Relative Entropy Regularized Sample-Efficient Reinforcement Learning With Continuous Actions

Zhiwei Shang, Renxing Li, Chunhua Zheng, *Member, IEEE*, Huiyun Li, *Senior Member, IEEE*, and Yunduan Cui, *Member, IEEE*

***Abstract*— In this article, a novel reinforcement learning (RL) approach, continuous dynamic policy programming (CDPP), is proposed to tackle the issues of both learning stability and sample efficiency in the current RL methods with continuous actions. The proposed method naturally extends the relative entropy regularization from the value function-based framework to the actor–critic (AC) framework of deep deterministic policy gradient (DDPG) to stabilize the learning process in continuous action space. It tackles the intractable softmax operation over continuous actions in the critic by Monte Carlo estimation and explores the practical advantages of the Mellowmax operator. A Boltzmann sampling policy is proposed to guide the exploration of actor following the relative entropy regularized critic for superior learning capability, exploration efficiency, and robustness. Evaluated by several benchmark and real-robot-based simulation tasks, the proposed method illustrates the positive impact of the relative entropy regularization including efficient exploration behavior and stable policy update in RL with continuous action space and successfully outperforms the related baseline approaches in both sample efficiency and learning stability.**

***Index Terms*— Reinforcement learning (RL), robot learning.**

I. Introduction

IN RECENT years, deep reinforcement learning (DRL), the combination of deep learning [1] and reinforcement learning (RL) [2], has led to an appealing solution of artificial intelligence that iteratively learns optimal or suboptimal control policies of complex tasks from high-dimensional unprocessed sensory information. Besides the successful implementations that outperform humans in a wide range of tasks based on programming code or human-made rule [3], [4] using discrete actions, many DRL approaches, such as deep deterministic policy gradient (DDPG) [5], soft actor–critic (SAC) [6], and proximal policy optimization (PPO) [7], are extended to the continuous action space to properly learn continuous control policy in a wide range of engineering applications [8], [9], [10], [11], [12], [13], [14], [15]. On the other hand, the practical DRL approach still remains a long-term ambition as a result of the black-box property and the data-driven nature of its neural network structure, which result in both the instability during the learning process and the massive samples required for exploration [16], [17]. These issues have become more serious in the continuous action-based approaches: their actor–critic (AC) framework requires more samples to separately train the networks of value function and policy, while the mismatch between two networks turns to an extremely unstable learning process due to not only the huge overestimation of value function and policy but also the strong sensitivity to hyperparameters [18]. To address these issues, advantage actor–critic (A2C) and asynchronous advantage actor–critic (A3C) [19], [20] introduce the advantage function as a baseline to alleviate the unstable learning caused by the overestimation of the value function in the AC framework that focuses on value function approximation. The distributional value function was also integrated with SAC to reduce the overestimation of Q function [21]. The exploration efficiency was addressed in [22] by introducing weakly pessimistic value estimation and optimistic policy optimization. For the policy-based approaches, additional constraints are employed in the policy gradient to avoid overlarge policy updates and achieve more stable learning procedures [23], [24]. The off-policy version of the existing approach was proposed with a theoretical guarantee of monotonic improvement for superior sample efficiency [25].

As one potential solution to tackle the issues above in a natural way, the dynamic policy programming (DPP) [26] first introduces the relative entropy, i.e., Kullback–Leibler (KL) divergence between the current and previous policies into the value function-based RL as a regularization term to avoid overlarge policy update and smooth the learning process. Theoretical research has demonstrated that the relative entropy term implicitly averages the error of the approximated value function with state-of-the-art (SOTA) error dependency [27], [28]. In terms of applications, this

Manuscript received 24 June 2022; revised 22 June 2023 and 24 September 2023; accepted 29 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62103403, in part by the Shenzhen Basic Key Research Project under Grant JCYJ20200109115414354, in part by the Shenzhen Science and Technology Funding under Grant KCXST20221021111210023, in part by the Shenzhen Science and Technology Innovation Commission under Grant JCYJ20210324115800002, and in part by the International Partnership Program of the Chinese Academy of Sciences under Grant 321GJHZ2022057MI. *(Corresponding author: Yunduan Cui.)*

Zhiwei Shang is with the Systems Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong 518055, China.

