MIND YOUR ENTROPY: FROM MAXIMUM ENTROPY TO TRAJECTORY ENTROPY-CONSTRAINED RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Maximum entropy has become a mainstream off-policy reinforcement learning (RL) framework for balancing exploitation and exploration. However, two bottlenecks still limit further performance improvement: *(1) non-stationary Q-value estimation* caused by jointly injecting entropy and updating its weighting parameter, i.e., temperature; and *(2) short-sighted local entropy tuning* that adjusts temperature only according to the current single-step entropy, without considering the effect of cumulative entropy over time. In this paper, we extends maximum entropy framework by proposing a trajectory entropy-constrained reinforcement learning (TECRL) framework to address these two challenges. Within this framework, we first separately learn two Q-functions, one associated with reward and the other with entropy, ensuring clean and stable value targets unaffected by temperature updates. Then, the dedicated entropy Q-function, explicitly quantifying the expected cumulative entropy, enables us to enforce a trajectory entropy constraint and consequently control the policy's long-term stochasticity. Building on this TECRL framework, we develop a practical off-policy algorithm, DSAC-E, by extending the state-of-the-art distributional soft actor-critic with three refinements (DSAC-T). Empirical results on the OpenAI Gym benchmark demonstrate that our DSAC-E can achieve higher returns and better stability.

1 INTRODUCTION

Balancing exploration and exploitation remains a central challenge in reinforcement learning (RL) (Sutton & Barto, 2018; Li, 2023). To address this, off-policy methods have leveraged the maximum entropy principle, which encourages agents to act as randomly as possible while still optimizing for high returns (Wang et al., 2022; Haarnoja et al., 2017). By augmenting the objective with a temperature-weighted entropy term, algorithms like Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) and its distributional variant DSAC (Duan et al., 2021; 2025) have achieved state-of-the-art performance on continuous control benchmarks like MuJoCo, proving to be highly effective and robust (Eysenbach & Levine, 2022).

However, a fixed temperature parameter can lead to a policy that is either excessively stochastic or unnecessarily deterministic (Rawlik et al., 2012). This is because a single temperature value cannot optimally balance exploration and exploitation across all phases of training; a high temperature may hinder convergence, while a low temperature can lead to premature exploitation of a suboptimal solution (Fox et al., 2016). To mitigate this issue, modern maximum entropy RL incorporates an automated temperature adjustment mechanism (Haarnoja et al., 2018b). Using the policy's current per-step entropy as a feedback signal, this mechanism dynamically tunes the temperature throughout training, aligning it with a predefined target. Therefore, it ensures that a desired level of stochasticity is maintained across all situations (Hazan et al., 2019).

Despite the remarkable empirical success, maximum entropy methods still face two critical bottlenecks that hinder further progress. (1) The first issue is *non-stationary Q-value estimation*, which stems from the tight coupling of reward and entropy (Schulman et al., 2017a). Since the temperature parameter is updated simultaneously, the injected temperature-weighted entropy term is directly altering the Q-value targets, causing them to become non-stationary. This process can destabilize value learning and ultimately undermine policy optimization (Lillicrap et al., 2016). (2) Second, and perhaps more fundamentally, while some works have explored constraining entropy, they all

suffer from *short-sighted local entropy tuning* (Haarnoja et al., 2018b; Duan et al., 2021; 2025). By regulating only the local current-step entropy, these methods neglect the long-term influence of stochasticity over entire trajectories. More critically, they enforce a uniform entropy target across all states, as if every situation demands the same degree of randomness. This one-size-fits-all assumption is overly restrictive; it fails to acknowledge that effective exploration should adapt to the underlying dynamics and the agent's learning progress (Tokic, 2010; Sun et al., 2022). This fundamental disconnect ignores the varying exploration needs of different situations.

The observed two bottlenecks naturally raise a question: *can we move beyond maximum entropy by directly and cleanly controlling what really matters—the cumulative entropy of the policy?* We argue the answer is yes by introducing a trajectory entropy-constrained (TEC) RL framework. To ensure a stable and interpretable learning process, our core innovation is to completely decouple the reward and entropy signals by learning two separate Q-functions. This separation ensures clean Q-value targets, and the dedicated entropy critic enables us to enforce a trajectory-level constraint on the policy's cumulative entropy. This design inherently breaks from traditional single-step restriction, enabling a more principled and long-term control of policy stochasticity.

To demonstrate the practical advantages of our framework, we introduce DSAC-E, an extension of the state-of-the-art Distributional Soft Actor-Critic with Three refinements (DSAC-T) algorithm (Duan et al., 2025). DSAC-E integrates the strengths of DSAC-T's distributional value estimation with our proposed trajectory entropy constraint. By decoupling the reward and entropy Q-values and adjusting the trajectory-level entropy budget, our DSAC-E achieves cleaner and more effective exploitation alongside more controllable exploration. Empirical results on the OpenAI Gym continuous control benchmark (Brockman et al., 2016) demonstrate that DSAC-E not only achieves superior final returns but also exhibits better training stability than strong maximum entropy baselines.

Our contributions are summarized in threefold:

- We identify and analyze the impact of two bottlenecks in conventional maximum entropy RL: *(1) non-stationary Q-value estimation* and *(2) short-sighted local entropy tuning*. These issues motivate us to execute reward-entropy separation and trajectory-level entropy constraint;
- To address these two identified bottlenecks, we propose the TECRL framework. Within this framework, we first eliminate the *(1) non-stationary Q-value estimation* problem by decoupling reward and entropy signals into two separate critics, while temperature is excluded from the learning processes of both critics. Then the dedicated entropy critic allows us to enforce a trajectory-level entropy constraint, thereby overcoming the issue of *(2) short-sighted local entropy tuning*. Furthermore, we provide a rigorous theoretical analysis demonstrating that appropriately selecting a trajectory entropy budget can yield a higher performance bound;
- We introduce DSAC-E, a practical instantiation of our TECRL framework built on DSAC-T, the state-of-the-art maximum entropy algorithm. Through this instantiation, we demonstrate that our framework enables superior performance on complex continuous control tasks.

