Human-in-the-Loop Reinforcement Learning in Continuous-Action Space

Biao Luo, *Senior Member, IEEE*, Zhengke Wu, Fei Zhou, and Bing-Chuan Wang

***Abstract*— Human-in-the-loop for reinforcement learning (RL) is usually employed to overcome the challenge of sample inefficiency, in which the human expert provides advice for the agent when necessary. The current human-in-the-loop RL (HRL) results mainly focus on discrete action space. In this article, we propose a Q value-dependent policy (QDP)-based HRL (QDP-HRL) algorithm for continuous action space. Considering the cognitive costs of human monitoring, the human expert only selectively gives advice in the early stage of agent learning, where the agent implements human-advised action instead. The QDP framework is adapted to the twin delayed deep deterministic policy gradient algorithm (TD3) in this article for the convenience of comparison with the state-of-the-art TD3. Specifically, the human expert in the QDP-HRL considers giving advice in the case that the difference between the twin Q-networks' output exceeds the maximum difference in the current queue. Moreover, to guide the update of the critic network, the advantage loss function is developed using expert experience and agent policy, which provides the learning direction for the QDP-HRL algorithm to some extent. To verify the effectiveness of QDP-HRL, the experiments are conducted on several continuous action space tasks in the OpenAI gym environment, and the results demonstrate that QDP-HRL greatly improves learning speed and performance.**

***Index Terms*— Continuous action space, human-in-the-loop, reinforcement learning (RL).**

I. Introduction

REINFORCEMENT learning (RL) [1] collects huge amounts of data by interacting with the environment and utilizes the data to train the agent, which aims to maximize the cumulative reward. Due to its powerful reasoning ability, RL has achieved great success in many complex problems, such as AlphaGo [2] in the field of Go, OpenAI Five [3], and AlphaStar [4] in the field of games. RL is considered an important method to achieve artificial general intelligence [5], [6]. However, one common challenge of the current RL method is inefficient sampling and inadequate exploration when training the agent. For traditional RL algorithms, it requires a lot of trial-and-error learning before achieving the task goal for the first time. In fact, the most recent and popular RL methods [7], [8], [9] also require a large amount of training data to learn the optimal policy, where the main reason is the state and action spaces are usually high-dimensional or continuous [10], [11], [12] in practice. Without any prior knowledge, an RL algorithm trains an agent that involves a large number of parameters from the beginning, for example, deep RL (DRL) requires tens of millions of frames of pictures to achieve good performance in Atari games [13].

For most existing RL algorithms, human experts' prior knowledge is not fully considered and utilized during the learning process. However, human experts have a good understanding of the tasks performed by the agent, which is extremely helpful in improving exploration efficiency and reducing the frequency of trial and error [14], [15], [16]. Therefore, it is important and promising to make full use of human experts' experience to overcome the sampling inefficiency difficulty that widely exists in the RL research community [17]. In recent years, interactive human–machine collaborative machine-learning methods have attracted great attention [18], [19], [20], where the human-in-the-loop RL (HRL) has been demonstrated as an effective way to address sampling inefficiency and obtain good policies while reducing training time [21], [22], [23], [24] at the same time. The key idea of HRL is establishing a feedback model between the learning agent and human experts. Generally speaking, human experts participate in the training of agents through different feedback methods. The first is learning directly from the human expert policy without environmental reward signals [25]. Another is the human expert providing a good/bad judgment on the current state–action pair for the agent [26], [27], [28], where the human feedback is interpreted as a guideline for the agent's current policy to follow, not as a value for the agent to maximize.

To the best of our knowledge, most existing results on HRL mainly focus on discrete action space [29], [30], [31], [32]. Developing HRL methods for continuous action space remains an open issue waiting for systematical study. The difficulty that HRL methods in discrete action space cannot be directly applied to continuous action space is mainly because the number of actions in discrete space is finite, which is essentially different from the continuous action space with infinite actions. To overcome this difficulty, in this article, a Q value-dependent policy (QDP)-based HRL framework is developed for the continuous action space, where the QDP mechanism is proposed to seek human experts' advice and train the critic network. The essence of the QDP is that the human expert's action usually performs better than the agent's

Manuscript received 7 June 2022; revised 19 February 2023; accepted 20 June 2023. Date of publication 7 July 2023; date of current version 30 October 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62022094 and Grant 61873350, in part by the Zhejiang Laboratory under Grant 2021NB0AB01, and in part by the Hunan Provincial Natural Science Foundation of China under Grant 2020JJ2049. *(Corresponding author: Biao Luo.)*

The authors are with the School of Automation, Central South University, Changsha 410083, China (e-mail: biao.luo@hotmail.com; wuzhengkecsu@gmail.com; feizhoucsu@csu.edu.cn; bingcwang@csu.edu).

Digital Object Identifier 10.1109/TNNLS.2023.3289315

action based on current policy, which implies that it is better to use the human expert's action in the early training process rather than the agent's policy. Specifically, the importance of state–action pairs is evaluated based on the Q value difference, which is used as the basis for the agent to actively seek human experts' advice. Second, by using the gradient ascent method with expert experience, it can provide a potential direction for the update of the critic network, which means that both interactive data and expert experience can be employed for learning in developed QDP-based HRL (QDP-HRL).

The main contributions of this article can be briefly summarized in threefold. First, the novel QDP-HRL method is developed for continuous action space. Second, an active learning framework is proposed for the agent to seek advice actively from human experts in crucial states at the early training stage. Thus, the human expert is not required to give continuous supervision in the learning process, which reduces the costs of human experts' monitoring. Finally, the double experience buffer pool method is adopted to improve the learning efficiency of the agent.