Renxing Li is with the School of Software Engineering, University of Science and Technology of China, Hefei, Anhui 230026, China.

Chunhua Zheng, Huiyun Li, and Yunduan Cui are with the CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems and the Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China (e-mail: cuiyunduan@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3329513.

Digital Object Identifier 10.1109/TNNLS.2023.3329513

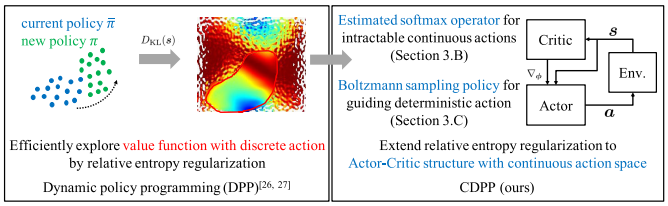


Fig. 1. Principle of CDPP proposed in this work.

characteristic also contributes to superior sample efficiency and learning capability in a wide range of engineering tasks from robot control [29], [30] to chemical platform optimization [31], [32] where the agents efficiently explore the target tasks with a limited number of interactions using smoothly updated policies. However, despite the promising results in practice, the current works are mainly limited to tasks with discrete actions, while directly extending the relative entropy regularization to the DDPG-like RL approaches with continuous action space is tricky: the relative entropy regularization in DPP requires traversal softmax operations over the entire discrete action space, which is intractable to the continuous actions under the AC structure of DDPG. One related work improved the robustness of the AC structure-based RL by restricting the actor using the relative entropy term [33], while a general approach that jointly constrains both the actor and critic remains an open issue.

This article aims to tackle the issues of both learning stability and sample efficiency in the learning of complex tasks with continuous action space. Following the principle described in Fig. 1, a novel RL approach, continuous DPP (CDPP), is proposed to naturally introduce the relative entropy regularization from the existing value function-based approach DPP [26] with discrete action space to the AC structure of DDPG [5] with continuous action space. For the critic, CDPP tackles the intractable softmax operation through Monte Carlo estimation and explores the advantages of the Mellowmax operator instead of the softmax operator for a better convergence performance toward the unique fixed point according to the theoretical study [34]. For the actor, CDPP proposes a Boltzmann sampling policy to guide the deterministic action to explore following the relative entropy regularized value function. CDPP migrates the Boltzmann sampling policy, which contributes to high exploration efficiency in DPP, from a discrete action space to a continuous one. This naturally brings about superior learning capability, exploration efficiency, and robustness over the original deterministic policy with additional exploration noise in the traditional AC framework, as reported by Cesa-Bianchi et al. [35] and Wang et al. [36]. Evaluated by several benchmark and real-robot-based simulation tasks with different complexities, the proposed CDPP naturally integrated the sample efficiency and learning capability of relative entropy regularization to the AC structure with continuous action space and successfully demonstrated a superior learning performance with not only more smooth exploration behavior but also less overestimation of the value function compared with other baseline approaches. According to the relationship of related works concluded in Table I, the contributions of this article are summarized as follows.

TABLE I
COMPARISON OF CDPP AND THE RELATED APPROACHES

Approach	Relative entropy	Continuous action	Mellowmax
DQN [3]	×	×	×
DPP [26], [27]	◯	×	×
DDPG [5]	×	◯	×
SAC [6]	×	◯	×
PPO [7]	×	◯	×
CDPP (ours)	◯	◯	◯

1) For the theoretic study, this work novelly extended the relative entropy regularized RL from the value function-based approach with discrete action space to the AC structure with continuous action space, which expands the potential of relative entropy regularization as an emerging direction of practical RL.
2) For the algorithm development, the proposed CDPP successfully integrated both sample efficiency and stability of DPP and the effectiveness of the Mellowmax operator with theoretically better convergence performance toward the unique fixed point within one algorithm, which explores the advantages of Mellowmax in the relative entropy regularized RL.
3) For the experimental validations, the proposed approach was evaluated in various benchmark tasks with different complexities compared with the related baseline approaches. The experimental results clearly illustrated the advantages of CDPP, including not only better learning stability and sample efficiency but also less overestimation of the value function.