2 PRELIMINARIES

Maximum entropy RL. While standard RL seeks a policy that maximizes the expected accumulated return, maximum entropy RL (Haarnoja et al., 2017) extends this by adopting an objective function that incorporates a policy entropy term as

$$J_\pi = \mathop{\mathbb{E}}_{s_t\sim\rho_\pi}\Big[\sum_{t=0}^{\infty}\gamma^t[r_t + \alpha\mathcal{H}(\pi(\cdot|s_t))]\Big], \tag{1}$$

where $\gamma \in (0,1)$ is the discount factor, ρ_t is the state visitation distribution, α is the temperature coefficient, and the single-step policy entropy $\mathcal{H}$ is expressed as

$$\mathcal{H}(\pi(\cdot|s_t)) = \mathop{\mathbb{E}}_{a_t\sim\pi(\cdot|s_t)}\big[-\log\pi(a_t|s_t)\big]. \tag{2}$$

The optimal policy can be derived through a maximum entropy variant of policy iteration, commonly known as soft policy iteration (Wang et al., 2022). This iterative process alternates between two key stages: (1) soft policy evaluation (PEV) and (2) soft policy improvement (PIM).

In soft PEV, provided a policy π, for a given policy π, we can apply the soft Bellman operator $\mathcal{B}^\pi$ to learn the soft Q-value, as shown by the soft Bellman expectation equation:

$$\mathcal{B}^{\text{soft}}[Q^{\text{soft}}(s,a)] = r + \gamma \mathbb{E}_{s'\sim p, a'\sim\pi}[Q^{\text{soft}}(s',a') - \alpha \log \pi(a'|s')], \tag{3}$$

where the definition of soft Q-value is

$$Q^{\text{soft}}(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t + \sum_{t=1}^{\infty} \gamma^t \alpha \mathcal{H}(\pi(\cdot|s_t)) \,\middle|\, s_0 = s, a_0 = a \right]. \tag{4}$$

One might ask why we write the reward and entropy signals as two separate summation terms. The reason is to highlight the difference in their starting indices. The reward signal is accumulated from the current time step, with a summation index of $t = 0$, while the policy entropy is accumulated from the next time step, with a summation index of $t = 1$. This difference is evident from the soft Bellman expectation equation in Eq. (3): the first term on the right-hand side, r, does not have a corresponding policy entropy term at the same time step. In fact, the missing current entropy $\mathcal{H}(\pi(\cdot|s_0))$ occurs in the subsequent soft PIM step.

In soft policy improvement (PIM), the goal is to find a new policy that outperforms the current policy. This is achieved by directly maximizing an entropy-augmented objective, a process equivalent to:

$$\pi_{\text{new}} = \arg\max_\pi \mathop{\mathbb{E}}_{s\sim\rho_\pi, a\sim\pi} \big[Q^{\text{soft}}(s,a) - \alpha \log \pi(a|s) \big]. \tag{5}$$

The convergence of soft policy iteration to the optimal maximum entropy policy is a well-established result in the field, as shown by (Haarnoja et al., 2017).

Temperature tuning. A key advancement in the latest maximum entropy frameworks is the automatic management of the temperature parameter α. Instead of being a fixed hyperparameter, α is treated as a learnable variable. The objective is to minimize

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi}\big[-\alpha\big(\log \pi(a_t|s_t) + \mathcal{H}_0 \big)\big], \tag{6}$$

where the default value of $\mathcal{H}_0$ is commonly set as $-\dim(\mathcal{A})$, i.e., the minus of the number of action dimensions. This mechanism achieves a dynamic balance between exploration and exploitation by maintaining the policy's local entropy close to a predefined target entropy $\mathcal{H}_0$ across all situations (Haarnoja et al., 2018a).

3 Method

3.1 Two Bottlenecks of Maximum Entropy RL

Previously, we briefly introduced two bottlenecks that exist in the current maximum entropy RL framework. Now, combining with specific formulas, we will more formally and mathematically explain their origins and their impact on policy learning.

(1) Non-stationary Q-value estimation. In each soft PEV step, as shown in Eq. (3), the target value is calculated by

$$y_{\text{target}} = r(s,a) + \gamma[Q^{\text{soft}}(s',a') + \alpha \mathcal{H}(\pi(\cdot|s'))]. \tag{7}$$

When the temperature α is updated at the same time, the target value distribution shifts dynamically. This entanglement injects additional variance and bias into Q-value estimation, degrading subsequent policy improvement steps that rely on stable value predictions.

(2) Short-sighted local entropy tuning. In each soft PIM step, as shown in Eq. (6), the existing temperature tuning mechanism aligns every local single-step entropy to a fixed target by adjusting α to match $\mathbb{E}[-\log \pi(a|s)]$ to some desired value. However, it would be better to adjust the trajectory entropy to control the long-term policy stochasticity, which is defined as:

$$\mathcal{H}_{\text{traj}}(s) = \mathbb{E}_{\tau\sim\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \,\Big|\, s_0 = s \Big]. \tag{8}$$

In summary, while the maximum-entropy framework is a powerful tool for policy learning, its effectiveness is still hindered by the two identified bottlenecks. These limitations motivate us to execute reward-entropy separation to ensure clean and stable value learning and rethink maximum entropy RL from a trajectory-level entropy constraint perspective.

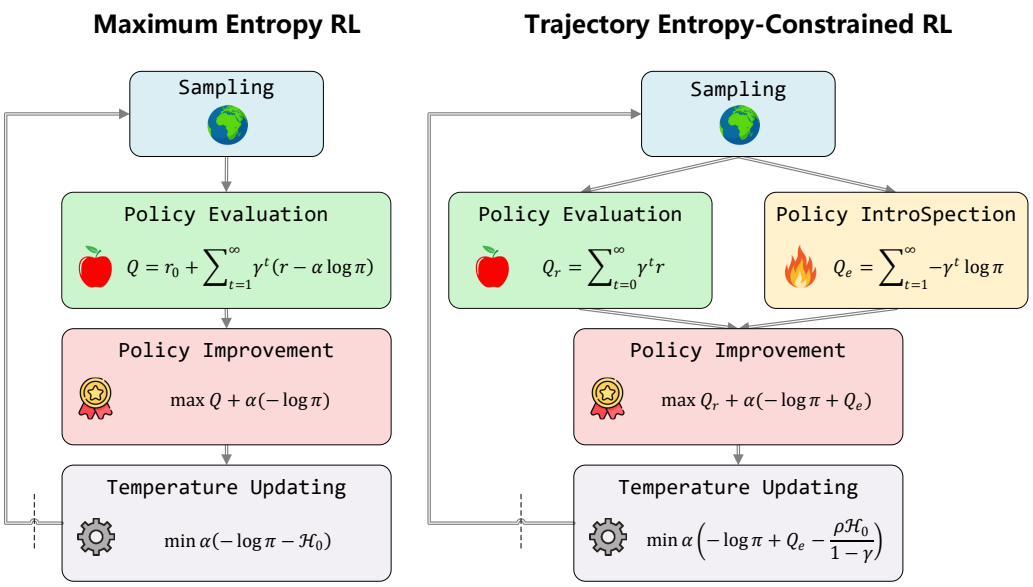


Figure 1: **Comparison** between standard maximum entropy RL (**left**) and our trajectory entropy-constrained (TEC) RL (**right**). Our TECRL framework comprises four key components: a reward-centric policy evaluation (PEV), an entropy-centric policy introspection (PIS), a policy improvement (PIM) that retains the exact soft policy objective, and a temperature update (TUP) tuning the temperature guided by the trajectory entropy constraint.

