2025 lEEE 14th Data Driven Control and Learning Systems Conference
May 9-11, 2025, Wuxi, China

An Efficient Human-in-the-loop Method for Stochastic Policy Reinforcement Learning

Guoqing Xie[1,2], Jianfei Wei[1,2], Longlong Wang[1,2], Guangyu Wang[1,2], Shuping He[1,2], and Liang Zhang*[1,2]
1. Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education
2. School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China
E-mail: liangzhang@ahu.edu.cn

Abstract: This study introduces Entropy-Q Human-in-the-Loop Reinforcement Learning (EQ-HRL), a novel human-in-the-loop reinforcement learning framework designed for continuous control tasks. To tackle the dual challenges of sample inefficiency and excessive reliance on expert input in interactive RL, we propose a dual-condition advising mechanism that integrates policy entropy analysis with Q-value discrepancy assessment. The framework utilizes dynamic thresholding to enable adaptive guidance triggering and incorporates a dual-experience replay architecture to enhance knowledge assimilation. Extensive experiments conducted across benchmark environments demonstrate that, within a constrained advice budget, the proposed method significantly reduces expert queries while maintaining competitive success rates. These results highlight the framework's ability to foster effective human-AI collaboration in complex decision-making scenarios by striking a balance between autonomous exploration and strategic expert intervention.

Key Words: Reinforcement learning, Human-in-the-loop, Policy entropy, Expert advice

2025 IEEE 14th Data Driven Control and Learning Systems (DDCLS) | 979-8-3503-5731-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/DDCLS66240.2025.11065639

1 Introduction

Within the field of machine learning, Reinforcement Learning (RL)[1] is a pivotal domain, focusing on the development of optimal strategies to maximize cumulative rewards through interactions between an intelligent agent and its environment. At the start of the 21st century, the fusion of deep learning and reinforcement learning led to the emergence of Deep Reinforcement Learning (Deep RL), enabling significant advancements in tackling intricate challenges. Prominent examples include DeepMind's DQN (Deep Q-Network), which outperformed human players in Atari games[2], and AlphaGo's triumph over the world champion in Go[3]. Beyond this, reinforcement learning has exhibited substantial capabilities in a wide range of disciplines, spanning robotics, autonomous navigation, financial strategies, and clinical decision support[4], [5]. Nevertheless, the practical deployment of RL remains hindered by several challenges. Chief among these is its sample inefficiency, especially in complex environments requiring extensive interaction data, which limits its effectiveness in real-world applications[6]. Additionally, algorithmic instability and convergence issues are particularly pronounced in deep RL, complicating the training process[7]. Furthermore, challenges such as hyperparameter sensitivity, sparse reward scenarios, and optimization difficulties in continuous action spaces further exacerbate the complexity of RL implementations[8]. To address these issues, researchers have proposed various enhancements, including experience replay[2], target networks[9], entropy regularization[6], and hierarchical reinforcement learning[10], aiming to improve algorithmic stability and efficiency. However, further advancements in both theoretical and practical aspects of RL are essential to unlock its potential across a broader spectrum of applications.

The dominant reinforcement learning algorithms in current use[11], [12], [13], [14] are fundamentally a posteriori in nature, as their training processes do not depend on the accumulation of a priori knowledge. While this characteristic underpins the superior performance of these algorithms, it also contributes to the pervasive issue of low sample efficiency in the field. The emergence of human-computer interaction machine learning research[15],[16], has given rise to Human-in-the-Loop reinforcement learning methods, which aim to enhance sample utilization efficiency within constrained training periods while establishing effective collaboration mechanisms between intelligent agents and human experts[17]. This approach has introduced a novel breakthrough direction in the field.