The remainder of this article is organized as follows. The preliminaries are presented in Section II. The proposed CDPP is detailed in Section III. The experimental results are demonstrated in Section IV. The conclusions are given in Section V.

II. PRELIMINARIES

A. Markov Decision Process

The environment is modeled as Markov decision processes in RL with a five-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, where $\mathcal{S}$ denotes the finite set of states, $\mathcal{A}$ is a set of actions called the action space, $\mathcal{R}^{\boldsymbol{a}}_{ss'}$ is the immediate reward received after transitioning from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ under action $\boldsymbol{a}$, $\mathcal{P}$ represents the corresponding state transition probability matrix following $\mathcal{P}^{\boldsymbol{a}}_{ss'} = p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$, and $\gamma \in [0, 1]$ is the discount factor. A policy function $\pi(\boldsymbol{a}|\boldsymbol{s})$ maps states to a probability distribution that represents the probability of selecting each action. The value function is defined as the expected discounted total reward in state $\boldsymbol{s}$

$$V_\pi(\boldsymbol{s}) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_{\boldsymbol{s}_t} | \boldsymbol{s}_0 = \boldsymbol{s}\right] \tag{1}$$

where $r_{\boldsymbol{s}_t} = \sum_{\boldsymbol{s}_{t+1} \in \mathcal{S}}^{\boldsymbol{a} \in \mathcal{A}} \pi(\boldsymbol{a}|\boldsymbol{s}_t) \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}_t \boldsymbol{s}_{t+1}} \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}_t \boldsymbol{s}_{t+1}}$ is the expected reward from state $\boldsymbol{s}_t$. The objective of RL is to find optimal policy π^* that maximizes the value function following

Bellman equation [2]:

$$V^*(\boldsymbol{s}) = \max_{\pi} \sum_{\substack{\boldsymbol{a}\in\mathcal{A}\\ \boldsymbol{s}'\in\mathcal{S}}} \pi(\boldsymbol{a}|\boldsymbol{s})\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma V^*(\boldsymbol{s}')\right). \tag{2}$$

It is also equivalent to represent such an optimal Bellman equation through the value function of state–action pairs $(\boldsymbol{s}, \boldsymbol{a})$

$$Q^*(\boldsymbol{s},\boldsymbol{a}) = \max_{\pi} \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma \sum_{\boldsymbol{a}'\in\mathcal{A}} \pi(\boldsymbol{a}'|\boldsymbol{s}')\, Q^*(\boldsymbol{s}',\boldsymbol{a}')\right). \tag{3}$$

B. Deep Deterministic Policy Gradient

As a traditional RL approach for continuous action space, DDPG [5] employs the AC structure to separately train the approximated Q-function network (the critic) $\hat{Q}$ for evaluating the state–action pairs and the policy network $\hat{\mu}$ (the actor) for decision making with independent parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. During step t in the sample collection stage, the exploration behavior of DDPG is achieved by an additional noise following Gaussian distribution or Ornstein–Uhlenbeck process $\boldsymbol{a}_t = \hat{\mu}(\boldsymbol{s}_t, \boldsymbol{\phi}) + \boldsymbol{\omega}$. The agent executes $\boldsymbol{a}_t$ to the environment, moves to the next state $\boldsymbol{s}_{t+1}$, and receives an immediate reward $\mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}$. The transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, \mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t\boldsymbol{s}_{t+1}}, \boldsymbol{s}_{t+1})$ will be stored in memory replay buffer $\mathcal{D}$.

The training process mainly follows deep Q-learning (DQL) [3] with memory replay and target networks. The parameter of critic network $\boldsymbol{\theta}$ is optimized by minimizing the following loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{J}\sum_{j=1}^{J}\left(y_j - \hat{Q}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta})\right)^2 \tag{4}$$

$$y_j = \mathcal{R}^{\boldsymbol{a}_j}_{\boldsymbol{s}_j\boldsymbol{s}_{j+1}} + \gamma \hat{Q}_{\text{target}}\left(\boldsymbol{s}_{j+1}, \hat{\mu}_{\text{target}}(\boldsymbol{s}_{j+1}, \boldsymbol{\phi}^-), \boldsymbol{\theta}^-\right) \tag{5}$$

