average over these ten trials. The table presents a comparison between MeSO and SAC. For each of the six environments, the value shown in Table II is the maximum control reward among the six averaged control rewards.

TABLE II
AVERAGE CONTROL REWARDS OF MESO AND SAC

| Environment | MeSO | SAC |
|---|---|---|
| Humanoid | -2.89 | -70.37 |
| Ant | -358.07 | -639.56 |
| HalfCheetah | -348.82 | -356.42 |
| Walker2d | -1.77 | -2.05 |
| Hopper | -0.24 | -0.61 |
| Swimmer | -0.04 | -0.02 |

### D. Exploration Change During Training

Our approach claims to achieve stable optimization under the maximum entropy framework. It encourages greater entropy during exploration to promote trajectory-level exploration while considering smaller entropy (with extra parameters $\beta$ and $\alpha'$) in policy improvement and $Q$ value update to stabilize these processes. Therefore, we primarily compared the changes in entropy during the training process of the MeSO and SAC algorithms.

In Fig. 6, we present the average standard deviation of the Gaussian policy. Unlike the direct comparison of continuous actions depicted in Fig. 5, this figure empirically estimates the average entropy of the policy for all states. This entropy serves as an indicator of exploration at the trajectory level. We observe that MeSO displays a larger standard deviation

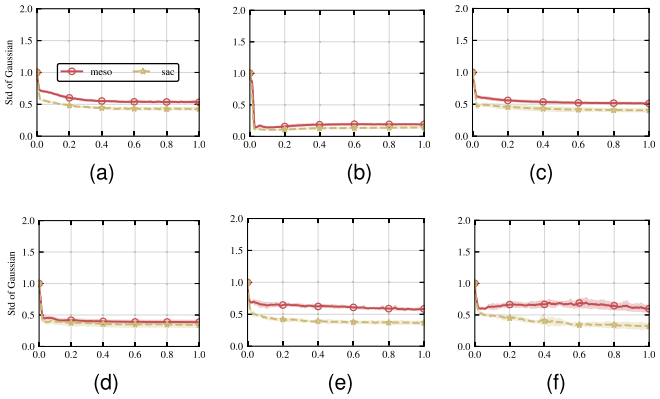



Fig. 6. Entropy variation of SAC and MeSO, where the $x$-axis represents time steps and the $y$-axis represents empirical estimation of standard deviation of Gaussian policy over all trajectories. (a) Humanoid-v2. (b) Ant-v2. (c) HalfCheetah-v2. (d) Walker2d-v2. (e) Hopper-v2. (f) Swimmer-v2.

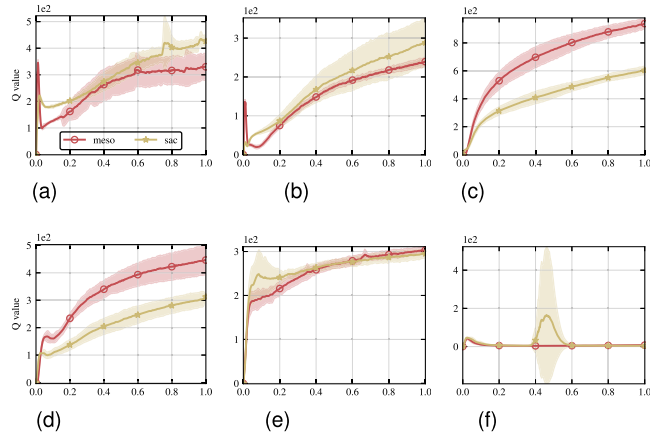



Fig. 7. Learning curves of $Q$ value in the six environments, the x-axis represents million time steps, and the y-axis represents empirical estimation of $Q$ value. (a) Humanoid-v2. (b) Ant-v2. (c) HalfCheetah-v2. (d) Walker2d-v2. (e) Hopper-v2. (f) Swimmer-v2.

compared to SAC, resulting in superior trajectory-level performance. MeSO controls the influence of entropy on policy optimization and $Q$ value update while encouraging policy-level exploration. This approach promotes diversity in policy exploration and leads to improved outcomes.