Early research on interactive reinforcement learning primarily focused on discrete action spaces. Seminal work by Torrey et al. developed an interactive learning framework based on action guidance[18], later formalized as a teacher-student model with a suggestion budget. This study marked a significant advancement by identifying contexts in which teachers can optimize student learning efficiency through action-guided suggestions. A major contribution of this study was the quantification of state importance as the extreme difference in Q-values between different actions within the same state, providing a quantitative basis for teacher intervention. Building on this, Amir et al. introduced an action selection uncertainty condition on the student side[19], refining the framework into a dual-condition mechanism. This approach, which balances state importance and action selection uncertainty, substantially improves the efficiency of suggestion budget utilization. Further progress has been made through the integration of uncertainty-aware mechanisms in reinforcement learning optimization. For instance, Silva et al. proposed a confidence modulation-based policy recommendation framework (RCMP)[20], which enhances sample efficiency by evaluating the cognitive uncertainty of an intelligent agent and requesting advice when un-

This work was supported in part by the State Key Laboratory of Micro-Spacecraft Rapid Design and Intelligent Cluster under grant No. MS01240110, the Anhui Provincial Key Research and Development Project under Grant 2022i01020013, the University Synergy Innovation Program of Anhui Province under Grant GXXT-2021-010, and the Anhui Higher Education Scientific Research Project under Grant 2022AH050068.

certainty is high. Similarly, Thakur et al. employed Bayesian neural networks (BNN) to quantify policy uncertainty in multi-task environments, aiming to improve data efficiency and task adaptation through proactive data requests[21]. These studies have opened new avenues for strategic learning optimization.

As reinforcement learning rapidly evolves in continuous action spaces, traditional methods based on discrete action Q-value polarity have struggled to accurately assess state importance and action selection uncertainty. To address this, Luo et al. proposed a novel suggestion mechanism tailored for continuous spaces[22]. Leveraging the dual-Q network property of the TD3 algorithm[12], they effectively provide action suggestions to agents when significant discrepancies arise in the outputs of the dual-critique network, thereby resolving the challenge of determining action selection uncertainty. However, a research gap remains in the assessment of state importance within continuous action spaces, offering valuable insights for our work.

Inspired by the concept of "urgent point" in the game of Go, we propose the EQ-HRL algorithm, which quantifies state importance by measuring the entropy of the action distribution generated by the policy network. States with low entropy action distributions are typically more significant, and we utilize this metric as an advising condition for the teacher, combined with the dual Q-network uncertainty condition. This ensures that more impactful action advice is provided during critical states, thereby enabling precise guidance of the decision-making process. Additionally, we introduce dynamic sequence thresholding and dual-experience replay mechanisms, which enhance the efficiency of leveraging important experiences while substantially conserving the advising budget. Empirical findings reveal that our approach attains performance on par with, or exceeding, current human-in-the-loop guidance techniques, as evidenced by reward trajectories and task completion rates, confirming the efficacy and refinement of the proposed methodology.

2 Preliminaries

2.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) serves as a mathematical framework for modeling decision-making in stochastic and controllable environments. Formally, an MDP is defined as a quintuple comprising states, actions, state transition probabilities, immediate rewards, and a discount factor:

$$\text{MDP} \sim (S, A, P, R, \gamma)$$

During the interaction with the environment, the agent's decision at each step depends solely on the current state. Specifically, the agent selects and executes an action based on its current policy, subsequently receiving an immediate reward through the reward function. The MDP framework incorporates a discount factor and a state transition probability function to balance immediate and future rewards. This process is iterative, with the system progressively refining its policy to approximate an optimal strategy that maximizes the expected cumulative reward. The ultimate goal is to identify a trajectory that optimizes the desired reward.

Two additional concepts are critical to the decision-making process: the value function and the action-value function. The value function $V_\pi(s)$ under a policy π represents the expected cumulative reward starting from state s and following π thereafter:

$$V^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s, \pi\right] \tag{1}$$

Similarly, the action-value function $Q_\pi(s, a)$ represents the expected cumulative reward starting from state s, taking action a, and following π thereafter:

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s, a_t = a, \pi\right] \tag{2}$$

These definitions establish the core principles for policy evaluation and improvement in reinforcement learning.

2.2 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy reinforcement learning algorithm designed to optimize stochastic policies by simultaneously maximizing expected cumulative rewards and policy entropy. By promoting high-entropy policies, SAC enhances exploration, improves sample efficiency, and ensures robustness. The optimization target is represented by the following objective function:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right] \tag{3}$$

where $\mathcal{H}(\pi(\cdot|s_t))$ represents the entropy of the policy, while α acts as a temperature coefficient that modulates the intensity of entropy regularization.