II. Related Works

Over the past few years, HRL methods have received great attention and are regarded as a promising research direction in the RL community. Generally speaking, HRL aims to utilize human experts' intelligent knowledge to guide the agents' learning. According to our best knowledge, the HRL methods can be classified into three categories: TAMER methods [33], [34], [35], [36], learning from human's preferences [14], [17], [37], [38], [39], [40], and learning with human's advice [41], [42], [43], [44], [45]. For the TAMER methods, human feedback is used to train the reward function $R(s, a)$, and the agent does not receive reward signals from the environment. It is worth mentioning that the agent's strategy in the TAMER framework is short-sighted, which means only focusing on short-term rewards. Based on the TAMER framework, a planning algorithm [46] is proposed to maximize the cumulative reward function, where the environment model is required, and some restrictions are involved in termination conditions. By using a deep neural network for function approximation, the TAMER framework was extended to high-dimensional state spaces [35], [36]. Note that these works [33], [34], [35], [36] mainly concern discrete action spaces. For the HRL methods that learn from human preferences, the human preferences are linked to RL through feedback [17], [37], [38]. The agent randomly selects two segments from the experience buffer pool, and the human gives good, bad, or equal feedback to these two trajectory segments according to their preference. The agent uses the trajectory segments and the feedback data to train a reward predictor. These works [14], [17], [37], [38], [39], [40] utilize human experts' preferences rather than their prior knowledge, while reward signals from the environment are ignored. In addition, some works [47], [48] use human experts' demonstration data for learning. Although they can reduce the energy consumption of human experts, they will also bring compounding errors to the agent and affect the learning efficiency of the agent. In these works [49], [50], [51], human expert data can be used to solve sparse reward environments, the stability and learning efficiency of the agent are not satisfactory, and the generalization of hyperparameters is difficult. In this article, the proposed QDP involves human experts in the learning of the agent, giving them more opportunities to correct dangerous behaviors during the agent's learning process and broaden the channels of human–computer interaction. It enables agents to perform well even when experts are suboptimal. For the HRL methods that learn from human advice, the human advice is interpreted as labels for policies. Then, the policy is improved by using the probabilistic model on the difference between the agent's action and human advice. In [41], the difference $I(s) = \max_a Q(s, a) - \min_a Q(s, a)$ is used to evaluate the importance of actions, where the agent seeks a human's advice if $I(s)$ exceeds a predetermined threshold. The uncertainty of the state is evaluated using a deep Q-network with multiple heads [42] and a Bayesian network [43]. Then, the agent seeks advice from humans when the variance of the value $Q(s, a)$ exceeds a threshold. In addition, the works [44], [45] also learn from the behavior of human experts to accelerate the learning process.

However, most of these works were designed for discrete action spaces with numerous actions. For continuous action space, the problem becomes extremely difficult since there are infinite actions. It is still an open issue for HRL on continuous action space, which motivates our current study. To address this problem, the QDP-HRL method is developed in this article. The QDP strategy is proposed for seeking human advice and training, and a new advantage function is introduced based on human advice.

III. Preliminaries

In this section, some necessary preliminaries are presented. To begin with, the Markov decision process (MDP) is given, which is the standard problem for RL algorithms. Subsequently, the twin delayed deep deterministic policy gradient algorithm (TD3) [52] is introduced, which is used as a representative algorithm to intergrade our QDP strategy in this article. Finally, the basic idea of active learning is discussed, which is usually used to determine when human advice is required.

A. Markov Decision Process

RL is usually used to solve sequential decision problems, which can be modeled as an MDP [53]. MDP is represented by with $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, \gamma)$, where $\mathcal{S}$ denotes the state space, $s \in \mathcal{S}$ is the state, $\mathcal{A}$ is the action space, $a \in \mathcal{A}$ is the action, $\mathcal{R}$ means the reward function space, $r \in \mathcal{R}$ is the instant reward, T is the state transition function of the environment, and $\gamma \in [0, 1]$ represents the discount factor. $\pi(s) : \mathcal{S} \to \mathcal{A}$ is the agent's policy that is a mapping from state space to action space. The goal of the agent is to find an optimal policy $\pi^*(s)$ in which the cumulative reward $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is maximized. The state transition function T is unknown for the agent for model-free RL algorithms. Hence, the agent should interact with the environment to collect samples as the type of tuple (s, a, r, s'), which is utilized for training the agent. For

a policy $\pi(s)$, there are two types of value functions, that is, the action-value function $Q^{\pi}(s,a)$ and the state-value function $V^{\pi}(s)$. $Q^{\pi}(s,a)$ is defined as the expected return by following the policy $\pi(s)$ after taking the action a at the current state s, that is,

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1}\middle| s_t = s, a_t = a\right]. \tag{1}$$

$V^{\pi}(s)$ is defined as the expected return that by following policy $\pi(s)$ at state s, that is,

$$V^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1}\middle| s_t = s\right]. \tag{2}$$

B. TD3 Algorithm

The TD3 algorithm [52] is well suited for continuous action space, which is designed based on the actor-critic framework. It greatly reduces the overestimation problem that widely exists in RL methods. In the TD3, two critic networks are employed to estimate $Q(s,a)$, and the smaller estimation values are used as the target value, which reduces the deviation of Q value estimation to some extent. With training, the Q value estimation will become more accurate, and finally, the two evaluations converge to the optimal Q function according to the Bellman equation. The loss function of the critic network is given by

$$\begin{cases} y_i = r + \gamma Q_{\theta_i'}(s', \tilde{a}) \\ \text{loss}^{Q} = N^{-1}\sum\left(\min_{i=1,2} y_i - Q_{\theta_i}(s,a)\right)^2 \end{cases} \tag{3}$$

where $Q_{\theta_i'}(s,a)$ represents the target-critic network, y_i is the updating target, i represents one of the two critical networks with a value of 1 or 2, and $\tilde{a}$ means the output of target policy network $\pi_{\phi'}(s)$ by adding the noise ε, respectively. The noise ε is sampled from the Gaussian distribution. Because the Q value of similar actions in the same state is similar in the continuous action space, the purpose of adding noise is to make the critic network smoother. The actor-network adopts delayed update technology to update parameters by gradient ascent. The update process is given as follows:

$$\nabla_{\phi} J(\phi) = \frac{\sum \nabla_a Q_{\theta_1}(s,a)\big|_{a=\pi_{\phi}(s)} \nabla_{\phi}\pi_{\phi}(s)}{N} \tag{4}$$

where $Q_{\theta_1}(s,a)$ represents the critic network, a is the output of actor-network $\pi_{\phi}(s)$, and N is the batch size. The neural network weights of the target network are obtained by the soft update of the training network, and the soft update process of the target network is given as follows:

$$\begin{aligned} \theta_i' &\leftarrow \tau\theta_i + (1-\tau)\theta_i' \\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi' \end{aligned} \tag{5}$$

where θ_i' is the parameter of the target-critic network, and ϕ' is the parameter of the target policy network. The setting of the target network makes the agent more stable in the training process.

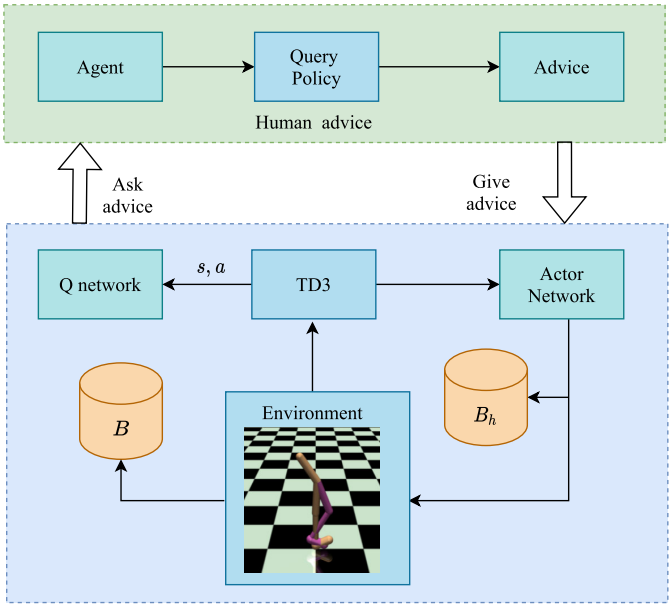

Fig. 1. Architecture of QDP. Train Q network and actor-network according to the agent interaction data stored in B and the human expert experience stored in B_h.