### E. Q Value Difference Between MeSO and SAC

Generally, a higher $Q$ value suggests that a better policy can be attained, as $Q$ values provide valuable gradient information for policy optimization. However, exceptions exist. $Q$ values can sometimes be overestimated, which can impede policy optimization by supplying inaccurate information for the optimization process.

As shown in Fig. 7(a), (b), and (d), the corresponding $Q$ value of MeSO is better learned than that of SAC, resulting in better outcomes as shown in Fig. 4. However, this view does not hold when there is significant overestimation in the $Q$ value. For instance, in Fig. 7(b), (e), and (f), a large $Q$ value does not guarantee stable optimization of policy, as these values are accumulated from the overestimation of the target $Q$ value. They appear larger but are not actually

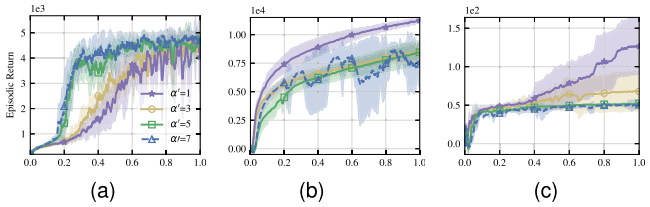



Fig. 8. How policy learning is indirectly influenced by entropy through the target $Q$ value? We tested the effect of different $\alpha'$'s in three environments. (a) Humanoid-v2. (b) HalfCheetah-v2. (c) Swimmer-v2.

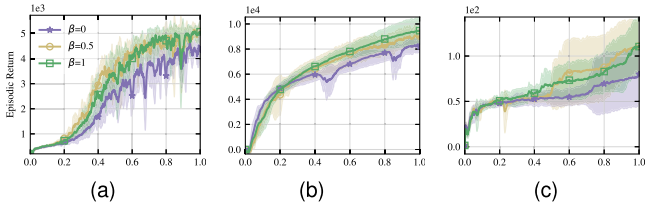



Fig. 9. How policy learning is directly influenced by entropy? We tested the effect of different $\beta$'s in three environments. (a) Humanoid-v2. (b) HalfCheetah-v2. (c) Swimmer-v2.

beneficial for policy optimization. MeSO mitigates the impact of overestimation on value learning by annealing the entropy term in the $Q$ value calculation, thereby accelerating policy learning.

### F. Result Analysis

As shown in Fig. 4, our algorithm performs relatively well in the Humanoid, HalfCheetah, and Swimmer environments. We are trying to analyze which hyperparameter has a decisive impact on the algorithm's performance. Additionally, these three environments range from complex to simple, which provides some reference value for our parameter settings.

*1) Entropy on Q Value Update:* Max-entropy methods incorporate log probabilities into $Q$ value iterations, which indeed help with exploration. We are concerned about the impact of the hyperparameter $\alpha'$, which controls the contribution of entropy to the $Q$ value, on policy learning.

As shown in Fig. 8(a), large entropy (large $\alpha'$) involving the target $Q$ value clearly encourages exploration, leading to better policy optimization. However, this is not always the case. In some simpler environments, excessive entropy can cause policy instability, as seen in the results of Fig. 8(b) and (c). We can conclude that max-entropy methods are suitable for exploration tasks, larger parameters can be beneficial in the early stages of training, and the initial value of $\alpha'$ should be large when the environment is complex. As training progresses and the policy approaches convergence, smaller parameters should be considered. In our experiment, we linearly attenuate $\alpha'$ during the training process and achieve good results.

*2) Entropy on Policy Optimization:* Compared to the impact of entropy on $Q$ value, entropy has a more direct influence on policy optimization. We aim to study the effect of the parameter $\beta$ in the objective function on policy optimization.