SAC adopts an actor-critic architecture, comprising the following components:

- Stochastic Policy (Actor): The policy $\pi_\phi(a_t|s_t)$ generates a probability distribution over actions and is optimized to maximize entropy-regularized rewards.
- Critic Networks: Two Q-networks, Q_{θ_1} and Q_{θ_2}, are employed to mitigate overestimation bias. The target Q-value y_t is computed as:

$$y_t = r_t + \gamma \mathbb{E}_{a_{t+1} \sim \pi_\phi}\left[Q_{\theta^-}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1})\right] \tag{4}$$

  The update of the Q-networks is achieved by minimizing the Bellman residual:

$$\mathcal{L}_Q(\theta_i) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}}\left[\left(Q_{\theta_i}(s_t, a_t) - y_t\right)^2\right] \tag{5}$$

  where $\mathcal{D}$ denotes the experience replay memory, while θ^- corresponds to the target network's parameters.
- Policy Optimization: The actor is updated to maximize:

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_\phi}\left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right] \tag{6}$$

SAC's integration of entropy regularization and double Q-learning renders it highly effective for continuous control tasks, achieving an optimal balance between exploration and exploitation.

2.3 Human-in-the-loop Reinforcement Learning

The interactive approach adopted for reinforcement learning in this paper is human-in-the-loop(HITL), where human experts provide interactive guidance during agent training. This methodology enables agents to learn not only from environmental rewards but also from interactive guidance, substantially enhancing the effectiveness of learning and policy optimization in complex settings.

The central premise of human-in-the-loop is to allow an external advisor to provide action recommendations or corrections during the agent's exploration. These interactions are selectively triggered based on predefined conditions, ensuring efficient and targeted use of external guidance. By combining autonomous exploration with strategic external feedback, HITL achieves faster convergence and more robust policies.

In this study, we leverage the principles of human-in-the-loop to enhance the agent's learning process, focusing on the effective integration of external feedback into the training framework.

3 EQ-HRL Algorithm Framework

3.1 Advising Condition

3.1.1 Early Advising

Given the difficulty for experts to provide real-time guidance continuously during the agent's training process, we introduce the "Early-Advising" budget mechanism[18]. This mechanism leverages a predefined advising budget threshold to fully utilize expert knowledge for guiding the agent in the initial stages of training, while gradually reducing expert intervention as the training progresses.

$$Iteration < AdvisingBudget \tag{7}$$

This approach not only effectively controls the cost of expert guidance but also fosters the agent's autonomous learning capabilities, achieving a smooth transition from dependence to independence.

3.1.2 Ask Condition: Dual Q-Network Discrepancy

The discrepancy between dual Q-networks serves as a measure of uncertainty in state-action value estimation[22]. In SAC, two independent Q-networks are employed to mitigate overestimation bias in Q-values. While the minimum Q-value is typically used for network updates, the two critic networks often produce divergent estimates during exploration. This discrepancy is quantified as:

$$\mathrm{D}(s) = |Q_{\theta_1}(s,a) - Q_{\theta_2}(s,a)|,\ a \sim \pi(\cdot|s) \tag{8}$$

When this difference exceeds a predefined threshold ($\mathrm{D}(s) > threshold$), it signals decision-making difficulties, prompting the agent to seek expert advice for policy refinement.

3.1.3 Response Condition: Policy Entropy

Under limited advice budgets, the expert cannot address all agent queries, as this may induce dependency and hinder convergence. To address this, we extend the discrete action space importance advising method[18], [19] to continuous action space stochastic policy reinforcement learning using policy entropy, defined as:

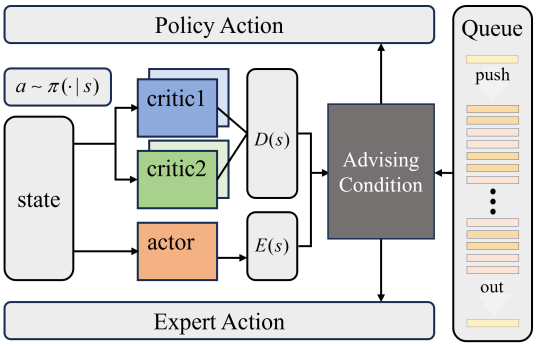


Fig. 1: Advising Condition of EQ-HRL

$$\mathrm{E}(s) = -\int \pi(a|s)\log\pi(a|s)\,da \tag{9}$$

For the expert, low policy entropy ($\mathrm{E}(s) < threshold$) indicates more deterministic actions in the current state, making critical decisions essential for optimal outcomes. By focusing on states with low policy entropy, the expert reduces dependency and conserves the advice budget.