where J is the samples randomly selected from the experience replay buffer, and $\hat{Q}_{\text{target}}$ and $\hat{\mu}_{\text{target}}$ are the target networks of critic and actor with parameters $\boldsymbol{\theta}^-$ and $\boldsymbol{\phi}^-$. The actor is updated by sampled policy gradient [2], [5]

$$\nabla_{\boldsymbol{\phi}} \leftarrow \frac{1}{J}\sum_{j=1}^{J} \nabla_{\boldsymbol{a}'_j} \hat{Q}\left(\boldsymbol{s}_j, \boldsymbol{a}'_j, \boldsymbol{\theta}\right)\big|_{\boldsymbol{a}'_j=\hat{\mu}(\boldsymbol{s}_j,\boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}}\mu(\boldsymbol{s}_j, \boldsymbol{\phi}). \tag{6}$$

Let $\tau \in (0, 1]$, and the target networks are smoothly updated as follows:

$$\begin{aligned} \boldsymbol{\theta}^- &\leftarrow \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}^- \\ \boldsymbol{\phi}^- &\leftarrow \tau\boldsymbol{\phi} + (1-\tau)\boldsymbol{\phi}^-. \end{aligned} \tag{7}$$

III. Approach

A. Relative Entropy Regularized AC Structure

In this section, we detail the proposed approach, CDPP, that extends the relative entropy regularization from value function-based method DPP [26] to the AC structure for continuous action space.

Defining the current and previous policies as $\pi(\boldsymbol{a}|\boldsymbol{s})$ and $\bar{\pi}(\boldsymbol{a}|\boldsymbol{s})$, the relative entropy between them is calculated as follows:

$$D_{\text{KL}}(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\mathcal{A}} \pi(\boldsymbol{a}|\boldsymbol{s}) \log\left(\frac{\pi(\boldsymbol{a}|\boldsymbol{s})}{\bar{\pi}(\boldsymbol{a}|\boldsymbol{s})}\right). \tag{8}$$

DPP attempts to introduce this relative entropy term into the value function as follows:

$$V^{\pi}_{\bar{\pi}}(\boldsymbol{s}) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(r_{\boldsymbol{s}_t} - \frac{1}{\eta}D_{\text{KL}}(\boldsymbol{s}_t)\right)|\boldsymbol{s}_0 = \boldsymbol{s}\right] \tag{9}$$

where the inverse temperature η controls the KL divergence penalty. The corresponding optimal value function, thus, satisfies a Bellman equation by combining (2) and (9)

$$V^*_{\bar{\pi}}(\boldsymbol{s}) = \max_{\pi} \sum_{\substack{\boldsymbol{a}\in\mathcal{A}\\ \boldsymbol{s}'\in\mathcal{S}}} \pi(\boldsymbol{a}|\boldsymbol{s})\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma V^*_{\bar{\pi}}(\boldsymbol{s}')\right) - \frac{1}{\eta}D_{\text{KL}}(\boldsymbol{s}). \tag{10}$$

The optimal value function $V^*_{\bar{\pi},t+1}(\boldsymbol{s})$ and policy $\bar{\pi}^*_{t+1}(\boldsymbol{a}|\boldsymbol{s})$ can be calculated by the following two equations, respectively, according to [37]:

$$\begin{aligned} V^{\pi}_{\bar{\pi},t+1}(\boldsymbol{s}) = \frac{1}{\eta}\log\sum_{\boldsymbol{a}\in\mathcal{A}} \bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s}) \\ \times \exp\left(\eta\sum_{\boldsymbol{s}'\in\mathcal{S}}\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma V^{\pi}_{\bar{\pi},t}(\boldsymbol{s}')\right)\right) \end{aligned} \tag{11}$$

$$\bar{\pi}_{t+1}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s})\exp\left(\eta\sum_{\boldsymbol{s}'\in\mathcal{S}}\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma V^{\pi}_{\bar{\pi},t}(\boldsymbol{s}')\right)\right)}{\exp\left(\eta V^{\pi}_{\bar{\pi},t+1}(\boldsymbol{s})\right)}. \tag{12}$$