As shown in Fig. 9, compared to not using our proposed objective function (i.e., $\beta = 0$), appropriately using our method can achieve better results. However, excessively large value of

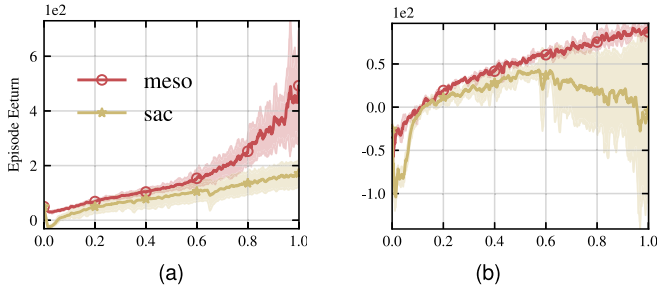



Fig. 10. General results compared with SAC in two more difficult control tasks on Roboschool. (a) HumanoidFlagrun. (b) HumanoidFlagrunHarder.

$\beta$ also does not improve policy optimization; instead, it can lead to the policy lacking exploration and getting stuck in local suboptimal solutions. As shown in Fig. 9(c), in the Swimmer environment, $\beta = 0.5$ achieves the best results, while not using or using excessively large $\beta$ value makes the results unstable in the later stages. In general, the use of $\beta$ does reduce the impact of entropy on policy optimization. In specific tasks, a value of $\beta$ between 0.5 and 1 can achieve better results.

### G. Compared With SAC Algorithm on Roboschool

To verify the robustness of our algorithm in dynamic environments, we tested it on more challenging control tasks.

We compare our algorithm with SAC on two more difficult environments on the Roboschool platform. In the HumanoidFlagrun environment, the robot must run toward a randomly generated flag. In the more challenging HumanoidFlagrunHarder environment, the robot is constantly bombarded by white cubes. As shown in Fig. 10, our algorithm exhibits a clear trend of performance improvement in the HumanoidFlagrun environment. In the more challenging HumanoidFlagrunHarder environment, our method demonstrates stability. In this environment, learning $Q$ value becomes more challenging. The soft $Q$ value of SAC degrades to the negative log probability. As a result, the SAC algorithm essentially maximizes entropy, losing the ability to maximize the environment reward and resulting in deteriorating performance.

### H. Limitation

While our approach is guided by certain theoretical principles, in its implementation, we propose several assumptions. One crucial assumption pertains to smoothness, requiring the stability of $Q$ value learning. A failure in learning the $Q$ values could detrimentally impact policy learning, triggering a chain reaction akin to the error propagation observed in $Q$ value updates during Bellman iteration. On the other hand, we have not thoroughly explored the relationships between different stability approaches. Stability in RL involves a wide range of factors, and some methods proposed in the literature appear to be rather tricky [47] and direct [48]. Our focus is primarily on trajectory stability, which also brings some benefits to the stability of the optimization process. However, the stability of the optimization process may not provide additional gains when combined with other stability methods.

## VI. Conclusion

In the long-term policy optimization process, stability is a significant concern. We have identified instability issues within the maximum entropy framework, namely, the tremendous policy and spiky $Q$-function. The key to addressing these issues lies in controlling the impact of entropy on policy optimization. In this article, we attempt to stabilize the impact of entropy on policy optimization by beta-symmetric KL divergence. We theoretically validate the feasibility of this method and experimentally verify its effectiveness. Based on beta-symmetric KL divergence, we have updated the soft policy iteration theorem and derived a new objective function. This new objective function encourages exploration while also mitigating the instability in the maximum entropy RL framework. We conducted experiments on the Mujoco and Roboschool simulation platforms, and the results demonstrate that the proposed method not only enhances stability but also facilitates policy learning under the maximum entropy RL framework.

In the future, we will explore stable optimization with deterministic policy. While the maximum entropy framework ensures some exploration capability by using a stochastic policy, balancing exploration and exploitation remains challenging. We believe that deterministic policy might offer an ideal solution, though achieving stable optimization with deterministic policies also presents its own set of challenges.