C. Active Learning

Active learning [54], [55], [56] is a machine-learning method used to reduce the labeling cost of human experts during training. Specifically, the model determines whether the current data needs to be labeled by experts according to the query strategy, and the query strategy is designed based on the uncertainty of the current model. If the predetermined condition is satisfied, the model will actively ask the human for the current data's label, reducing the unnecessary labeling cost. The training data with high uncertainty is given priority to human experts for labeling, and the training data is obtained during repeated interactions with experts. Obviously, the active learning method can reduce the workload of human annotation. In this article, the idea of active learning is also adopted in the developed QDP-HRL, where the agent actively asks the human experts what actions they should take in the current state according to the QDP query strategy to obtain the experience of the human experts. In the training process, the experience of human experts is integrated into the policy of the agent, thereby improving the training efficiency of the agent.

IV. QDP-Based HRL

In this section, we propose a novel QDP-HRL algorithm to address the challenge of low sampling efficiency of agents in continuous action space. The architecture of QDP is shown in Fig. 1, which is divided into two parts. The first part is the action selection strategy based on active learning. The agent actively seeks advice from human experts according to the query strategy in the early stage of training. Advice from human experts is stored in the experience buffer pool B_h, and the agent's interaction data is stored in B. The second part uses the experience (s, a_h) of human experts and the agent's interactive data (s, a, r, s') to train the critic network so that the critic network can be updated directionally to improve learning efficiency.

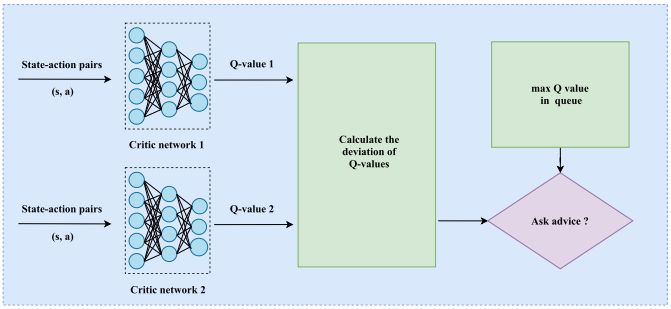

Fig. 2. According to the query strategy, the agent decides when to seek advice from human experts.

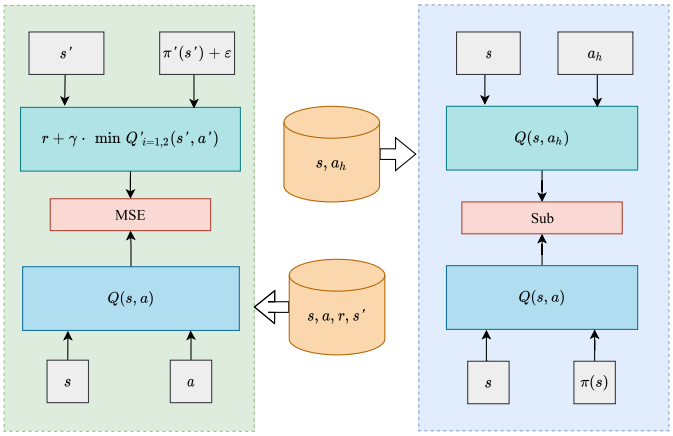

Fig. 3. Training process of the critic network uses the human expert experience and the interactive data between the agent and the environment. The expert data guides the updated direction of the critic network. The interactive data evaluates the state–action pairs of the critic network more and more accurately.

A. Action Query Strategy

Action query strategy is an important part of HRL, which directly determines when the agent seeks advice from the human expert. In the QDP, we propose a query strategy adapted to continuous action spaces based on active learning. First, the agent computes the critic network difference for the current state–action pair with

$$I(s) = Q_1(s,a) - Q_2(s,a) \tag{6}$$

where $Q_{i=1,2}(s,a)$ are the expected returns of the current critic networks for the state–action pair (s,a), and $I(s)$ represents the difference between the current-critic network estimates. Due to the distribution of $I(s)$ varying greatly in different tasks, it is impossible to set a fixed threshold, so we adopt the sliding window idea to store the most recent $I(s)$ in a fixed-length queue Que_n. When $I(s)$ satisfies the following condition:

$$I(s) > \max(\text{Que}_n) \tag{7}$$

that is, the agent asks for human advice if $I(s)$ exceeds the maximum value of the elements in the current queue. When the queue length exceeds the fixed length n, the first element to enter the queue is the first to go out. The action query strategy is shown in Fig. 2. Generally speaking, the value of $I(s)$ reflects the inaccuracy of the critic network estimation. Thus, the larger $I(s)$ implies that the current exploration of the agent is maybe inappropriate. Human experts' experience may help the agent choose appropriate actions in this situation. To guarantee the agent's exploration ability for the environment, human experts will only advise on the early stage of training by using the following method:

$$R_{\text{episode}} < I_R \tag{8}$$

where R_{episode} is the episode accumulate reward, $I_R = R_{\max}/t_h$ is the threshold, $R_{\max}$ is the maximum accumulate reward that can be given as appropriate value based on the task and experience, and t_h is an adjustable parameter represents the degree of human participation in the training of the agent and determines when humans stop giving advice to the agent. Because the parameter I_R plays a role in achieving a tradeoff between using human advice and encouraging exploration. A small I_R means that human advice has less chance to participate in learning to improve performance, while exploration is encouraged. In contrast, a large I_R results in human advice playing a more important role in the learning process, while the exploration level will be reduced. Thus, the parameter I_R provides more flexibility for the HRL algorithm based on the practical requirement. Another merit of (8) is that it can reduce human monitoring costs. When the HRL algorithm has already achieved a good convergence, human advice is not required, and exploration of the environment is encouraged to find potential better policies practical.