These equations can be expressed by the action preferences function following [2], [26], which is close to the Q function:

$$\Psi_{t+1}(\boldsymbol{s},\boldsymbol{a}) = \frac{1}{\eta}\log\bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s}) + \sum_{\boldsymbol{s}'\in\mathcal{S}}\mathcal{P}^{a}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma V^{\pi}_{\bar{\pi},t}(\boldsymbol{s}')\right). \tag{13}$$

Combine (13) with (11) and (12), the following simple double equations at the t-th iteration are obtained:

$$V^{\pi}_{\bar{\pi},t+1}(\boldsymbol{s}) = \frac{1}{\eta}\log\sum_{\boldsymbol{a}\in\mathcal{A}}\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a})) \tag{14}$$

$$\bar{\pi}_{t+1}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a}))}{\sum_{\boldsymbol{a}'\in\mathcal{A}}\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a}'))}. \tag{15}$$

The corresponding update rule is derived by inserting (14) and (15) into (13)

$$\begin{aligned} \Psi_{t+1}(\boldsymbol{s},\boldsymbol{a}) = \Psi_t(\boldsymbol{s},\boldsymbol{a}) - \mathcal{L}_{\eta}\Psi_t(\boldsymbol{s}) \\ + \sum_{\boldsymbol{s}'\in\mathcal{S}}\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\left(\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'} + \gamma\mathcal{L}_{\eta}\Psi_t(\boldsymbol{s}')\right) \end{aligned} \tag{16}$$

$$\mathcal{L}_{\eta}\Psi_t(\boldsymbol{s}) = \frac{1}{\eta}\log\sum_{\boldsymbol{a}\in\mathcal{A}}\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a})). \tag{17}$$

In practice, $\mathcal{L}_{\eta}$ in (17) can be replaced by a Boltzmann softmax operator $\mathcal{B}_{\eta}$ in order to solve the interaction problem in high-dimensional state–action space [26], [29]

$$\mathcal{B}_{\eta}\Psi_t(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\mathcal{A}}\frac{\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a}))\Psi_t(\boldsymbol{s},\boldsymbol{a})}{\sum_{\boldsymbol{a}'\in\mathcal{A}}\exp(\eta\Psi_t(\boldsymbol{s},\boldsymbol{a}'))}. \tag{18}$$

To introduce the relative entropy regularization from DPP to the AC structure, a straightforward solution is to replace

the Q function in DPPG's critic with the action preferences function Ψ. Defining $\hat{\Psi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta})$ and $\hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-)$ as the main and target networks of the original Ψ, the corresponding temporal-difference (TD) errors are defined as follows:

$$\begin{aligned} L(\boldsymbol{\theta}^{\Psi}) &= \mathbb{E}\Big[\big(y(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-) - \hat{\Psi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta})\big)^2\Big] \\ y(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-) &= \hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-) + \mathcal{R}^{\boldsymbol{a}}_{ss'} - \mathcal{B}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{\theta}^-) \\ &\quad + \gamma \mathcal{B}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s}',\boldsymbol{\theta}^-). \end{aligned} \tag{19}$$

However, the calculation of operator $\mathcal{B}_\eta$ in (19) is intractable in continuous action space due to the integration over all candidate actions in (18).

B. Critic With Estimated Softmax Operation

To tackle this issue, CDPP approximately estimates the Boltzmann softmax operation by Monte Carlo sampling following multiple dimensional Gaussian distributions in local action space. Given the input action $\boldsymbol{a}$ in (18), CDPP extends it to a set of $m+1$ Monte Carlo samples $\mathcal{A}_{\text{MC}} = [\boldsymbol{a}^0_{\text{MC}}, \boldsymbol{a}^1_{\text{MC}}, \ldots, \boldsymbol{a}^m_{\text{MC}}]$, where $\boldsymbol{a}^0_{\text{MC}} = \boldsymbol{a}$, and other elements are biased following a Gaussian distribution $\boldsymbol{a}^i_{\text{MC}} = \boldsymbol{a} + \boldsymbol{e}^i$, $\boldsymbol{e}^i \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\zeta})$ for $i = 1,\ldots,m$. The corresponding Boltzmann softmax operation $\mathcal{B}_\eta$ in (19) is calculated as follows:

$$\begin{aligned} &\mathcal{B}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{\theta}^-) \\ &\quad \approx \sum_{\boldsymbol{a}\in\mathcal{A}_{\text{MC}}} \frac{\exp\big(\eta \Psi_{\text{target}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-)\big)\Psi_{\text{target}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^-)}{\sum_{\boldsymbol{a}'\in\mathcal{A}_{\text{MC}}} \exp\big(\eta \Psi_{\text{target}}(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}^-)\big)}. \end{aligned} \tag{20}$$

Without the intractable integration over all continuous actions, CDPP statistically estimates the action preferences function near the given action $\boldsymbol{a}$, while the smoothness of the softmax operator is also controlled by η following DPP.