3.2 Dynamic Threshold

To determine a precise threshold, we adopt a dynamic threshold method, consistent with existing literature[22]. This approach eliminates extensive experimental tuning by dynamically determining expert advice conditions. Specifically, two fixed-length FIFO queues store recent values of uncertainty measures (dual Q-network discrepancies) and importance measures (policy entropy). The uncertainty queue $\mathcal{D}_{queue}$ and the entropy queue $\mathcal{E}_{queue}$ are updated at each step as follows:

$$\mathcal{D}_{queue} \leftarrow \mathrm{FIFO}(\mathcal{D}_{queue}, D(s)),$$
$$\mathcal{E}_{queue} \leftarrow \mathrm{FIFO}(\mathcal{E}_{queue}, E(s)).$$

where $D(s)$ is the current dual-Q network discrepancy, $E(s)$ is the current policy entropy, and FIFO denotes the operation of adding a new value to the queue while maintaining its fixed length. At each step, the agent's current uncertainty D(s) is compared to the maximum value in the uncertainty queue $\mathcal{D}_{queue}$, while the expert's policy entropy $E(s)$ is compared to the minimum value in the entropy queue $\mathcal{E}_{queue}$. As illustrated in Fig. 1, expert advice is triggered if:

$$D(s) > \max(\mathcal{D}_{queue}) \quad \text{and} \quad E(s) < \min(\mathcal{E}_{queue}) \tag{10}$$

This dynamic threshold mechanism ensures expert advice is provided only when the agent exhibits high uncertainty and the expert demonstrates high confidence, improving guidance efficiency and relevance.

3.3 Policy Update

The proposed algorithm employs a dual experience replay mechanism to enhance learning efficiency, as illustrated in Fig. 2. The agent maintains two distinct buffers: a standard buffer for general exploration and an advising buffer

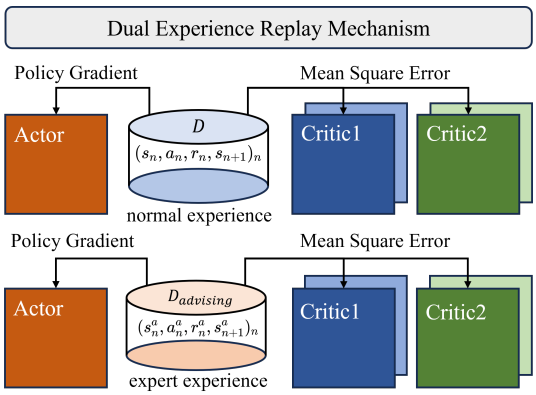


Fig. 2: Dual Buffer Mechanism of EQ-HRL

for storing expert-guided experiences. This dual-buffer design achieves a balance between autonomous learning and expert knowledge, enabling the algorithm to perform effectively in complex environments. During each update step, the policy is optimized through two stages. First, a batch of experiences from the standard buffer is used to update the policy network and critic network via policy gradient (PG) and mean squared error (MSE) respectively, fostering autonomous decision-making and ensuring the agent learns from its own exploration. Next, a batch from the advising buffer is employed to incorporate expert knowledge, accelerating learning in domains where exploration is less efficient or rewards are sparse.

The dual-update strategy ensures that the agent's exploration process remains intact while benefiting from expert guidance. By separating the updates, the algorithm prevents over-reliance on the advising buffer, preserving the agent's ability to explore independently while refining its decisions with expert insights. In summary, the dual experience replay mechanism achieves a balance between exploration and expert guidance, leading to faster convergence, higher sample efficiency, and greater robustness. This makes the algorithm well-suited for complex environments, where integrating expert knowledge accelerates learning while maintaining the agent's ability to adapt independently.