## Appendix
## Proof of Proposition 1

Proposition 1 (Convexity of the Beta-Symmetric KL Divergence). The beta-symmetric KL divergence is convex in the pair of probability distributions $(p, q)$, i.e.,

$$\begin{aligned}&\mathrm{D}_{\mathrm{SKL}}^{\beta}\left(\gamma p_1+(1-\gamma)\,p_2 \,\|\, \tau q_1+(1-\gamma)\,q_2\right)\\&\leq \gamma \mathrm{D}_{\mathrm{SKL}}^{\beta}\left(p_1 \,\|\, q_1\right)+(1-\gamma)\,\mathrm{D}_{\mathrm{SKL}}^{\beta}\left(p_2 \,\|\, q_2\right)\end{aligned} \tag{20}$$

where $(p_1, q_1)$ and $(p_2, q_2)$ are two pairs of probability distributions and $0 \leq \gamma \leq 1$.

*Proof:* Consider two pairs of probability distributions $(p_1, q_1)$ and $(p_2, q_2)$. We want to prove that for $0 \leq \gamma \leq 1$, the following inequality holds:

$$\begin{aligned}&\mathrm{SKL}\left(\gamma p_1+(1-\gamma)\,p_2, \gamma q_1+(1-\gamma)\,q_2\right)\\&\leq \gamma \mathrm{SKL}\left(p_1, q_1\right)+(1-\gamma)\,\mathrm{SKL}\left(p_2, q_2\right).\end{aligned} \tag{21}$$

Substituting the definition of SKL and KL into the inequality, we have

$$\begin{aligned}&\mathrm{SKL}\left(\gamma p_1+(1-\gamma)\,p_2, \gamma q_1+(1-\gamma)\,q_2\right)\\&=\mathrm{KL}\left(\gamma p_1+(1-\gamma)\,p_2 \| \gamma q_1+(1-\gamma)\,q_2\right)\\&\quad+\beta\left[\mathrm{KL}\left(\gamma q_1+(1-\gamma)\,q_2 \| \gamma p_1+(1-\gamma)\,p_2\right)\right]\\&\leq \gamma \mathrm{KL}\left(p_1 \| q_1\right)+(1-\gamma)\,\mathrm{KL}\left(p_2 \| q_2\right)\\&\quad+\beta\left[\gamma \mathrm{KL}\left(q_1 \| p_1\right)+(1-\gamma)\,\mathrm{KL}\left(q_2 \| p_2\right)\right]\\&=\gamma\left[\mathrm{KL}\left(p_1 \| q_1\right)+\beta \mathrm{KL}\left(q_1 \| p_1\right)\right]\\&\quad+(1-\gamma)\left[\mathrm{KL}\left(p_2 \| q_2\right)+\beta \mathrm{KL}\left(q_2 \| p_2\right)\right]\\&=\gamma \mathrm{SKL}\left(p_1, q_1\right)+(1-\gamma)\,\mathrm{SKL}\left(p_2, q_2\right).\end{aligned} \tag{22}$$

Equation (22) uses the convexity of the KL divergence.[2] Thus, we complete the proof, demonstrating that the beta-symmetric KL divergence is convex in the pair of probability distributions $(p, q)$.

### A. Proof of Proposition 2

*Proposition 2 (Consistent Minimization Direction):* Given two Gaussian distributions, learnable distribution $P$ and target distribution $Q$, initialize $P$ with a sufficiently small variance. When the reverse KL divergence decreases, the forward KL divergence will also decrease and vice versa.

*Proof:* Let $P$ and $Q$ be two univariate Gaussian distributions with parameters: $P \sim \mathcal{N}(\mu_p, \sigma_p^2)$ and $Q \sim \mathcal{N}(\mu_q, \sigma_q^2)$. The value[3] of KL-divergence from $P$ to $Q$ is given by

$$\mathrm{KL}\left(P \parallel Q\right) = \log \frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2}{2\sigma_q^2} + \frac{\left(\mu_q - \mu_p\right)^2}{2\sigma_q^2} - \frac{1}{2}.$$