According to [18], the overestimation of the approximated value function results in an accumulated error during the learning process. Define the noise from networks and samples as ϵ, the approximated value function usually follows $\mathbb{E}[\max_{\boldsymbol{a}'} Q(\boldsymbol{s},\boldsymbol{a}') + \epsilon] \geq \max_{\boldsymbol{a}'} Q(\boldsymbol{s},\boldsymbol{a}')$. The learning performance of the original DDPG is easily damaged by this accumulated error due to its AC structure: the overestimated value function in critic turns to improper policy update, which, in turn, leads to a poor learning process of the value function. As a comparison, CDPP's critic enjoys a relieved overestimation of the value function in the AC structure because of the smooth mean-like property in the estimated softmax operation, i.e., $\mathbb{E}[\max_{\boldsymbol{a}'} Q(\boldsymbol{s},\boldsymbol{a}') + \epsilon] \geq \mathbb{E}[\text{softmax}_{\boldsymbol{a}'} Q(\boldsymbol{s},\boldsymbol{a}') + \epsilon]$.

As reported by Asadi and Littman [34], the Boltzmann softmax operator has multiple fixed points without nonexpansion property that theoretically guarantees the convergence of "Q-learning-like" algorithms to a unique fixed point. To tackle the issue of convergence performance, CDPP replaces the Boltzmann softmax operator in (18) with the Mellowmax operator with a unique fix point and nonexpansion property following [34]:

$$\mathcal{M}_\eta \Psi_t(\boldsymbol{s}) = \frac{1}{\eta} \log\left(\frac{1}{|\mathcal{A}|}\sum_{\boldsymbol{a}\in\mathcal{A}} \exp(\eta \Psi_t(\boldsymbol{s},\boldsymbol{a}))\right) \tag{21}$$

where $|\mathcal{A}|$ is the number of the discrete actions and $\eta \in (0,\infty)$ is the inverse temperature. The corresponding estimated Mellowmax operation for continuous action space is summarized in Algorithm 1.

Algorithm 1 Estimated Mellowmax Operation

Input: $\boldsymbol{s}, \boldsymbol{a}, \eta, \boldsymbol{\zeta}, m$
Function `CalMm(`$\boldsymbol{s},\boldsymbol{a}$`)`**:**
  # Build $m+1$ Monte Carlo samples set
  $\mathcal{A}_{\text{MC}} = [\boldsymbol{a}^0_{\text{MC}}, \boldsymbol{a}^1_{\text{MC}}, \ldots, \boldsymbol{a}^m_{\text{MC}}]$
  $\boldsymbol{a}^0_{\text{MC}} = \boldsymbol{a}$
  $\boldsymbol{a}^i_{\text{MC}} = \boldsymbol{a} + \boldsymbol{e}^i$, $\boldsymbol{e}^i \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\zeta})$ for $i = 1,\ldots,m$
  # Estimate Mellowmax operation based on $\mathcal{A}_{\text{MC}}$
  $\mathcal{M}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{\theta}^-) = \frac{1}{\eta}\log\left(\frac{\sum_{\boldsymbol{a}'\in\mathcal{A}_{\text{MC}}} \exp(\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}^-))}{|\mathcal{A}_{\text{MC}}|}\right)$
  Return $\mathcal{M}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{\theta}^-)$