3.4 EQ-HRL Algorithm

Algorithm 1 EQ-HRL

Initialize parameters θ_1, θ_2, ϕ.
$\theta_1^- \leftarrow \theta_1, \theta_2^- \leftarrow \theta_2$.
Initialize empty replay buffer $\mathcal{D}$.
Initialize empty advising replay buffer $\mathcal{D}_{advising}$.
Initialize state importance queue $\mathcal{D}_{queue}$.
Initialize state importance queue $\mathcal{E}_{queue}$.
for each iteration **do**
  FLAG = False.
  for each step t $\in (1, T)$ **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$.
    get $D(s)$ and $E(s)$ according to Eq. 8 and Eq. 9.
    if $D(s) > \max(\mathcal{D}_{queue})$ and $E(s) < \min(\mathcal{E}_{queue})$ and $Iteration < AdvisingBudget$ **then**
      $\mathbf{a}_t \leftarrow \mathbf{a}_{human}$.
      FLAG $\leftarrow$ True.
    end if
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.
    if FLAG **then**
      $\mathcal{D}_{advising} \leftarrow \mathcal{D}_{advising} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$.
      FLAG $\leftarrow$ False.
    end if
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$.
    Sample $(s_n, a_n, r_n, s_{n+1})_{n=1\ to\ N}$ from $\mathcal{D}$.
    Sample $(s^a_n, a^a_n, r^a_n, s^a_{n+1})_{n=1\ to\ N}$ from $\mathcal{D}_{advising}$.
    get y_n and y^a_n according to Eq. 4.
    1st. Critic Update:
    $L(\theta_i) = \frac{1}{N}\sum_{n=1}^{N}(y_n - Q_{\theta_i}(s_n, a_n))$.
    2nd. Critic Update:
    $L(\theta_i) = \frac{1}{N}\sum_{n=1}^{N}(y^a_n - Q_{\theta_i}(s^a_n, a^a_n))$.
    Actor Update:
    $L(\phi) = \frac{1}{N}\sum_{n=1}^{N}(\alpha log\pi_\theta(a_n|s_n) - \min_{i=1,2} Q_{\theta_i}(s_n, a_n))$.
    $\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-, i = 1, 2$.
  end for
end for

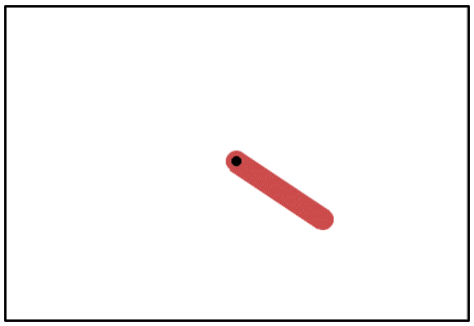
4 Experiments

4.1 Environment Setup

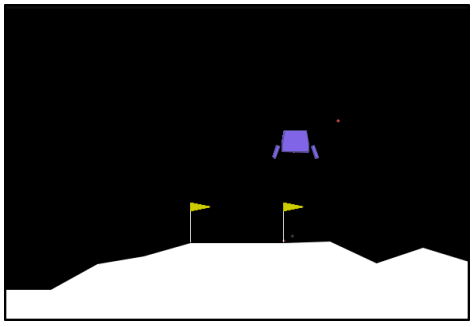
The experiments were carried out on our workstation, which is powered by an Intel Core i9-14900KF CPU, an NVIDIA GeForce RTX 4090 GPU, and 64 GB of RAM, with Ubuntu 20.04 as the operating system. The implementation was developed using Python 3.9, CUDA 12.1, and PyTorch 2.5.1. The reinforcement learning environments were simulated using OpenAI Gymnasium 1.0.0 and Box2D-py 2.3.5. Key hyperparameters are summarized in Table 1.

4.2 Algorithm Comparison

- SAC: A baseline off-policy RL algorithm[13] optimizing stochastic policies with entropy regularization.
- QASAC: Extends SAC with dual-Q discrepancy-based advising[22] and dual experience replay for efficient expert guidance.