3.2 Trajectory Entropy-Constrained Reinforcement Learning

To address the two bottlenecks identified earlier, we propose Trajectory Entropy-Constrained Reinforcement Learning (TECRL). It formulates an explicit equality constraint on the trajectory-level entropy to control the policy stochasticity, which yields the following policy optimization problem:

$$\begin{aligned} &\max_{\pi} \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\Big] \\ &\text{s.t.} \quad \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t))\Big] = \mathcal{H}_{\text{budget}}. \end{aligned} \tag{9}$$

Under this trajectory entropy constraint, the agent is required to strategically distribute a fixed budget of randomness across its entire trajectory. This offers a more principled way to mitigate the dilemma of under- and over-exploration.

To practically solve the optimal policy, our TECRL integrates four alternating steps: (1) Policy Evaluation (PEV) estimates the expected cumulative reward; (2) Policy Introspection (PIS) estimates the expected cumulative entropy; (3) Policy Improvement (PIM) jointly leverages both critics to formulate soft policy objective; and (4) Temperature Updating (TUP) adapts the temperature to enforce the trajectory entropy constraint. Below we detail these four steps one by one.

(1) Policy Evaluation (PEV). This step learns a reward-centric critic Q_r defined as

$$Q_r(s, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a\right], \tag{10}$$

The PEV loss follows the standard Bellman expectation equation:

$$\mathcal{L}_{\text{PEV}} = (Q_r(s, a) - y_r)^2, \quad \text{where} \quad y_r = r(s, a) + \gamma\, \mathbb{E}_{s', a'}[Q_r(s', a')], \tag{11}$$

This reward-centric critic explicitly excludes entropy bonuses, which ensures a clean value target uninfluenced by policy stochasticity.

Algorithm 1 Trajectory Entropy-Constrained Reinforcement Learning (TECRL)

Initialize policy π_θ, reward critic $Q_{r,\psi}$, entropy critic $Q_{e,\phi}$, temperature α, replay buffer $\mathcal{D}$
for each iteration **do**
  Observe s_t, sample $a_t \sim \pi_\theta(a|s_t)$, execute a_t, receive r_t, next state s_{t+1}
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
  Sample mini-batch $\{(s, a, r, s')\} \sim \mathcal{D}$
  Update Q_r with Eq. (11) ▷ (PEV) Policy Evaluation
  Update Q_e with Eq. (13) ▷ (PIS) Policy Introspection
  Update π_θ with Eq. (14) ▷ (PIM) Policy Improvement
  Update α with Eq. (15) ▷ (TUP) Temperature Updating
end for

(2) Policy Introspection (PIS). This step learns an entropy-centric critic Q_e. For a Gaussian policy, the entropy of the current step is straightforward to compute. Therefore, we define Q_e as the cumulative policy entropy from the next time step to infinity, which is defined as

$$Q_e(s,a) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \,\middle|\, s_0 = s, a_0 = a \right]. \tag{12}$$

Notably, it also does not contain the temperature α, so its target value is clean and explicit. The PIS loss follows an entropy Bellman expectation equation:

$$\mathcal{L}_{\text{PIS}} = (Q_e(s,a) - y_e)^2, \quad \text{where} \quad y_e = \gamma \mathcal{H}(\pi(\cdot|s')) + \gamma\, Q_e(s', a'). \tag{13}$$

The mathematical correspondence between Eq. (12) and Eq. (13) can be seen in the Appendix A.2, and the convergence proof of the newly proposed PIS is presented in Appendix A.3.

We refer to this process as policy introspection because the Q_e value reflects the future cumulative entropy of the current policy across different state-action pairs. In essence, it quantifies the long-term stochasticity inherent to the policy itself.

(3) Policy Improvement (PIM). With dual critics Q_r and Q_e, We can formulate a policy loss as:

$$\mathcal{L}_{\text{PIM}} = \underbrace{Q_r(s,a)}_{\text{cumulative reward}} + \alpha \underbrace{(-\log \pi(a|s) + Q_e(s,a))}_{\text{cumulative entropy}}. \tag{14}$$

This PIM loss aligns with the soft policy objective shown in Eq. (1). Q_r represents the cumulative reward, $-\log \pi(a|s)$ is the current policy entropy, and Q_e represents the cumulative entropy starting from the next time step. Therefore, our PIM is compliant with the maximization term in Eq. (9).

(4) Temperature Updating (TUP). Finally, the aim of TUP is tuning α to enforce the trajectory entropy constraint, whose loss is

$$\mathcal{L}_{\text{TUP}} = -\alpha \Big(\underbrace{-\log \pi(a|s) + Q_e(s,a)}_{\text{cumulative entropy}} - \mathcal{H}_{\text{budget}} \Big). \tag{15}$$

This mechanism extends existing temperature tuning in Eq. (6) by replacing uniform local entropy matching with a trajectory-level entropy constraint in Eq. (9). We set $\mathcal{H}_{\text{budget}}$ as $\rho H_0/(1-\gamma)$, The division by $(1-\gamma)$ is to keep the magnitude consistent with the local entropy tuning of the maximum entropy. ρ is an entropy scaling factor that can adjust the budget value.

Summary. Our proposed TECRL framework is grounded in two primary claims: (1) TECRL enables *more stable and effective exploitation*. This is because the reward-centric value function is now decoupled from the entropy objective, allowing it to provide a more accurate and dedicated prediction to guide policy improvement. (2) TECRL enables *more strategic and controllable exploration.* By having the agent dynamically allocate its finite entropy budget where it is most needed, the method facilitates the preservation of high-value behaviors while preventing unstable swings in policy stochasticity. The full pseudocode is summarized in Algorithm 1.