Similarly, the KL-divergence from $Q$ to $P$ is given by

$$\mathrm{KL}\left(Q \parallel P\right) = \log \frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2}{2\sigma_p^2} + \frac{\left(\mu_q - \mu_p\right)^2}{2\sigma_p^2} - \frac{1}{2}.$$

Let distribution $Q$ be the target distribution, and then, we have

$$\frac{\partial \mathrm{KL}\left(P \| Q\right)}{\partial \mu_p} = \frac{1}{\sigma_q^2}\left(\mu_q - \mu_p\right) \tag{23}$$

$$\frac{\partial \mathrm{KL}\left(Q \| P\right)}{\partial \mu_p} = \frac{1}{\sigma_p^2}\left(\mu_q - \mu_p\right). \tag{24}$$

It can be seen that the partial derivatives of these two KL divergences with respect to $\mu_p$ are in the same direction. Also

$$\frac{\partial \mathrm{KL}\left(P \| Q\right)}{\partial \sigma_p} = -\frac{1}{\sigma_p} + \frac{\sigma_p}{\sigma_q^2} = \frac{\sigma_p^2 - \sigma_q^2}{\sigma_p \sigma_q^2} \tag{25}$$

$$\frac{\partial \mathrm{KL}\left(Q \| P\right)}{\partial \sigma_p} = \frac{1}{\sigma_p} - \frac{\sigma_q^2 + \left(\mu_q - \mu_p\right)^2}{\sigma_p^3} \tag{26}$$

$$= \frac{\sigma_p^2 - \sigma_q^2 - \left(\mu_q - \mu_p\right)^2}{\sigma_p^3}. \tag{27}$$

It can be seen that if $\sigma_p^2 < \sigma_q^2$, the partial derivatives of these two KL divergences with respect to $\sigma_p$ are in the same direction. In practice, this assumption is usually easy to satisfy because we do not need to make any assumptions about the unknown distribution; we simply optimize the distribution $P$ with a sufficiently small variance.

### B. Proof of Theorem 1

*Theorem 1 (Policy Improvement With Beta-Symmetric KL Divergence):* Let $\pi_{\text{old}} \in \Pi$ and let $\pi_{\text{new}}$ be the optimizer of the minimization problem defined in (13). Then, $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.

*Proof:* We borrow a lot from the proof of soft policy improvement theorem [2]. In this article, the main idea is to modify the reverse KL divergence to the beta-symmetric KL divergence to ensure the method's rationality. The $\pi_{\text{new}}$ can be derived by

$$\begin{aligned}\pi_{\text{new}}\left(\cdot \mid \mathbf{s}_t\right) &= \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{SKL}}^{\beta}\left(\pi'\left(\cdot|\mathbf{s}_t\right) \parallel \pi_Q\left(\cdot|\mathbf{s}_t\right)\right) \\ &= \arg\min_{\pi' \in \Pi} J_{\pi_{\text{old}}}^{\mathrm{SKL}}\left(\pi'\left(\cdot \mid \mathbf{s}_t\right)\right).\end{aligned}$$

It must be the case that $J_{\pi_{\text{old}}}^{\mathrm{SKL}}(\pi_{\text{new}}(\cdot \mid \mathbf{s}_t)) \leq J_{\pi_{\text{old}}}^{\mathrm{SKL}}(\pi_{\text{old}}(\cdot \mid \mathbf{s}_t))$ since we can always choose $\pi_{\text{new}} = \pi_{\text{old}} \in \Pi$. Hence, according to Proposition 1 and Proposition 2, we have $\mathrm{D}_{\mathrm{KL}}\left(\pi_{\text{new}}(\cdot \mid \mathbf{s}_t) \parallel \pi_Q(\cdot \mid \mathbf{s}_t)\right) \leq \mathrm{D}_{\mathrm{KL}}\left(\pi_{\text{old}}(\cdot \mid \mathbf{s}_t) \parallel \pi_Q(\cdot \mid \mathbf{s}_t)\right)$, and then, rewrite it in expectation form