(a) Pendulum-v1



(b) LunarlanderContinuous-v3

Fig. 3: Experiment Environments

Table 1: Hyperparameter Settings

Parameter	Value
Hidden layers	256×256
Learning rate (A-net)	1×10^{-4}
Learning rate (C-net)	1×10^{-3}
Learning rate (α)	1×10^{-4}
Target α	Negative action dim
Random seed	42
Discount factor (γ)	0.99
Minimal buffer size	5,000
Replay buffer size	500,000
Soft update rate (τ)	0.005
$\mathcal{D}/\mathcal{E}_{queue}$ sequence length	15
Critic activation function	ReLU
Actor activation function	Tanh
Weight initialization	Kaiming
Filter	Savitzky-Golay

- EQSAC: Building upon the EQ-HRL framework, EQSAC introduces policy entropy as an additional advising condition, ensuring targeted and efficient expert intervention.

4.3 Pendulum-v1

The Pendulum-v1 environment (Fig. 3a) simulates a simple pendulum system, where the goal is to swing the pendulum to an upright position and maintain it with minimal control effort. The state space comprises the pendulum's angle and angular velocity, while the action space is unbroken, indicating the force delivered to the pendulum. The reward function penalizes deviations from the upright position and control input magnitude, encouraging stability with minimal energy. This environment is widely used for testing continuous control algorithms due to its simplicity and clear optimization objectives.

Table 2: Experimental Settings and Results on Pendulum

Parameter/Result	Value
Number of Experiments	5
Total Episodes	200
Max Steps per Episode	100
Early Advising Budget	0-30
SAC Average Advice Count	0
QASAC Average Advice Count	316
EQSAC Average Advice Count	29

The experimental configuration for Pendulum is detailed in Table 2. Over 200 episodes, SAC, QASAC, and EQSAC algorithms all demonstrated rapid reward ascent in lower-dimensional settings. As expert experience was incorporated, Fig. 4 shows that QASAC and EQSAC achieved faster reward escalation and convergence compared to standard SAC. Notably, EQSAC exhibited significantly fewer query-advice requests than QASAC, as shown in Fig. 5, with its average total advice requests being merely 1/11th of QASAC's. Furthermore, EQSAC's task success rate (Fig. 6) consistently approached and exceeded that of QASAC, significantly outperforming SAC.

4.4 LunarLanderContinuous-v3

The LunarLanderContinuous-v3 environment in Fig. 3b models a lunar landing scenario, where the agent controls a spacecraft to land safely on a landing pad. The state space includes the lander's position, velocity, angle, angular velocity, and ground contact information. The action space is defined by continuous values, which denote the force generated by the lander's engines. The reward function encourages smooth and efficient landing by penalizing excessive fuel consumption, high velocities, and deviations from the landing pad. This environment is more complex than Pendulum-v1, offering a challenging testbed to assess the robustness and generalization capabilities of reinforcement learning algorithms.

Table 3: Experimental Settings and Results on Lunarlander

Parameter/Result	Value
Number of Experiments	10
Total Episodes	1000
Max Steps per Episode	600
Early Advising Budget	0-300
SAC Average Advice Count	0
QASAC Average Advice Count	9017
EQSAC Average Advice Count	1390

In the higher-dimensional LunarLander environment, EQSAC demonstrated even more pronounced advantages, as detailed in Table III. During the 1000-episode training process, EQSAC achieved reward curve levels comparable to QASAC with only 1,390 advice queries, far fewer than QASAC's average of 9,017. The superior stability and lower standard deviation of EQSAC, as shown in Fig. 7, further highlight its robustness. Fig. 8, which presents a heatmap of per-episode advice queries across multiple experimental runs, reveals that EQSAC's advice requests were both fewer and more evenly distributed. Additionally, EQSAC outperformed SAC and QASAC in success rate range, median, and other metrics, as illustrated in Fig. 9.