B. Policy Training

Considering TD3 is an actor-critic framework algorithm with excellent performance in continuous action space, we implement the QDP strategy based on TD3 as an illustration. It is worth pointing out that the proposed QDP can be combined with other DRL algorithms with a pre-condition that the algorithm includes two evaluation networks, for example, DDPG, SAC, and so on. The main reason is that the query strategy in the QDP relies on two evaluation networks. In the combination process, it requires increasing the query strategy in the DRL algorithms and the advantage loss function in the training process. In the actor-critic framework, the more accurate the critic network estimation is, the more beneficence the actor-network will gain. Therefore, we use the advantage loss function defined in QDP to improve the convergence speed of the critic network, thereby improving the performance of the actor network. The experience buffer pool B_h stores the human's advice (s,a_h), where a_h is a human's action regarded as the suboptimal action at state s. For the case that the critic network $Q(s,a)$ converges to its optimum $Q^*(s,a)$, the human's action a_h usually results in a larger Q value than other actions a, that is, $Q^*(s,a_h) > Q^*(s,a)$. Based on this idea, we define the new advantage loss function $A_i(s,a)$ by using the experience of a human expert, which is given by

$$A_i(s,a) = \frac{\sum [Q_i(s,a_h) - Q_i(s,\pi(s))]}{N} \tag{9}$$

where $Q(s,\pi(s))$ is the expected return of the agent's action in the state s by following the policy $\pi(s)$, $Q(s,a_h)$ represents the expected return of human expert experience and its training data are sampled from the human expert experience buffer

pool B_h, and N is the batch size. The advantage loss function represents the difference between the expert's experience and the agent's current policy. Considering the expert experience is usually better than the current action, we use the gradient ascent to improve the Q value of human expert experience and provide a direction for updating the critic network. On the other hand, the critic network is updated based on the following TD error:

$$\text{loss}^Q = \frac{\sum \left(\min_{i=1,2} y_i - Q_{\theta_i}(s,a)\right)^2}{N} \tag{10}$$

where y_i represents the minimum value of the target network given by

$$\begin{aligned} y_i &= r + \gamma Q_{\theta_i'}\left(s', \pi_{\phi'}(s') + \varepsilon\right) \\ \varepsilon &\sim \text{clip}(N(0,\sigma), -c, c). \end{aligned} \tag{11}$$

The training process of the critic network is shown in Fig. 3. Using the target network mechanism and the minimum target value mechanism can effectively reduce the influence of the bias in the Q value estimation on the algorithm. The advantage loss function provides a direction for updating the critic network. In the training process, the purpose of the advantage loss function is to improve the Q value at (s, a_h), which is the criterion for guiding the update of the critic network to improve the Q value. Because the actor network is constantly updated, the action a taken by the agent in the state s will be closer to a_h. To adapt the advantage loss function to the policy, we use the difference of $Q_1(s,a_h) - Q_1(s,\pi(s))$. It is found that using the critic network Q_1 only can improve the algorithm's performance and stability, where the main reason may result from the use of (4) for updating the actor network based on Q_1. The TD error makes the critic network estimate the current state–action pair's Q value more accurately. Because TD error is the part that plays an essential role in the update of the critic network, the advantage loss function will be limited during training to make it only fine-tune the critic network. Each training-critic network will be updated according to the advantage loss function and TD error, in which the training data for calculating loss^Q is sampled from the buffer pool B. The actor network is updated with

$$\nabla_\phi J(\phi) = \frac{\sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)}{N}. \tag{12}$$

The use of human expert experience improves the convergence speed of the critic network, and the more accurate the critic network is, the more favorable it is for the actor network, which indirectly affects the actor network and improves its training speed.

The advantage loss function (9) defined in QDP captures the relative effect of the Q value action of the human expert and the Q value of the agent's action. During the training process, the agent's policy is constantly changing, and the agent can timely correct the updating direction of the critic network according to the advantage loss function. Note that the QDP-HRL algorithm is suitable for continuous action space since it only uses the actions generated by the policy without knowing the Q value of all action–state pairs. The detailed QDP-HRL algorithm is described in Algorithm 1.

Algorithm 1 QDP-HRL Algorithm

Initialize critic networks Q_{θ_1}, Q_{θ_2} and actor network π_ϕ with random parameters θ_1, θ_2 and ϕ.
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$.
Initialize replay buffer B and human replay buffer B_h.
Initialize the Q value queue Que_n.

for $t = 1 \rightarrow t = T$ **do**
  select action $a \sim \pi_\phi(s) + N(0,\sigma)$.
  compute $I(s)$ according to equation (6).
  if $I(s) > max(Que)$ and $R_{episode} < I_R$ **then**
    If the human expert gives advice a_h, execute a_h, otherwise execute the action a.
  end if
  observe reward r and new state s'.
  store transition tuple (s, a, r, s') in B.
  store transition tuple (s, a_h) in B_h.

  begin training:
  TD error:
  sample mini-batch of N transitions from B.
  $a' = \pi_{\phi'}(s') + \varepsilon$;
  $\varepsilon \sim clip(N(0,\sigma'), -c, c)$;
  $y_i = r + \gamma Q_{\theta_i'}(s', a')$.
  update critics:

$$\theta_i \leftarrow \frac{1}{N}\sum \left(\min_{i=1,2} y_i - Q_{\theta_i}(s,a)\right)^2;$$

  Advantage loss:
  sample mini-batch of N transitions from B_h.
  update critics:

$$\theta_i \leftarrow \frac{1}{N}\sum [Q_i(s,a_h) - Q_i(s,\pi(s))];$$

  if $t\%d = 0$ **then**
    update ϕ by the deterministic policy gradient:

$$\nabla_\phi J(\phi) = \frac{\sum \nabla_a Q(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)}{N};$$

    update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$;
    $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$;
  end if
end for

In Algorithm 1, we use double experience buffer pools, where the expert's data is stored in B_h, and the agent's interaction data with the environment is stored in B. The double experience buffer pool not only helps to further distinguish expert and agent data, but also allows expert data to be more utilized.

V. Experiments

In this section, to verify the convergence speed and algorithm effect of the QDP algorithm, we conduct comparative experiments with the developed QDP-HRL, the probabilistic HRL (PHRL), and the original TD3 algorithms.

TABLE I
NETWORK STRUCTURE AND PARAMETERS

Network	Structure	Parameter
Critic network	Input layer	(state+action)×256
	Activation function	Relu
	Fully connected layer	256×256
	Activation function	Relu
	Output layer	256×1
Actor network	Input layer	state×256
	Activation function	Relu
	Fully connected layer	256×256
	Activation function	Relu
	Output layer	256×1
	Activation function	Tanh
Other	Batch size	256
	Delay step	2
	Learning rate	$1\times e^{-4}$
	τ	$5\times e^{-3}$
	γ	0.99
	t_h	2
	n	15

For the PHRL [41], the agent asks the human's advice, and the expert decides whether to give advice based on predetermined probability distribution. To make the experiment more sufficient, we also consider the case that the QDP-HRL is under the guidance of a suboptimal expert, denoted as QDP-subHRL, that is, the QDP-HRL with a suboptimal expert. Since the experimental environment of the Pendulum is relatively simple, it is just a simple experiment for illustration, the QDP-subHRL is validated on the lunar lander continuous, Bipedal Walker Hardcore, and Walker2d. In addition, we also plot the performance of the expert strategy and the suboptimal expert strategy. To comprehensively compare the above four algorithms, use the same parameters setting.