$$\begin{aligned}&\mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{new}}}\left[\log \pi_{\text{new}}\left(\mathbf{a}_t|\mathbf{s}_t\right) - \alpha^{-1} Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \log Z^{\pi_{\text{old}}}\left(\mathbf{s}_t\right)\right] \\ &\leq \mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{old}}}\left[\log \pi_{\text{old}}\left(\mathbf{a}_t|\mathbf{s}_t\right) - \alpha^{-1} Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \log Z^{\pi_{\text{old}}}\left(\mathbf{s}_t\right)\right]\end{aligned}$$

and since the partition function $Z^{\pi_{\text{old}}}$ depends only on the state, the inequality reduces to

$$\begin{aligned}&\mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{new}}}\left[\alpha \log \pi_{\text{new}}\left(\mathbf{a}_t|\mathbf{s}_t\right) - Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right] \\ &\leq \mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{old}}}\left[\alpha \log \pi_{\text{old}}\left(\mathbf{a}_t|\mathbf{s}_t\right) - Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right].\end{aligned}$$

Then, we have

$$\begin{aligned}&\mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) - \alpha \log \pi_{\text{new}}\left(\mathbf{a}_t|\mathbf{s}_t\right)\right] \\ &\geq \mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{old}}}\left[Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) - \alpha \log \pi_{\text{old}}\left(\mathbf{a}_t|\mathbf{s}_t\right)\right] \\ &= V^{\pi_{\text{old}}}\left(\mathbf{s}_t\right).\end{aligned} \tag{28}$$

Next, consider the soft Bellman equation

$$\begin{aligned}Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) &= r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V^{\pi_{\text{old}}}\left(\mathbf{s}_{t+1}\right)\right] \\ &\leq r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[\mathbb{E}_{\mathbf{a}_{t+1} \sim \pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right)\right.\right. \\ &\qquad \left.\left. - \alpha \log \pi_{\text{new}}\left(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}\right)\right]\right] \\ &\quad \vdots \\ &\leq Q^{\pi_{\text{new}}}\left(\mathbf{s}_t, \mathbf{a}_t\right)\end{aligned} \tag{29}$$

where we have repeatedly expanded $Q^{\pi_{\text{old}}}$ on the RHS by applying the soft Bellman equation and the bound in (28).

TABLE III
MESO HYPERPARAMETERS

| Parameter | Value |
|---|---|
| *Shared* | |
| optimizer | Adam |
| learning rate | $3 \cdot 10^{-4}$ |
| discount ($\gamma$) | 0.99 |
| replay buffer size | $10^6$ |
| number of hidden layers (all networks) | 2 |
| number of hidden units per layer | 256 |
| number of samples per minibatch | 256 |
| nonlinearity | ReLU |
| target smoothing coefficient ($\tau$) | 0.005 |
| target update interval | 1 |
| gradient steps | 1 |
| *SAC* | |
| Target entropy automatic tuning ($\alpha$) | method from [15] |
| *MeSO* | |
| weight of forward KL divergence($\beta$) | 0.7 |
| decay rate of entropy in target Q ($\alpha'$) | $\alpha \cdot (1 - rate) \cdot w$ |

[2] https://statproofbook.github.io/P/kl-conv.html

[3] https://stats.stackexchange.com/questions/7440/kl-divergence-between-two-univariate-gaussians

TABLE IV
INITIAL WEIGHT OF DIFFERENT ENVIRONMENTS

| Environment | Initial weight ($w$) |
|---|---|
| Humanoid | 7 |
| Ant | 7 |
| HalfCheetah | 1 |
| Walker | 1 |
| Hopper | 1 |
| Swimmer | 1 |
| HumanoidFlagrun | 0 |
| HumanoidFlagrunHard | 0 |

### C. Hyperparameters

Table III provides an overview of the common MeSO parameters utilized in the comparative evaluation presented in Fig. 4. We anneal the parameter $\alpha'$ by set $\alpha' = (1 - \text{current/total}) \cdot \alpha \cdot \text{weight}$. In a specific task, to get the best result, we set different initial weights of $\alpha'$, as listed in Table IV.