3.3 Theoretical Analysis on Performance Bound

We formalize how a trajectory entropy constraint affects policy performance and demonstrate why a properly chosen entropy budget can raise the performance upper bound. We first denote π^*_{soft} as the optimal policy under the standard maximum entropy RL setting, which maximizes the soft objective

$$J_{\text{MaxEnt}}(\pi^*_{\text{soft}}) = R^*_{\text{MaxEnt}} + \alpha^*_{\text{soft}}\,\mathcal{H}^*_{\text{soft}}, \tag{16}$$

where R^*_{MaxEnt} and $\mathcal{H}^*_{\text{soft}}$ represent the optimal return and cumulative entropy, respectively, and $\alpha^*_{\text{soft}} > 0$ is the optimal temperature parameter.

Let R_{TEC} be the return of our TECRL policy. We assume that the entropy budget $\mathcal{H}_{\text{budget}}$ is chosen to be within the feasible range of entropy values encountered during the MaxEnt optimization process. Specifically, it is neither smaller than the minimal achievable entropy nor larger than the maximal entropy $\mathcal{H}^*_{\text{soft}}$ obtained by the optimal maximum-entropy policy π^*_{soft}. Therefore, with the same temperature α^*_{soft}, we have the following inequality

$$J_{\text{MaxEnt}}(\pi^*_{\text{soft}}) \geq R_{\text{TEC}} + \alpha^*_{\text{soft}}\,\mathcal{H}^*_{\text{budget}}. \tag{17}$$

By rearranging this inequality, the return of our TECRL can be bounded from above as

$$\begin{aligned} R_{\text{TEC}} &\leq J_{\text{MaxEnt}}(\pi^*_{\text{soft}}) - \alpha^*_{\text{soft}}\,\mathcal{H}_{\text{budget}} \\ &= R^*_{\text{MaxEnt}} + \alpha^*_{\text{soft}}\,(\mathcal{H}^*_{\text{soft}} - \mathcal{H}_{\text{budget}}). \end{aligned} \tag{18}$$

This inequality explicitly shows that our achievable return is bounded by a quantity proportional to the entropy gap $\mathcal{H}^*_{\text{soft}} - \mathcal{H}_{\text{budget}}$. This analysis demonstrates that appropriately selecting a trajectory entropy budget can lead to a higher performance bound.

4 Experiments

4.1 Main Experiment

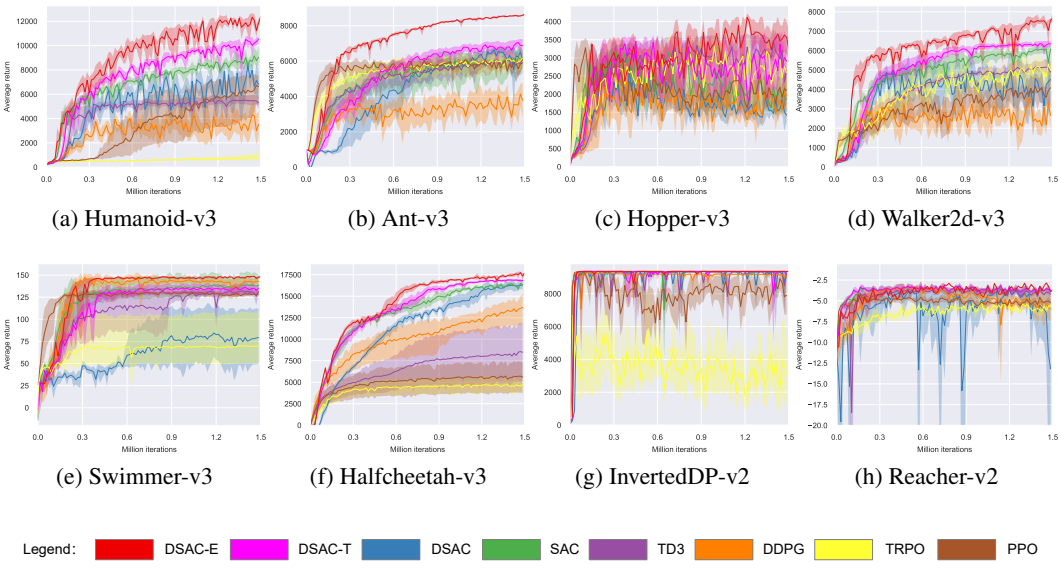
Benchmark. We evaluate performance on a suite of standard continuous control tasks from the OpenAI Gym interface (Brockman et al., 2016). Specifically, we choose 8 Mujoco tasks: Humanoid-v3, Ant-v3, Hopper-v3, Walker2d-v3, Swimmer-v3, HalfCheetah-v3, InvertedDoublePendulum-v2 (abbreviated as InvertedDP-v2) and Reacher-v2. Details are provided in Appendix B.

Baselines. We consider 7 well-known model-free algorithms, including trust region policy optimization (TRPO) (Schulman et al., 2015), proximal policy optimization (PPO) (Schulman et al., 2017b), deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016), twin delayed deep deterministic policy gradient (TD3) (Fujimoto et al., 2018), soft actor-critic (SAC) (Haarnoja et al., 2018a), Distributional SAC (DSAC) (Duan et al., 2021) and its latest version DSAC-T (Duan et al., 2025). See Appendix D for detailed hyperparameters.

Our method. Our proposed DSAC-E algorithm is built on the DSAC-T, inheriting all of its hyperparameters. For the newly introduced hyperparameter ρ, we set its value to 20 for the Humanoid-v3 and Walker2d-v3 tasks, and to 1 for all other tasks. The reason for setting larger ρ values for these two tasks is that they are relatively high-dimensional and that the robots are particularly prone to falling over due to overly random actions. Recall that the base single-step entropy budget H_0 is a negative value, so a larger ρ means a smaller budget $\rho H_0/(1-\gamma)$.

Evaluation protocol. The total training step for all experiments is set at 1.5 million, with the results of all experiments averaged over 5 random seeds. For each seed, the metric is derived by averaging the highest return values observed during the final 10% of iteration steps in each run, with evaluations conducted every 15,000 iterations. Each assessment result is the average of ten episodes. The results from the 5 seeds are then aggregated to calculate the mean and standard deviation.

Main results. Figure 2 and Table 1 display all the learning curves and numerical performance results , respectively. Our comprehensive findings reveal that across all evaluated 8 tasks, the DSAC-E algorithm consistently matched or surpassed the performance of all competing benchmark algorithms, establishing new state-of-the-art results. Notably, it achieved less oscillation and substantial performance improvements on the Humanoid-v3, Ant-v3, Walker2d-v3, and Hopper-v3 tasks, with improvements of 15.82%, 21.93%, 21.11% and 6.6% over the second-best.