Algorithm 2 Boltzmann Sampling Policy

Input: $\boldsymbol{s}$, $\boldsymbol{\zeta}$, $\mathcal{D}$
Function `CBSP(`$\mathcal{D}$`)`**:**
  # Generate deterministic action from actor
  $\boldsymbol{a} = \hat{\mu}_{\text{target}}(\boldsymbol{s},\boldsymbol{\phi}^-)$
  # Build $m+1$ Monte Carlo samples set based on $\boldsymbol{a}$
  $\mathcal{A}_{\text{MC}} = [\boldsymbol{a}^0_{\text{MC}}, \boldsymbol{a}^1_{\text{MC}}, \ldots, \boldsymbol{a}^m_{\text{MC}}]$
  $\boldsymbol{a}^0_{\text{MC}} = \boldsymbol{a}$
  $\boldsymbol{a}^i_{\text{MC}} = \boldsymbol{a} + \boldsymbol{e}^i$, $\boldsymbol{e}^i \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\zeta})$ for $i = 1,\ldots,m$
  # Calculate the probability of each action in $\mathcal{A}_{\text{MC}}$ and store in vector $\boldsymbol{p}$
  for $j = 0$ *to* m **do**
    $p(\boldsymbol{a}^j_{\text{MC}}) = \frac{\exp(\hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{a}^j_{\text{MC}},\boldsymbol{\theta}^-))}{\sum_{\boldsymbol{a}'\in\mathcal{A}_{\text{MC}}} \exp(\hat{\Psi}_{\text{target}}(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}^-))}$
    $\boldsymbol{p} \leftarrow p(\boldsymbol{a}_j)$
  # Sample one action following probability vector $\boldsymbol{p}$
  $\boldsymbol{a}_{\text{sampled}} \sim \boldsymbol{p}$
  Execute $\boldsymbol{a}_{\text{sampled}}$, obtain $\boldsymbol{s}'$ and $\mathcal{R}^{\boldsymbol{a}_{\text{sampled}}}_{ss'}$
  # Store the 4-tuple to the memory buffer $\mathcal{D}$
  $\mathcal{D} \leftarrow (\boldsymbol{s}, \boldsymbol{a}_{\text{sampled}}, \mathcal{R}^{\boldsymbol{a}_{\text{sampled}}}_{ss'}, \boldsymbol{s}')$

C. Actor With Boltzmann Sampling Policy

In the original DDPG, the exploration behavior of the actor is implemented by introducing an additional noise following Gaussian distribution or the Ornstein–Uhlenbeck process, which is independent of the current value function. As a comparison, the Boltzmann sampling policy following (15) where the probability of selecting one action is based on its action preferences function has proved to enjoy superior learning performances in practical applications of DPP with discrete actions [29], [30], [31].

In order to gap such a mismatch between the deterministic continuous actions with independent exploration noise in DDPG and the Boltzmann sampling policy of discrete actions in DPP, CDPP employs an estimated Boltzmann sampling policy in actor following the Monte Carlo sampling trick introduced in Section III-B. Given a deterministic action from the approximated actor network $\boldsymbol{a} = \hat{\mu}(\boldsymbol{s},\boldsymbol{\phi}^-)$, CDPP randomly

selects an exploration action $\boldsymbol{a}_{\text{sampled}}$ within the expanded set of $m+1$ Monte Carlo samples $\mathcal{A}_{\text{MC}}$ based on the following probability:

$$p(\boldsymbol{a}_{\text{sampled}}) = \frac{\exp\left(\hat{\Psi}_{\text{target}}\left(\boldsymbol{s}, \boldsymbol{a}_{\text{sampled}}, \boldsymbol{\theta}^{-}\right)\right)}{\sum_{\boldsymbol{a}' \in \mathcal{A}_{\text{MC}}} \exp\left(\hat{\Psi}_{\text{target}}\left(\boldsymbol{s}, \boldsymbol{a}', \boldsymbol{\theta}^{-}\right)\right)}. \tag{22}$$

Denoting the memory replay buffer as $\mathcal{D}$, the interaction between the agent and environment is concluded in Algorithm 2, where the critic and actor are naturally generalized with the regularization of relative entropy by statistically biasing the deterministic action from actor based on critic's value function smoothed by the Boltzmann softmax operator.