A. Experimental Setup

The machine used for experimental simulation is equipped with an Intel Xeon Silver 4210R CPU, 128-GB memory, and an Nvidia Tesla V100S GPU with 32-GB memory. The QDP-HRL, PHRL, and TD3 algorithms are implemented with PyTorch in Python. For comparison fairness, the setting for the network structure and parameters is the same, which is given in Table I, and remains unchanged for the four environments. To encourage the agent to explore the environment, a Gaussian noise $N(0,\sigma^2)$ is added to the action a output by the policy network $\pi(s)$, where σ is initially set to 0.4 and updated with $\sigma = \sigma \times 0.998$ every 2000 steps. To smooth the critic network during training, the Gaussian noise $N(0,\sigma'^2)$ is added to the action a' output by the target actor network, where σ' is set to 0.2. Each episode begins with an agent in a random state, and the turn ends when the agent reaches the maximum number of steps or dies to avoid the agent falling into a local optimum. To facilitate experiments, we pretrain a policy as an expert policy.

B. Pendulum

The inverted pendulum is a classic problem in control. In this environment, the agent starts at a random angle between $-\pi$ and π with a random speed, and the goal is to swing it up, so it stays upright. The action space has only one dimension, that is, the control torque of the motor in the interval $[-2, 2]$. The state dimension is 3-D. There is no terminal state, and the maximum number of steps of the agent per round is 200. We conducted experiments on different random seeds. The results are shown in Fig. 4, where the X-axis represents the training episodes, and the Y-axis represents the cumulative reward obtained by the agent. Human experts will only give rewards when the cumulative reward of the agent is lower than -800 to avoid excessive interference of human experts in the learning of the agent. It can be seen from Fig. 4 that the convergence speed of QDP-HRL is faster than that of TD3 and PHRL, which proves the effectiveness of our algorithm.

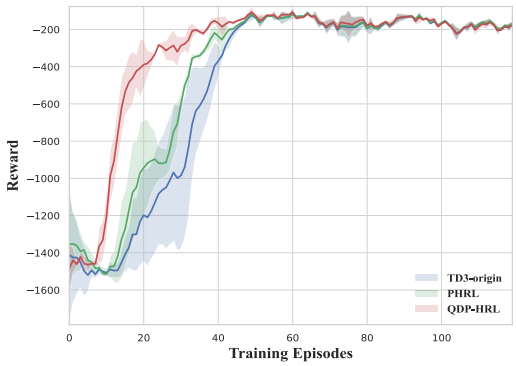

Fig. 4. Cumulative reward change curve on pendulum environment.

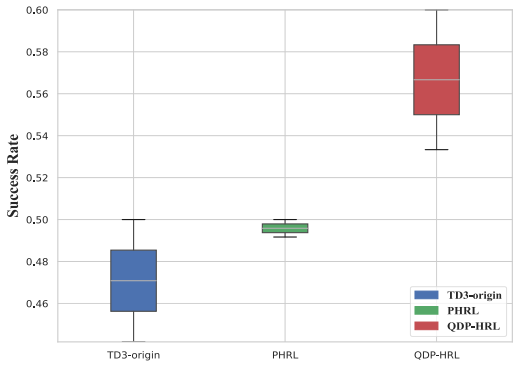

Fig. 5. Success rate curve on pendulum environment.

Fig. 5 shows the success rate in the training process, where an experiment is regarded as successful when the cumulative reward of an episode exceeds -200. We use box plots to record the success rate of each experiment with different random seeds. The center line in the square shows the median success rate of the agent under different random seeds, and the bottom and top of the square represent the 25th and 75th scores, respectively. It is observed from Fig. 5 that the success rate of QDP-HRL is the highest. From Fig. 6, it is shown that the number of human suggestions of the QDP-HRL algorithm is lower than PHRL, which means that the QDP-HRL algorithm is more effective in utilizing the human experience. Since the original TD3 has no human participation in training, the number of suggestions from human experts is 0.

C. Lunar Lander Continuous

The action space in lunarlandercontinuous-v2 is continuous, where the action vector contains four real values in the

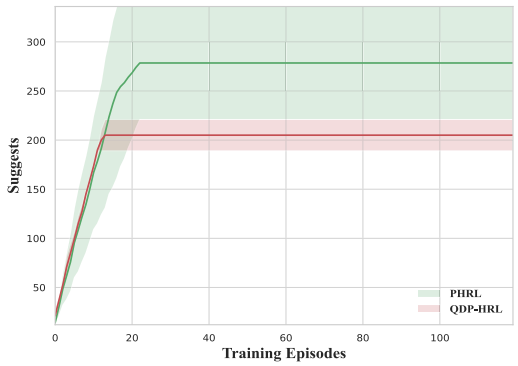

Fig. 6. Suggestions amount change curve on pendulum environment.

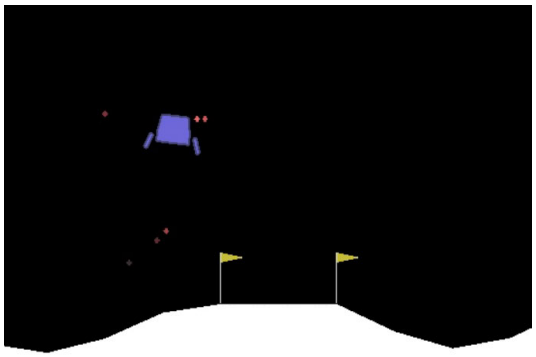

Fig. 7. Lunar lander continuous environment.

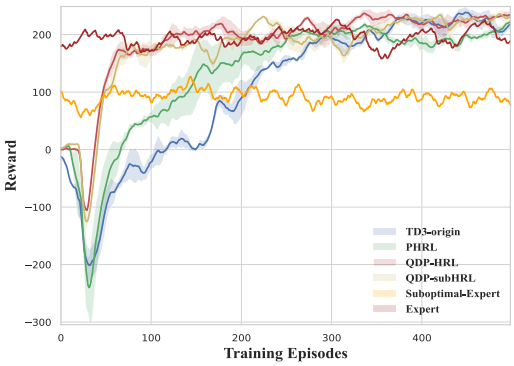

Fig. 8. Cumulative reward change curve on lunar lander environment.

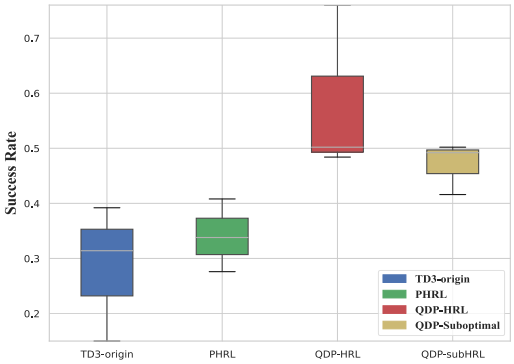

Fig. 9. Success rate curve on lunar lander environment.