## REFERENCES


[1] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2171–2186.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[3] E. Hazan, S. M. Kakade, K. Singh, and A. V. Soest, "Provably efficient maximum entropy exploration," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2681–2691.

[4] G. Jäger, "Maximum entropy models and stochastic optimality theory," in *Architectures, Rules, and Preferences: Variations on Themes By Joan W. Bresnan*, A. Zaenen, J. Simpson, T. H. King, J. Grimshaw, J. Maling, and C. Manning, Ed., Stanford, CA, USA: CSLI Publications, Stanford Univ., 2007, pp. 467–479.

[5] P. J. Coles, M. Berta, M. Tomamichel, and S. Wehner, "Entropic uncertainty relations and their applications," *Rev. Mod. Phys.*, vol. 89, no. 1, Feb. 2017, Art. no. 015002.

[6] B. Eysenbach and S. Levine, "Maximum entropy RL (provably) solves some robust RL problems," 2021, *arXiv:2103.06257*.

[7] Z. Ahmed, N. L. Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 151–160.

[8] R. Zhao, X. Sun, and V. Tresp, "Maximum entropy-regularized multi-goal reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7553–7562.

[9] L. Shani, Y. Efroni, and S. Mannor, "Adaptive trust region policy optimization: Global convergence and faster rates for regularized MDPs," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 5668–5675.

[10] G. Qiao, G. Liu, P. Poupart, and Z. Xu, "Multi-modal inverse constrained reinforcement learning from a mixture of demonstrations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 60384–60396.

[11] S. Mohammadpour, E. Bengio, E. Frejinger, and P. Bacon, "Maximum entropy GFlowNets with soft Q-learning," in *Proc. 27th Int. Conf. Artif. Intell. Statist.*, 2023, pp. 2593–2601.

[12] J. Hao et al., "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 8762–8782, Jul. 2024.

[13] T. A. Berrueta, A. Pinosky, and T. D. Murphey, "Maximum diffusion reinforcement learning," *Nature Mach. Intell.*, vol. 6, no. 5, pp. 1–11, May 2024.

[14] D. Tiapkin et al., "Fast rates for maximum entropy exploration," in *Proc. 40th Int. Conf. Mach. Learn.*, Jul. 2023, pp. 34161–34221.

[15] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[16] J. Shore and R. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 1, pp. 26–37, Jan. 1980.

[17] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2020, pp. 737–744.

[18] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.

[19] OpenAI. (2017). *Introducing Roboschool*. [Online]. Available: https://openai.com/research/roboschool

[20] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

[21] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[22] M. Volodymyr et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[23] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2016, vol. 30, no. 1, pp. 2094–2100.

[24] H. V. Hasselt, "Double Q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 2613–2621.

[25] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 4, Feb. 2018, pp. 2587–2601.

[26] Z. Wang, N. D. Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 1995–2003.

[27] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *Proc. 32nd Conf. Uncertainty Artif. Intell.*, Jun. 2016, pp. 202–211.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, vol. 5, Jul. 2018, pp. 2976–2989.

[29] C. Banerjee, Z. Chen, and N. Noman, "Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experiences," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 3121–3129, Mar. 2024.

[30] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *Proc. 19th Int. Conf. Mach. Learn.*, 2002, pp. 267–274.

[31] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015, *arXiv:1502.05477*.

[32] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 22–31.

[33] W. Meng, Q. Zheng, Y. Shi, and G. Pan, "An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 5, pp. 2223–2235, May 2022.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[35] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 1607–1612.

[36] C. Daniel, G. Neumann, O. Kroemer, and J. Peters, "Hierarchical relative entropy policy search," *J. Mach. Learn. Res.*, vol. 17, no. 93, pp. 1–50, 2016.

[37] A. Abdolmaleki, J. Tobias Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a Posteriori policy optimisation," 2018, *arXiv:1806.06920*.

[38] H. Francis Song et al., "V-MPO: On-policy maximum a Posteriori policy optimization for discrete and continuous control," 2019, *arXiv:1909.12238*.