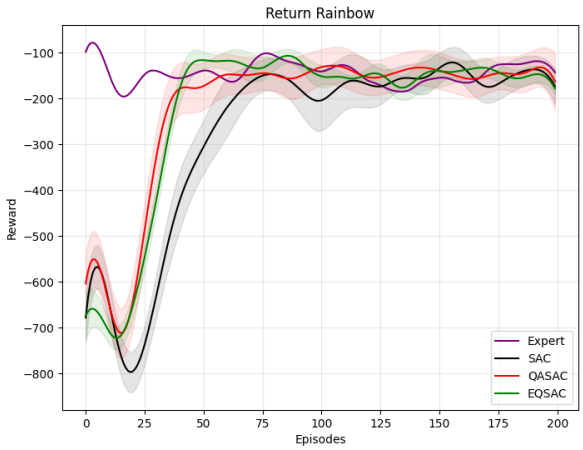


Fig. 4: Returns on Pendulum

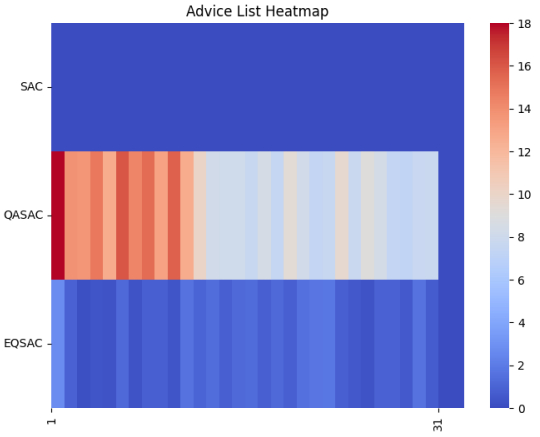


Fig. 5: Advised amount per episode on Pendulum

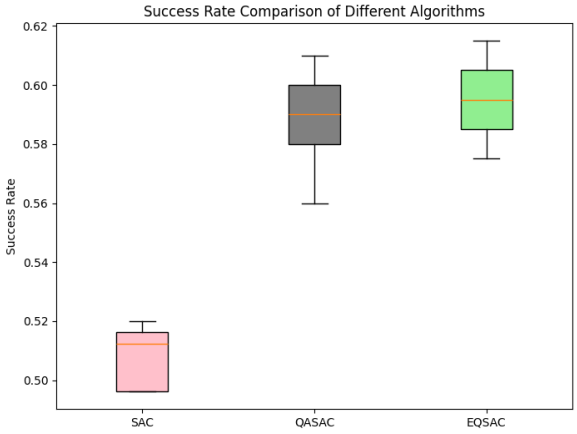


Fig. 6: Success rate on Pendulum

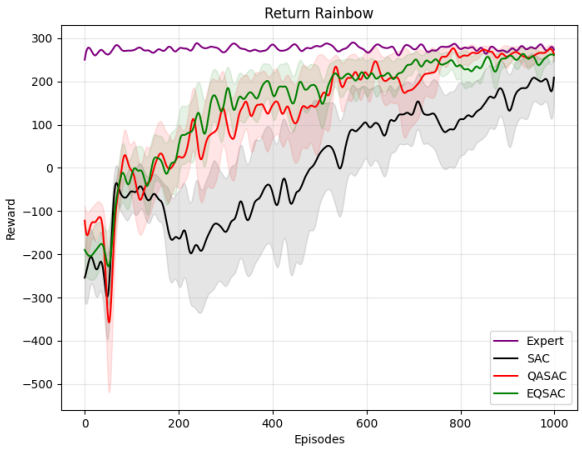


Fig. 7: Returns on Lunarlander

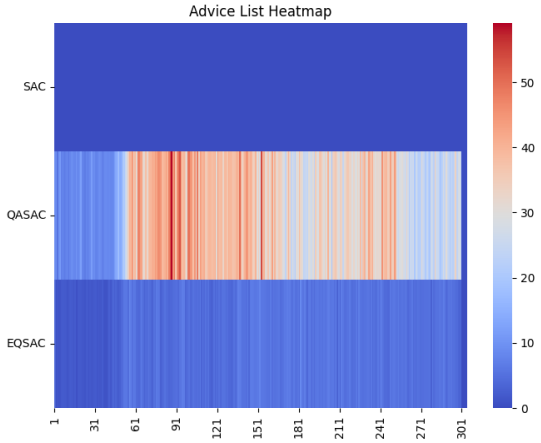


Fig. 8: Advised amount per episode on Lunarlander

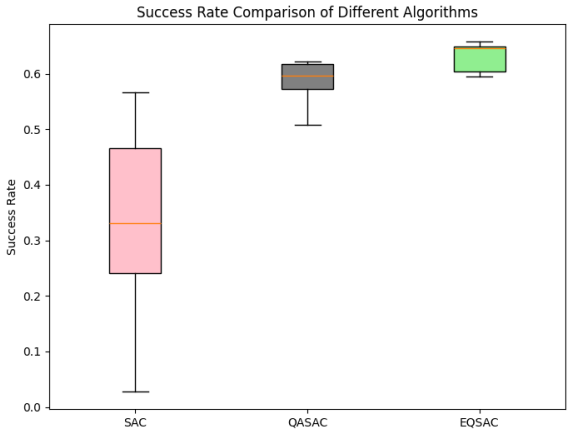


Fig. 9: Success rate on Lunarlander

5 Conclusion

Our proposed human-in-the-loop RL framework effectively addresses key challenges in reinforcement learning by integrating expert advice in a targeted and efficient manner. The dynamic threshold mechanism and dual experience replay strategy enable the agent to achieve high performance with minimal expert intervention, demonstrating robustness and sample efficiency. Experimental results across multiple benchmark environments validate the framework's ability to reduce expert queries while maintaining competitive task performance. Additionally, the introduction of policy entropy as an advising condition ensures precise and timely expert intervention, further enhancing learning efficiency. Future work could explore extending this framework to multi-agent systems, where cooperative learning and shared expertise could further enhance performance. Furthermore, investigating adaptive mechanisms for varying advice budgets and integrating diverse forms of human feedback, such as natural language instructions, could increase the adaptability of our method in real-world applications.

References

[1] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[2] Volodymyr Mnih, Koray Kavukcuoglu, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[3] David Silver, Aja Huang, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[4] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 2013.

[5] Sergey Levine, Chelsea Finn, et al. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2015.

[6] Tuomas Haarnoja, Aurick Zhou, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018.

[7] Peter Henderson, Riashat Islam, et al. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2017.