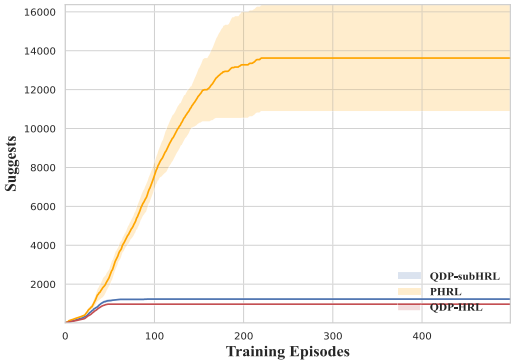

Fig. 10. Suggestions amount change curve on lunar lander environment.

interval $[-1, 1]$, that is, main, left, and right engine power. The state space is a continuous 8-D vector. The goal is to control the three engines (main/left/right) so that it lands in a designated area, for example, between two flags as shown in Fig. 7. The experimental results are demonstrated in Figs. 8–10. As shown in Fig. 8, the QDP-HRL algorithm achieves fast convergence of about 120 episodes, while PHRL and TD3 algorithms achieve about 250 episodes. Even under the guidance of suboptimal experts, QDP-subHRL can achieve a similar performance to QDP-HRL. For the lunar lander task, the task is regarded as successful when the cumulative reward of the agent in an experiment exceeds 200. In Fig. 9, it is demonstrated that the QDP-HRL algorithm achieves the highest success rate among the three algorithms. Agents can also achieve high success rates under the guidance of suboptimal experts. From Fig. 10, the human expert provides suggestions at the first about 50 and 200 training episodes for the QDP-HRL and PHRL algorithms, and the suboptimal expert only provides suggestions for the agent in QDP-subHRL at the first 50 training episodes, which means that the QDP-HRL algorithms require fewer manual monitoring sets than the PHRL algorithm.

D. Bipedal Walker Hardcore

The Bipedal Walker Hardcore environment is shown in Fig. 11. The agent interacts with the environment and learns to run and walk under different road conditions. Since this environment has obstacles such as ladders, tree stumps, and traps, the agent needs to learn skills such as running, crossing pits, crossing obstacles, and descending stairs. The agent's state space is 24-D, including the agent's angular velocity, horizontal velocity, vertical velocity, joint position, joint angular velocity, the contact between the legs and the ground, and ten lidar rangefinder measurements. There is no coordinate position in the state vector. The action space of the agent is a 4-D continuous action space. The original environment rewards $r = -100$ when it falls, and the reward under normal circumstances is probably in the interval $[-1, 1]$. Note that the reward difference between falls and normal circumstance is large, leading to a mutation in the Q-function that is not conducive to learning. Thus, we set reward $r = -1$ when falling occurs. As shown in Fig. 12, the convergence of the QDP-HRL algorithm is better than the original TD3 and PHRL. Under the guidance of the suboptimal expert, QDP-subHRL can also achieve similar effects to QDP-HRL, which shows that our proposed algorithm has good robustness.

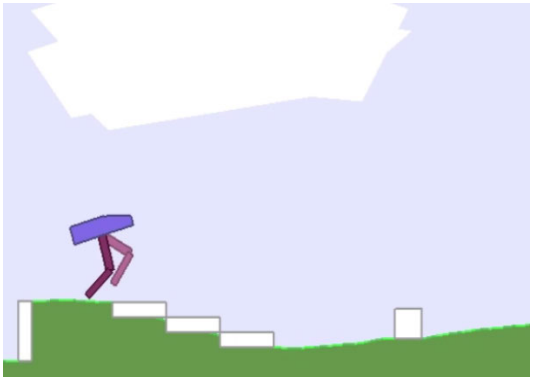

Fig. 11. Bipedal Walker Hardcore environment.

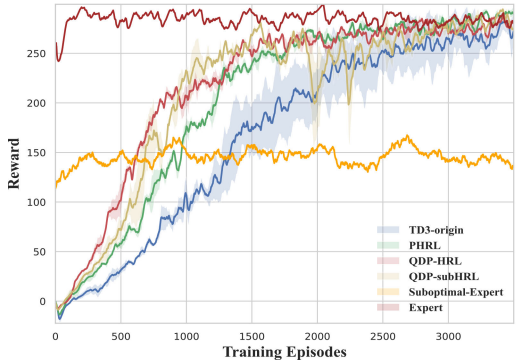

Fig. 12. Cumulative reward change curve on Bipedal Walker environment.

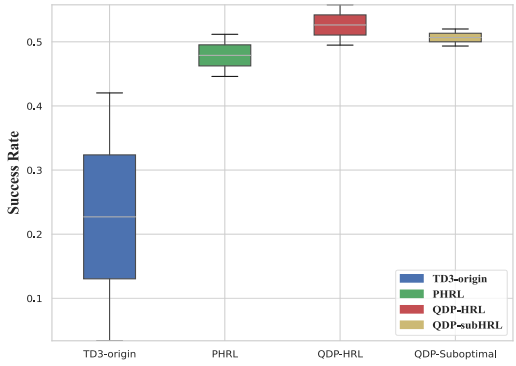

Fig. 13. Success rate curve on Bipedal Walker environment.

As shown in Figs. 13 and 14, the success rate of the QDP-HRL algorithm is the best among the three algorithms, and the number of human experts' suggestions is less than half that of the PHRL, indicating that the QDP-HRL algorithm achieves good results with fewer human suggestions.

E. Walker2d

The Walker2d environment is shown in Fig. 15. The agent's goal is to make a 2-D bipedal robot move forward as fast as possible without falling. The state space is a 17-D vector, and the action space is a 6-D vector. The reward curve is shown in Fig. 16. It is observed that the QDP-HRL algorithm not only achieves the fastest convergence speed in this task, but also outperforms TD3 and PHRL in performance. The guidance of suboptimal experts can also help the agent to achieve good performance. The QDP-HRL and QDP-subHRL algorithm achieves a good convergence after training to 1000 episodes,

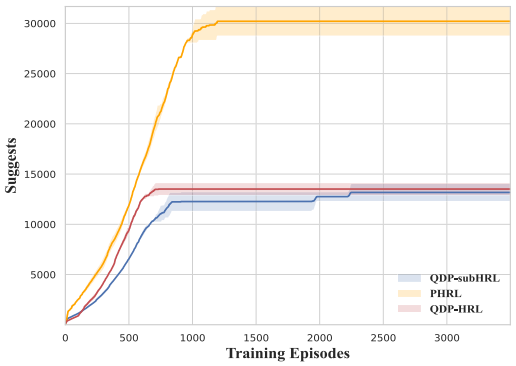

Fig. 14. Suggestions amount change curve on Bipedal Walker environment.