(a) Humanoid-v3 (b) Ant-v3 (c) Hopper-v3 (d) Walker2d-v3

(e) Swimmer-v3 (f) Halfcheetah-v3 (g) InvertedDP-v2 (h) Reacher-v2

Figure 2: **Training curves on benchmarks.** The solid lines correspond to mean and shaded regions correspond to the 95% confidence interval over five runs.

Table 1: **Average final return.** Computed as the mean of the highest return values observed in the final 10% of iteration steps per run. ± corresponds to standard deviation over 5 runs.

Algorithm			Humanoid-v3	Ant-v3	Hopper-v3	Walker2d-v3
Off	w/ entropy	DSAC-E	**12542±280**	**8640±57**	**3901±385**	**7780±137**
		DSAC-T	10829±243	7086±261	3660±533	6424±147
		DSAC	9074±286	6862±53	2135±434	5413±865
		SAC	9336±696	6427±805	2483±943	6201±264
	w/o entropy	TD3	5632±436	6184±487	3569±455	5238±336
		DDPG	5292±663	4549±789	2644±659	4096±68
On	w/ entropy	TRPO	965±555	6203±579	3474±400	5503±593
		PPO	6869±1563	6157±185	2647±482	4832±638
		⇑	15.82%	21.93%	6.58%	21.11%

Algorithm			Swimmer-v3	Halfcheetah-v3	InvertedDP-v2	Reacher-v2
Off	w/ entropy	DSAC-E	**149.3±0.3**	**17904±100**	**9360±0**	**-2.9±0.1**
		DSAC-T	137.6±6.4	17025±157	**9360±0**	-3.1±0.2
		DSAC	83.9±35.6	16542±514	9359±1	-4.3±1.9
		SAC	140.4±14.3	16573±224	**9360±0**	-3.1±0.2
	w/o entropy	TD3	134.0±5.4	8633±4041	9347±15	-3.4±0.2
		DDPG	145.6±4.3	13970±2083	9183±10	-4.5±1.3
On	w/ entropy	TRPO	70.4±38.1	4785±968	6260±2066	-5.0±0.6
		PPO	130.3±2.0	5790±2201	9357±2	-4.4±0.2
		⇑	2.54%	5.16%	0%	6.45%

* **Bolded** and red = best, blue = second-best. ⇑ *means the improvement of the best over the second-best.*

4.2 Ablation Study

We conduct ablation studies on the Humanoid-v3 task to evaluate the contribution of each component.

Reward-entropy separation (RES) and trajectory Entropy Constraint (TEC). We perform a step-wise ablation, considering four algorithms: (1) Our full DSAC-E. (2) DSAC-E w/o TEC, which replaces our trajectory entropy constraint with existing local entropy tuning. (3) DSAC-E w/o TEC and RES, which is close to DSAC-T but with a ρ value of 20. (4) original DSAC-T, which can be understood as having a ρ of 1. As shown in Figure 3 and Table 2, the performance of the algorithms progressively declines as more components are removed. This result confirms the effectiveness of both our RES and TEC modules. Next we will provide a more systematic analysis of the ρ.

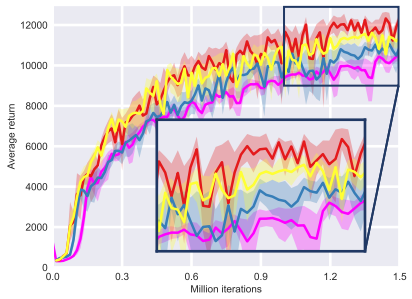


Figure 3: Ablation on the TEC and RES.

Table 2: Results of ablation on TEC and RES.

Algorithm	TAR
DSAC-E (full)	**12542** $\pm$ **280**
DSAC-E w/o TEC	11786 $\pm$ 374
DSAC-E w/o TEC & RES	11455 $\pm$ 404
DSAC-T	10829 $\pm$ 243

Impact of ρ controlling trajectory entropy budget. We further investigate the effect of varying the trajectory entropy budget. Specifically, we apply different ρ values for both DSAC-T and our DSAC-E. As shown in Figure 4 and Table 3, the performance gain of DSAC-T (Figure 4a) is not significant with the adjustment of ρ. Its performance varies only slightly and all results cluster closely together. In contrast, our DSAC-E (Figure 4b) consistently outperforms DSAC-T across all settings, and its performance shows a clearer, more structured dependence on ρ.

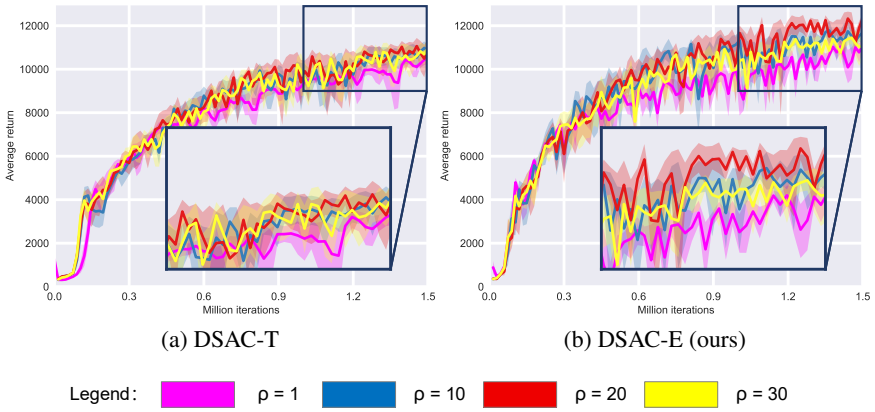


Figure 4: Ablation on the sensitivity to the trajectory entropy budget.

For both DSAC-T and our DSAC-E, performance first improves and then degrades as ρ increases, which aligns with our theoretical analysis: a properly chosen entropy budget can lift the performance bound, whereas an excessively large ρ (corresponding to an overly small entropy budget) reduces exploration and leads to a performance drop. Overall, our DSAC-E achieves higher performance and exhibits a more interpretable sensitivity to ρ, making it easier to tune for high returns.

Table 3: Performance of DSAC-T and our DSAC-E under different ρ values.