[8] Timothy P. Lillicrap, Jonathan J. Hunt, et al. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[9] H. V. Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, 2015.

[10] Ofir Nachum, Shixiang Shane Gu, et al. Data-efficient hierarchical reinforcement learning. In *Neural Information Processing Systems*, 2018.

[11] John Schulman, Filip Wolski, et al. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[12] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

[13] Tuomas Haarnoja, Aurick Zhou, et al. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018.

[14] Arsenii Kuznetsov, Pavel Shvechikov, et al. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, 2020.

[15] Guangliang Li, Randy Gomez, et al. Human-centered reinforcement learning: A survey. *IEEE Transactions on Human-Machine Systems*, 49(4):337–349, 2019.

[16] Georgios Fragkos, Jay Johnson, et al. Dynamic role-based access control policy for smart grid applications: An offline deep reinforcement learning approach. *IEEE Transactions on Human-Machine Systems*, 52(4):761–773, 2022.

[17] W. Bradley Knox and Peter Stone. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pages 292–297, 2008.

[18] Lisa A. Torrey and Matthew E. Taylor. Teaching on a budget: agents advising agents in reinforcement learning. In *Adaptive Agents and Multi-Agent Systems*, 2013.

[19] Ofra Amir, Ece Kamar, et al. Interactive teaching strategies for agent training. In *International Joint Conference on Artificial Intelligence*, 2016.

[20] Felipe Leno da Silva, Pablo Hernandez-Leal, et al. Uncertainty-aware action advising for deep reinforcement learning agents. In *AAAI Conference on Artificial Intelligence*, 2020.

[21] Sanjay Thakur, Herke van Hoof, et al. Uncertainty aware learning from demonstrations in multiple contexts using bayesian neural networks. *2019 International Conference on Robotics and Automation (ICRA)*, pages 768–774, 2019.

[22] Biao Luo, Zhengke Wu, Fei Zhou, and Bing-Chuan Wang. Human-in-the-loop reinforcement learning in continuous-action space. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):15735–15744, 2024.