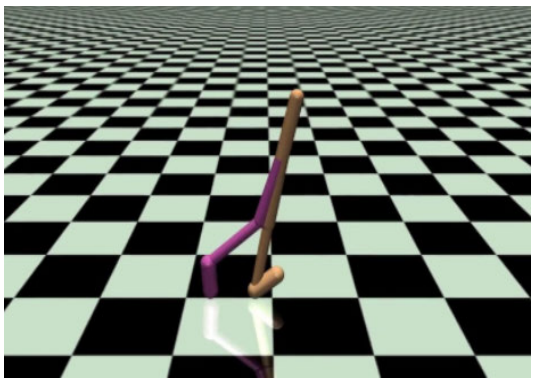

Fig. 15. Walker2d environment.

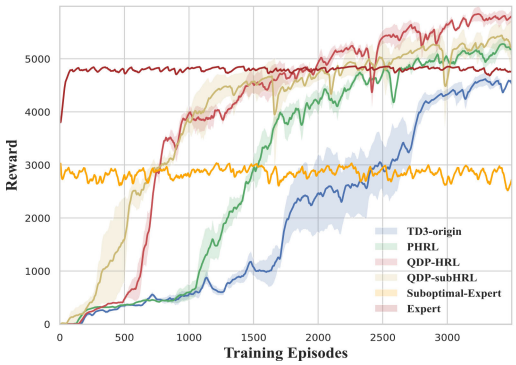

Fig. 16. Cumulative reward change curve on Walker2d environment.

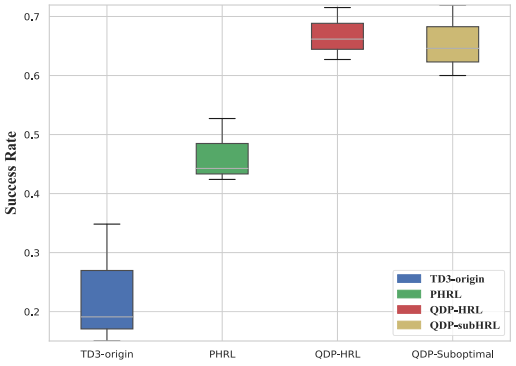

Fig. 17. Success rate curve on Walker2d environment.

while the PHRL and TD3 algorithms require training for 2000 and 2800 episodes, respectively, to achieve a similar effect. In this task, there is no terminal state. Thus, an experiment is regarded as successful if the obtained cumulative reward

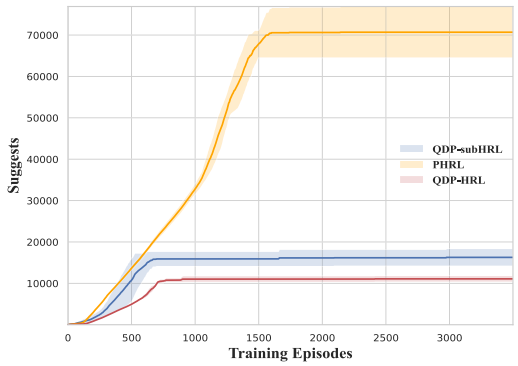

Fig. 18. Suggestions amount change curve on Walker2d environment.

exceeds 4000. As shown in Fig. 17, the QDP-HRL has the highest success rate during training. QDP-subHRL also has a similar success rate to QDP-HRL. For training, 3500 episodes are employed for three algorithms. From Fig. 18, it is shown that the human participates in training in the first 700 and 1500 episodes for the QDP-HRL and PHRL algorithms, and the suboptimal-expert participates in training in the first 800 for the QDP-subHRL, respectively. It means less human monitoring and less human advice are required in the QDP-HRL algorithm.

VI. Conclusion

In this article, we developed a QDP-HRL algorithm for continuous action space. By proposing a QDP strategy, the agent actively asks human experts for suggestions based on the difference in the Q value in the early training stage. The experts' suggestions are stored in the experience buffer pool. We define an advantage loss function using human expert experience and agent policy, which is used to guide the update of the critic network and further facilitate the improvement of the agent's policy. To verify the developed QDP-HRL algorithm, comparative simulation studies with the PHRL and TD3 algorithms are conducted in four environments. The results demonstrate that the QDP-HRL algorithm achieves a much higher success rate and faster convergence speed with fewer human suggestions.

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning—An introduction," in *Adaptive Computation and Machine Learning*. Cambridge, MA, USA: MIT Press, 1998.

[2] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[3] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[4] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[5] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artif. Intell.*, vol. 299, pp. 35–45, Oct. 2021.

[6] T. Zahavy, B. O'Donoghue, G. Desjardins, and S. Singh, "Reward is enough for convex MDPs," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 2546–2559.

[7] 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. (ICML)*, 2018, pp. 1861–1870.

[8] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022, doi: 10.1109/TNNLS.2021.3055499.

[9] X. Wang, Y. Gu, Y. Cheng, A. Liu, and C. L. P. Chen, "Approximate policy-based accelerated deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1820–1830, Jun. 2020.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[11] N. Dilokthanakul, C. Kaplanis, N. Pawlowski, and M. Shanahan, "Feature control as intrinsic motivation for hierarchical reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3409–3418, Nov. 2019.

[12] S. Pateria, B. Subagdja, A. Tan, and C. Quek, "End-to-end hierarchical reinforcement learning with integrated subgoal discovery," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7778–7790, Dec. 2022, doi: 10.1109/TNNLS.2021.3087733.

[13] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Jun. 2015.

[14] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[15] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone, "Leveraging human guidance for deep reinforcement learning tasks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1026–1034.

[16] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan, "Feature expansive reward learning: Rethinking human input," in *Proc. 16th ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, Mar. 2021, pp. 216–224.

[17] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4299–4307.

[18] G. Li, R. Gomez, K. Nakamura, and B. He, "Human-centered reinforcement learning: A survey," *IEEE Trans. Hum.-Mach. Syst.*, vol. 49, no. 4, pp. 337–349, Aug. 2019.

[19] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "Learning a low-dimensional representation of a safe region for safe reinforcement learning on dynamical systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2513–2527, May 2023, doi: 10.1109/TNNLS.2021.3106818.

[20] G. Fragkos, J. Johnson, and E. E. Tsiropoulou, "Dynamic role-based access control policy for smart grid applications: An offline deep reinforcement learning approach," *IEEE Trans. Human-Machine Syst.*, vol. 52, no. 4, pp. 761–773, Aug. 2022, doi: 10.1109/THMS.2022.3163185.

[21] J. Karalus and F. Lindner, "Accelerating the learning of TAMER with counterfactual explanations," 2021, *arXiv:2108.01358*.

[22] R. Pérez-Dattari, C. Celemin, J. Ruiz-Del Solar, and J. Kober, "Interactive learning with corrective feedback for policies based on deep neural networks," in *Proc. Int. Symp. Exp. Robot. (ISER)*, 2018, pp. 353–363.