Algorithm	$\rho = 1$	$\rho = 10$	$\rho = 20$	$\rho = 30$
DSAC-T	10829 $\pm$ 243	11079 $\pm$ 457	11455 $\pm$ 404	11182 $\pm$ 705
DSAC-E (ours)	**11382** $\pm$ **447**	**12118** $\pm$ **505**	**12542** $\pm$ **280**	**11747** $\pm$ **365**

5 Related Work

Exploration remains a central challenge in RL, and prior studies have proposed various strategies to inject and regulate stochasticity into the policy (Amin et al., 2021). Broadly, existing approaches can be grouped into two main categories: action-noise-based and maximum-entropy-based exploration (Hao et al., 2023). While other alternatives, such as curiosity-driven (Sun et al., 2022) or uncertainty-based (An et al., 2021) exploration, have been explored, they remain less commonly adopted in standard model-free RL algorithms.

Action-noise based exploration. A line of methods in off-policy RL encourages exploration by directly perturbing the agent's actions with a noise process. For instance, DDPG first (Lillicrap et al., 2016) employs Ornstein–Uhlenbeck noise to facilitate temporally correlated exploration, and the TD3 family (Fujimoto et al., 2018; 2023; Seo et al., 2025) turn to simply apply Gaussian noise to each action dimension to effectively maintain randomness during training. Although these approaches are intuitive and easy to implement, they suffer from two key drawbacks. First, the noise is added externally and is entirely separate from the policy's learning objective. The policy itself is unaware of this exploration mechanism, making it a blind, ad hoc process (Plappert et al., 2018; Li et al., 2021). Second, it creates a fundamental inconsistency between training and evaluation. A policy trained with exploratory noise is different from the final policy used for deployment, which can lead to a policy-value mismatch and hinder convergence to a truly optimal solution (Hollenstein et al., 2022; Sikchi et al., 2022). Overall, although action-noise based exploration is straightforward to implement and can yield good performance, its largely heuristic nature diminishes its reliability.

Maximum-entropy based exploration. A more principled framework for exploration is provided by maximum-entropy RL (Haarnoja et al., 2017). By augmenting the standard RL objective with an entropy term, methods such as SAC (Haarnoja et al., 2018a) optimize for both expected return and policy entropy, thereby encouraging diverse behaviors (Nachum et al., 2017). While the latest extensions of SAC further incorporate distributional critics to improve performance (Duan et al., 2021; 2025), they share the same tuning principle of maintaining the policy's single-step entropy at a fixed target. Recent work has explored the use of generative models, such as diffusion models, as policy functions (Yang et al., 2023a; Zhu et al., 2023). While it's difficult to accurately compute the entropy of this class of functions (Yang et al., 2023b), these methods still try to follow the standard maximum-entropy principle and entropy tuning mechanism for exploration, for example, by approximating the policy entropy via GMM fitting or alternatively optimizing the lower bound (Wang et al., 2024; 2025; Ding et al., 2024; Celik et al., 2025). Their entropy tuning mechanism remains inherently uniform across all situations and does not explicitly account for long-term policy stochasticity and the inherent need for adaptive exploration. Our TECRL also employs entropy to monitor policy's stochasticity. However, we shift the focus from local entropy tuning to *trajectory entropy constraint*, highlighting a new perspective on managing policy's long-term stochasticity. We believe this work provides a new avenue for better resolving the exploitation-exploration dilemma, leading to higher performance.

6 Conclusion

In this paper, we revisit the standard maximum entropy RL framework and introduce the trajectory entropy-constrained reinforcement learning (TECRL) framework. Our work addresses two key limitations: (1) non-stationary Q-value estimation and (2) short-sighted local entropy tuning. By separating the reward and entropy Q-functions and applying the trajectory entropy constraint, our framework ensures stable value targets and effective control of long-term policy stochasticity. Building on this, we develop a practical algorithm, DSAC-E, which extends the state-of-the-art DSAC-T baseline. Empirical results on the OpenAI Gym benchmark show that DSAC-E achieves superior returns and greater stability, validating the effectiveness of our TECRL framework.

Moving forward, we plan to validate the applicability of our TECRL framework to real-world robotics and large language models (LLMs). This integration will allow agents to benefit from TECRL's superior long-term stochasticity management, leading to more effective and robust behaviors. We believe this work offers a promising paradigm for addressing the exploration-exploitation trade-off and paving the way for more powerful and robust RL agents.

REFERENCES

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke Van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

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

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE transactions on neural networks and learning systems*, 33(11):6584–6598, 2021.

Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, Shengbo Eben Li, Chang Liu, Ya-Qin Zhang, Bo Cheng, and Keqiang Li. Distributional soft actor-critic with three refinements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. In *International Conference on Learning Representations*, 2022.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 202–211, 2016.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pp. 1587–1596, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. *Advances in neural information processing systems*, 36:61573–61624, 2023.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 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 *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pp. 1861–1870, Stockholmsmässan, Stockholm Sweden, 2018a. PMLR.

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.

Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):8762–8782, 2023.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Jakob Hollenstein, Sayantan Auddy, Matteo Saveriano, Erwan Renaudo, and Justus Piater. Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *arXiv preprint arXiv:2206.03787*, 2022.

Min Li, Tianyi Huang, and William Zhu. Adaptive exploration policy for exploration–exploitation tradeoff in continuous action control optimization. *International Journal of Machine Learning and Cybernetics*, 12(12):3491–3501, 2021.

Shengbo Eben Li. *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer Verlag, Singapore, 2023.

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. In *4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico, 2016.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *30th Advances in Neural Information Processing Systems (NeurIPS 2017)*, pp. 2775–2785, Long Beach, CA, USA, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

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.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, (ICML 2015)*, pp. 1889–1897, Lille, France, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

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

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022.

Hao Sun, Lei Han, Rui Yang, Xiaoteng Ma, Jian Guo, and Bolei Zhou. Exploit reward shifting in value-based deep-rl: Optimistic curiosity-based exploration and conservative exploitation via linear reward shaping. *Advances in neural information processing systems*, 35:37719–37734, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Michel Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual conference on artificial intelligence*, pp. 203–210. Springer, 2010.

Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2022.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

Yinuo Wang, Mining Tan, Wenjun Zou, Haotian Lin, Xujie Song, Wenxuan Wang, Tong Liu, Likun Wang, Guojian Zhan, Tianze Zhu, et al. Enhanced dacer algorithm with high diffusion efficiency. *arXiv preprint arXiv:2505.23426*, 2025.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023a.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023b.

Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Haoquan Guo, Tingting Chen, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023.