[39] D. Wang, H. Liu, and Q. Liu, "Variational inference with tail-adaptive f-divergence," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 5737–5747.

[40] D. Ye et al., "Mastering complex control in MOBA games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 6672–6679.

[41] X. Chen et al., "The sufficiency of off-policyness and soft clipping: PPO is still insufficient according to an off-policy measure," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 7078–7086.

[42] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. thesis, Dept. Mach. Learn., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2010.

[43] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 322–330.

[44] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[45] J. Duan et al., "Distributional soft actor-critic with three refinements," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 5, pp. 3935–3946, May 2025.
[46] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6584–6598, Nov. 2022.
[47] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.
[48] E. Nikishin et al., "Improving stability in deep reinforcement learning with weight averaging," in *Proc. Uncertainty Artif. Intell. Workshop Uncertainty Deep Learn.*, 2018, pp. 307–312.

**Xing Chen** received the Ph.D. degree from the School of Artificial Intelligence, Jilin University, Changchun, China, in 2025.

His current research interests focus on deep reinforcement learning algorithms and applications.

**Yewen Li** received the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2025.

His research interests include reinforcement learning, generative models, and computational advertising.

**Xiaofeng Cao** received the Ph.D. degree from Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW, Australia, in 2021.

He is an Associate Professor with the School of Artificial Intelligence, Jilin University, Changchun, China. He has more than 60 academic works published in IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, ICML, NeurIPS, ICLR, and ECML. His research interests include the PAC learning theory, agnostic learning algorithms, generalization analysis, and hyperbolic (non-Euclidean) geometry.

Dr. Cao served as a PC Member and a reviewer.

**Hechang Chen** received the Ph.D. degree from the College of Computer Science and Technology, Jilin University (JLU), Changchun, China, in December 2018.

He is an Associate Professor with the School of Artificial Intelligence, JLU. He has published more than 60 articles in many top international conferences and journals, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *ACM Transactions on Knowledge Discovery from Data*, NeurIPS, AAAI, IJCAI, SIGIR, WWW, and ICDE. His current research interests include machine learning, data mining, reinforcement learning, and knowledge engineering.

**Hengshuai Yao** received the Ph.D. degree in reinforcement learning from the Reinforcement Learning and Artificial Intelligence (RLAI) Laboratory, University of Alberta, Edmonton, AB, Canada, in 2014.

During his Ph.D. studies, he worked on reinforcement learning theory, algorithms, and web applications. He is currently an Adjunct Professor with the University of Alberta, Edmonton, and works on reinforcement learning and large language models for sapient intelligence. His thesis focused on model-based reinforcement learning with linear function approximation.

**Bo An** (Senior Member, IEEE) received the Ph.D. degree in computer science from The University of Massachusetts, Amherst, MA, USA, in 2010.

He is currently a President's Chair Professor of computer science, the Head of the Division of Artificial Intelligence, and the Director of the Centre of AI-for-X, Nanyang Technological University, Singapore. He has published more than 150 refereed papers at AAMAS, IJCAI, AAAI, ICLR, NeurIPS, ICML, AISTATS, ICAPS, KDD, UAI, EC, WWW, JAAMAS, and AIJ. His research results have been successfully applied to many domains, including infrastructure security, sustainability, and e-commerce. His current research interests include artificial intelligence, multiagent systems, computational game theory, reinforcement learning, and optimization.

**Yi Chang** (Senior Member, IEEE) is currently the Dean of the School of Artificial Intelligence, Jilin University, Changchun, China. He became a Chinese National Distinguished Professor in 2017 and the ACM Distinguished Scientist in 2018. Before joining academia, he was the Technical Vice President with Huawei Research America, in charge of knowledge graph, question answering, and vertical search projects. Before that, he was the Research Director of Yahoo Labs/Research, Sunnyvale, CA, USA, from 2006 to 2016, in charge of search relevance of Yahoo's web search engine and vertical search engines. He has authored two books and more than 100 articles in top conferences or journals. His research interests include information retrieval, data mining, machine learning, natural language processing, and artificial intelligence.