[23] C. Celemin and J. Ruiz-Del-Solar, "An interactive framework for learning continuous actions policies based on corrective feedback," *J. Intell. Robotic Syst.*, vol. 95, no. 1, pp. 77–97, Jul. 2019.

[24] S. Ross et al., "Learning monocular reactive UAV control in cluttered natural environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 1765–1772.

[25] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using Gaussian mixture models," in *Proc. 6th Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2007, pp. 1–8.

[26] J. Broekens, "Emotion and reinforcement: Affective facial expressions facilitate robot learning," in *Artifical Intelligence for Human Computing*. Cham, Switzerland: Springer, 2007, pp. 113–132.

[27] W. Bradley Knox and P. Stone, "TAMER: Training an agent manually via evaluative reinforcement," in *Proc. 7th IEEE Int. Conf. Develop. Learn. (ICDL)*, Aug. 2008, pp. 292–297.

[28] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *Proc. IEEE Int. Conf. Rehabil. Robot. (ICRR)*, Jun. 2011, pp. 1–7.

[29] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2000, pp. 663–670.

[30] K. M. Jagodnik, P. S. Thomas, A. J. van den Bogert, M. S. Branicky, and R. F. Kirsch, "Human-like rewards to train a reinforcement learning controller for planar arm movement," *IEEE Trans. Hum.-Mach. Syst.*, vol. 46, no. 5, pp. 723–733, Oct. 2016.

[31] K. Lee, L. M. Smith, and P. Abbeel, "PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 139, 2021, pp. 6152–6163.
[32] M. Matarese, A. Sciutti, F. Rea, and S. Rossi, "Toward Robots' behavioral transparency of temporal difference reinforcement learning with a human teacher," *IEEE Trans. Hum.-Mach. Syst.*, vol. 51, no. 6, pp. 578–589, Dec. 2021.
[33] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The TAMER framework," in *Proc. 5th Int. Conf. Knowl. Capture*, Sep. 2009, pp. 9–16.
[34] W. B. Knox, B. D. Glass, B. C. Love, W. T. Maddox, and P. Stone, "How humans teach agents," *Int. J. Social Robot.*, vol. 4, no. 4, pp. 409–421, Nov. 2012.
[35] R. Arakawa, S. Kobayashi, Y. Unno, Y. Tsuboi, and S.-I. Maeda, "DQN-TAMER: Human-in-the-loop reinforcement learning with intractable feedback," 2018, *arXiv:1810.11748*.
[36] G. Warnell, N. R. Waytowich, V. Lawhern, and P. Stone, "Deep TAMER: Interactive agent shaping in high-dimensional state spaces," in *Proc. Assoc. Advancement Artif. Intell. (AAAI)*, 2018, pp. 1545–1554.
[37] M. Palan, G. Shevchuk, N. C. Landolfi, and D. Sadigh, "Learning reward functions by integrating human demonstrations and preferences," 2019, *arXiv:1906.08928*.
[38] Z. Cao, K. Wong, and C. Lin, "Weak human preference supervision for deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5369–5378, Dec. 2021.
[39] F. Behbahani et al., "Learning from demonstration in the wild," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 775–781.
[40] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6DoF closed-loop grasping from low-cost demonstrations," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4978–4985, Jul. 2020.
[41] L. Torrey and M. E. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proc. Int. Conf. Auto. Agents Multi-Agent Syst. (AAMAS)*, 2013, pp. 1053–1060.
[42] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proc. Assoc. Advancement Artif. Intell. (AAAI)*, 2020, pp. 5792–5799.
[43] S. Thakur, H. van Hoof, J. C. G. Higuera, D. Precup, and D. Meger, "Uncertainty aware learning from demonstrations in multiple contexts using Bayesian neural networks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 768–774.
[44] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2013, pp. 2625–2633.
[45] O. Amir, E. Kamar, A. Kolobov, and B. J. Grosz, "Interactive teaching strategies for agent training," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2016, pp. 804–811.
[46] W. B. Knox and P. Stone, "Learning non-myopically from human-generated reward," in *Proc. Int. Conf. Intell. User Inter. (IUI)*, Mar. 2013, pp. 191–202.
[47] U. Syed, M. Bowling, and R. E. Schapire, "Apprenticeship learning using linear programming," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1032–1039.
[48] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.
[49] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 920–928.
[50] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," 2017, *arXiv:1710.11248*.
[51] S. K. S. Ghasemipour, R. Zemel, and S. Gu, "A divergence minimization perspective on imitation learning methods," in *Proc. Conf. Robot Learn.*, 2020, pp. 1259–1277.
[52] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn., (ICML)*, 2018, pp. 1582–1591.
[53] M. L. Puterman, "Markov decision processes," in *Handbooks in Operations Research and Management Science*, vol. 2. Berkeley, CA, USA: Elsevier, 1990, pp. 331–434.
[54] P. Ren et al., "A survey of deep active learning," *ACM Comput. Surveys*, vol. 54, no. 9, pp. 1–20, 2021.
[55] X. Zhan, H. Liu, Q. Li, and A. B. Chan, "A comparative survey: Benchmarking for pool-based active learning," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 4679–4686.
[56] P. Kumar and A. Gupta, "Active learning query strategies for classification, regression, and clustering: A survey," *J. Comput. Sci. Technol.*, vol. 35, no. 4, pp. 913–945, Jul. 2020.

Biao Luo (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Beihang University, Beijing, China, in 2014.

He is currently a Professor with the School of Automation, Central South University (CSU), Changsha, China. Before joining CSU, he was an Associate Professor and Assistant Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, from 2014 to 2018. His current research interests include intelligent control, reinforcement learning, deep learning, and decision-making.

Dr. Luo serves as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the Artificial Intelligence Review, the *Neurocomputing*, and the *Journal of Industrial and Management Optimization*. He is the Vice-Chair of the Adaptive Dynamic Programming and Reinforcement Learning Technical Committee and the Chinese Association of Automation.

Zhengke Wu received the B.E. degree in measuring and control technology and instrumentations from Henan University, Kaifeng, China, in 2020. He is currently pursuing the M.E. degree in artificial intelligence with the School of Automation, Central South University (CSU), Changsha, China.

His current research interests include human-in-the-loop, imitation learning, and deep reinforcement learning.

Fei Zhou received the B.E. degree from the Department of Mechanical Engineering and Automation, Shantou University, Shantou, China, in 2018. He is currently pursuing the master's degree with the Department of Automation, Central South University, Changsha, China.

His research interests include reinforcement learning and recommender systems.

Bing-Chuan Wang received the B.E. degree in automation and the M.S. degree in control science and engineering from the Central South University, Changsha, China, in 2013 and 2016, respectively, and the Ph.D. degree in system engineering and engineering management from the City University of Hong Kong, Hong Kong, in 2019.

He is currently with the School of Automation, Central South University. His current research interests include computational intelligence and physics-informed machine learning.