A Theoretical Analysis

A.1 Useful Lemmas

Lemma 1 (Convergence of γ-Contraction Mappings)**.** *Let (X, d) be a complete metric space, and let $\mathcal{B} : X \to X$ be a γ-contraction mapping with $0 < \gamma < 1$. This means that for all $x, y \in X$,*

$$d(\mathcal{B}(x), \mathcal{B}(y)) \leq \gamma \cdot d(x, y), \tag{19}$$

where d is the metric on X. According to Banach's fixed-point theorem, $\mathcal{B}$ has a unique fixed point $x^* \in X$, such that $\mathcal{B}(x^*) = x^*$. Furthermore, for any initial point $x_0 \in X$, the iterative sequence $\{x_n\}$ defined by $x_{n+1} = \mathcal{B}(x_n)$ converges to x^*. The convergence rate is geometric, and we have the inequality

$$d(x_n, x^*) \leq \gamma^n \cdot d(x_0, x^*), \quad \forall n \geq 0. \tag{20}$$

This result not only guarantees the existence and uniqueness of the fixed point but also provides a precise rate at which the sequence approaches x^*, demonstrating the efficiency of contraction mappings in finding fixed points.

A.2 Entropy Bellman Expectation equation in Policy Introspection (PIS)

Here, we build the correspondence between the definition of Q_e in Eq. (12) and the entropy Bellman expectation equation in Eq. (13).

Q_e represents the cumulative policy entropy starting from the next time step, expressed as:

$$Q_e(s, a) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \,\middle|\, s_0 = s, a_0 = a \right]. \tag{21}$$

Our proposed entropy Bellman expectation equation in Eq. (13) states

$$Q_e(s, a) = \gamma \mathcal{H}(\pi(\cdot|s')) + \gamma\, Q_e(s', a'). \tag{22}$$

Substitute the definition of Q_e into the RHS of Eq. (13), we have:

$$\begin{aligned} RHS &= \gamma \mathcal{H}(\pi(\cdot|s')) + \gamma\, Q_e(s', a') \\ &= \gamma \mathcal{H}(\pi(\cdot|s_1)) + \gamma \sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_{t+1})) \\ &= \gamma \mathcal{H}(\pi(\cdot|s_1)) + \sum_{t=1}^{\infty} \gamma^{t+1} \mathcal{H}(\pi(\cdot|s_{t+1})) \\ &= \gamma \mathcal{H}(\pi(\cdot|s_1)) + \sum_{t=2}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \\ &= \sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) = LHS. \end{aligned} \tag{23}$$

Thus, we have proven that the definition of Q_e is the solution of the entropy Bellman expectation equation.

A.3 Convergence of Policy Introspection (PIS)

We prove the convergence of PIS by showing that the entropy Bellman operator $\mathcal{B}_e$, defined as

$$\mathcal{B}_e Q_e(s, a) = \gamma[Q_e(s', a') - \alpha \log \pi(a'|s')], \tag{24}$$

is a γ-contraction mapping.

We analyze the infinity norm of $\mathcal{B}_e$. For any two functions $Q_{e,1}(s,a)$ and $Q_{e,2}(s,a)$, we have:

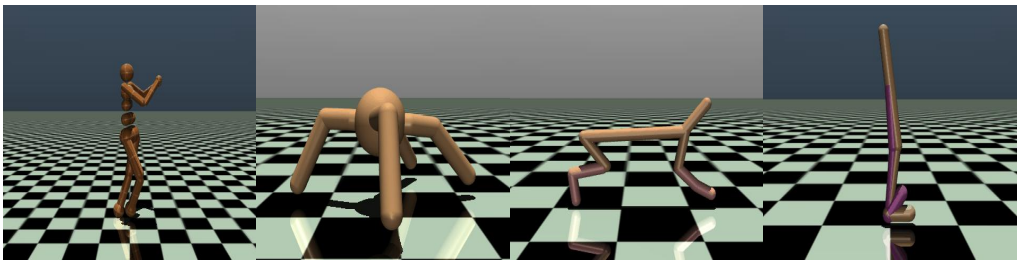
$$\begin{aligned}\|\mathcal{B}_e[Q_{e,1}(s,a)] - \mathcal{B}_e[Q_{e,2}(s,a)]\|_\infty &= \|\gamma[Q_{e,1}(s',a') - \alpha \log \pi(a'|s')] \\ &\quad - \gamma[Q_{e,2}(s',a') - \alpha \log \pi(a'|s')]\|_\infty \\ &\leq \|\gamma Q_{e,1}(s',a') - \gamma Q_{e,2}(s',a')\|_\infty \\ &= \gamma \|Q_{e,1}(s',a') - Q_{e,2}(s',a')\|_\infty.\end{aligned} \tag{25}$$

Since $\gamma \in (0,1)$, it follows that $\mathcal{B}_e$ is a γ-contraction mapping. By applying Lemma 1, we know that $\mathcal{B}_e$ has a unique fixed point. This fixed point can be obtained by iteratively applying $\mathcal{B}_e$ starting from an arbitrary initial $Q_{e,\text{init}}$. That is, as the iteration number k increases, the sequence of updated Q functions converges to a fixed point, i.e., the desired Q_e.

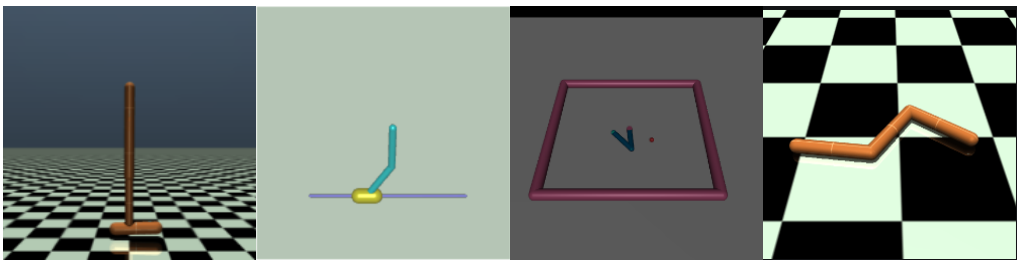
B Environmental Introduction

MuJoCo: This is a high-performance physics simulation platform widely adopted for robotic reinforcement learning research. The environment features efficient physics computation, accurate dynamic system modeling, and comprehensive support for articulated robots, making it an ideal benchmark for RL algorithm development.

In this paper, we concentrate on eight tasks: Humanoid-v3, Ant-v3, HalfCheetah-v3, Walker2d-v3, InvertedDoublePendulum-v3 (InvertedDP-v2), Hopper-v3, Reacher-v2, and Swimmer-v3, as illustrated in Figure 5. The InvertedDP-v3 task entails maintaining the balance of a double pendulum in an inverted state. In contrast, the objective of the other tasks is to maximize the forward velocity while avoiding falling. All these tasks are realized through the OpenAI Gym interface (Brockman et al., 2016).



(a) (b) (c) (d)



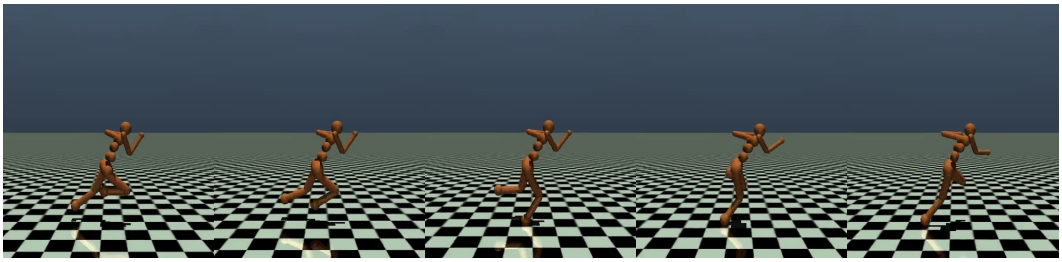
(e) (f) (g) (h)

Figure 5: Benchmarks. (a) Humanoid-v3: $(s \times a) \in \mathbb{R}^{376} \times \mathbb{R}^{17}$. (b) Ant-v3: $(s \times a) \in \mathbb{R}^{111} \times \mathbb{R}^{8}$. (c) HalfCheetah-v3: $(s \times a) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$. (d) Walker2d-v3: $(s \times a) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$. (e) Hopper-v3: $(s \times a) \in \mathbb{R}^{11} \times \mathbb{R}^{3}$. (f) InvertedDoublePendulum-v2: $(s \times a) \in \mathbb{R}^{6} \times \mathbb{R}^{1}$. (g) Reacher-v2: $(s \times a) \in \mathbb{R}^{11} \times \mathbb{R}^{2}$. (h) Swimmer-v3: $(s \times a) \in \mathbb{R}^{8} \times \mathbb{R}^{2}$.

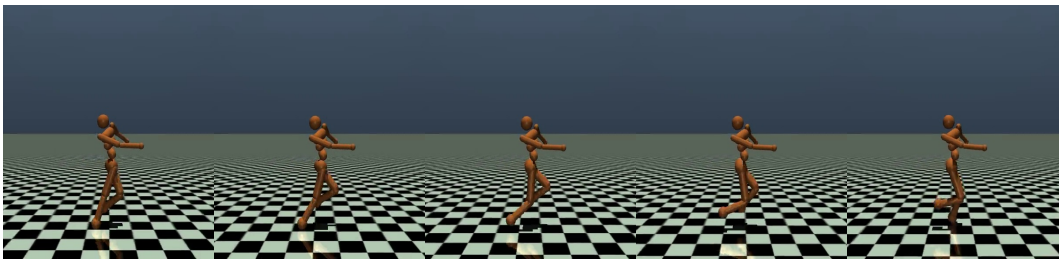
C VISUALIZATIONS

To demonstrate the effectiveness of DSAC-E in solving complex, high-dimensional locomotion tasks, we provide visualizations of policy control process on three of the most challenging benchmarks in the Humanoid task as shown in the following Figure 6. These tasks require precise coordination across many degrees of freedom and long-horizon reasoning.

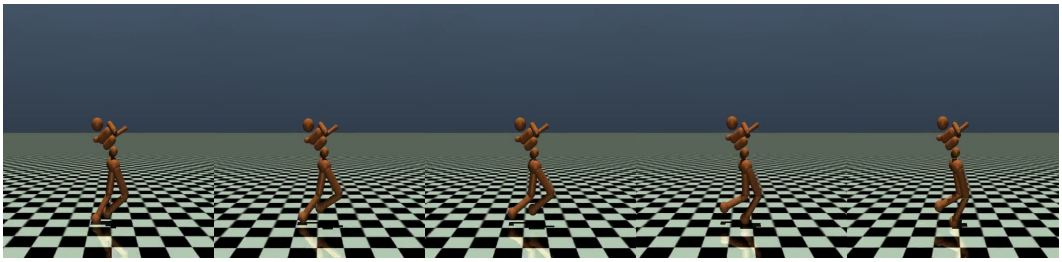
The visualization showcase that DSAC-E not only achieves successfully running but also learns robust posture and behaviors, highlighting its strong capabilities in difficult control scenarios.



(a) **DSAC-E** step 70 (b) step 72 (c) step 74 (d) step 76 (e) step 78



(f) **DSAC-T** step 70 (g) step 72 (h) step 74 (i) step 76 (j) step 78



(k) **SAC** step 70 (l) step 72 (m) step 74 (n) step 76 (o) step 78

Figure 6: Visualizations of control processes on Humanoid-v3 task.

D REPRODUCIBILITY STATEMENT

TABLE 4
DETAILED HYPERPARAMETERS.

Hyperparameters	Value
Shared	
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Actor learning rate	$1\mathrm{e}{-4}$
Critic learning rate	$1\mathrm{e}{-4}$
Discount factor (γ)	0.99
Policy update interval	2
Target smoothing coefficient (τ)	0.005
Reward scale	0.1
Number of iterations	1.5×10^6
Maximum-entropy framework	
Learning rate of temperature α	3×10^{-4}
Base expected entropy ($\overline{\mathcal{H}}$)	$\overline{\mathcal{H}} = -\dim(\mathcal{A})$
Deterministic policy	
Exploration noise	$\epsilon \sim \mathcal{N}(0, 0.1^2)$
Off-policy	
Sample batch size	20
Replay batch size	256
Replay buffer warm size	1×10^4
Replay buffer size	1×10^6
On-policy	
Sample batch size	2000
Replay batch size	2000
GAE factor	0.95
DSAC-T	
Variance clipping constant ζ	3
Stabilizing constant ϵ and ϵ_ω	0.1
DSAC-E (ours)	
ρ	20 for Humanoid and Walker2d, otherwise 1

Time efficiency. The CPU used for the experiment is the AMD Ryzen Threadripper 3960X 24-Core Processor, and the GPU is NVIDIA GeForce RTX 3090Ti. Taking Humanoid-v3 as an example, the time taken to train 1.5 million iterations using the JAX framework around is 2 hours.

E LLM USAGE DISCLOSURE

We used ChatGPT to polish grammar and improve text clarity. We reviewed all LLM-generated suggestions and are fully responsible for the final content of this paper.