Algorithm 3 CDPP

Initialize critic network $\hat{\Psi}$, actor network $\hat{\mu}$ with random parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$, memory replay buffer $\mathcal{D}$
# Pre-train phase
for $t = 1$ *to* $T_{initial}$ **do**
  # Collect samples with a random policy
  $\mathcal{D} \leftarrow (\boldsymbol{s}_t, \boldsymbol{a}_t, \mathcal{R}^{\boldsymbol{a}_t}_{\boldsymbol{s}_t \boldsymbol{s}_{t+1}}, \boldsymbol{s}_{t+1})$
for $t = 1$ *to* T **do**
  # Interaction phase
  $\mathcal{D} \leftarrow CBSP(\mathcal{D})$
  # Training phase
  Sample mini-batch of J samples from $\mathcal{D}$
  for $j = 1$ *to* J **do**
    $\boldsymbol{a}'_j = \hat{\mu}_{\text{target}}(\boldsymbol{s}_{j+1}, \boldsymbol{\phi}^{-})$
    $\mathcal{M}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s}_j, \boldsymbol{\theta}^{-}) \leftarrow CalMm(\boldsymbol{s}_j, \boldsymbol{a}_j)$
    $\mathcal{M}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s}_{j+1}, \boldsymbol{\theta}^{-}) \leftarrow CalMm(\boldsymbol{s}_{j+1}, \boldsymbol{a}'_j)$
    # Calculate TD error
    $y_j \leftarrow \mathcal{R}^{\boldsymbol{a}_j}_{\boldsymbol{s}_j \boldsymbol{s}_{j+1}} + \gamma \mathcal{M}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s}_j, \boldsymbol{\theta}^{-}) + \hat{\Psi}_{\text{target}}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}^{-}) - \mathcal{M}_\eta \hat{\Psi}_{\text{target}}(\boldsymbol{s}_{j+1}, \boldsymbol{\theta}^{-})$
  # Update $\boldsymbol{\theta}$ by gradient descent optimization
  $\boldsymbol{\theta} \leftarrow \arg\min_{\boldsymbol{\theta}} \frac{\sum_{j=1}^{J} (y_j - \hat{\Psi}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}))^2}{J}$
  # Update $\boldsymbol{\phi}$ by gradient descent optimization
  $\nabla_{\boldsymbol{\phi}} \leftarrow \frac{\sum_{j=1}^{J} \nabla_{\boldsymbol{a}'_j} \hat{\Psi}(\boldsymbol{s}_j, \boldsymbol{a}'_j, \boldsymbol{\theta})|_{\boldsymbol{a}'_j = \hat{\mu}(\boldsymbol{s}_j, \boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}} \hat{\mu}(\boldsymbol{s}_j, \boldsymbol{\phi})}{J}$
  # Update target networks
  $\boldsymbol{\theta}^{-} \leftarrow \tau \boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}^{-}$
  $\boldsymbol{\phi}^{-} \leftarrow \tau \boldsymbol{\phi} + (1-\tau)\boldsymbol{\phi}^{-}$
return $\hat{\Psi}$, $\hat{\mu}$

D. Summary of CDPP

Algorithm 3 summarizes the entire process of the proposed CDPP. Following the main process of DDPG, the proposed method employs main networks of critic and actor $\hat{\Psi}$ and $\hat{\mu}$ with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. The corresponding target networks $\hat{\Psi}_{\text{target}}$ and $\hat{\mu}_{\text{target}}$ are initialized with copied parameters $\boldsymbol{\theta}^{-} \leftarrow \boldsymbol{\theta}$ and $\boldsymbol{\phi}^{-} \leftarrow \boldsymbol{\phi}$ to stabilize the learning process. The initialized memory replay buffer $\mathcal{D}$ is first warmed up by T_{initial} steps samples collected by a random policy. The training procedure includes T steps. At each step, the CDPP agent first interacts with the environment and collects new samples following the Boltzmann sampling policy introduced in Algorithm 2. After the interaction, the agent moves to the training phase.

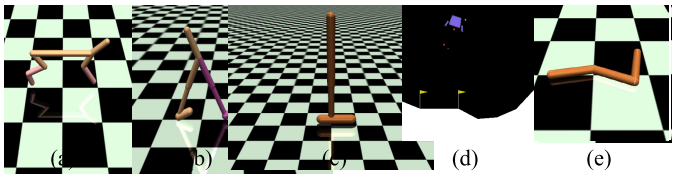


Fig. 2. Benchmark tasks for evaluation in this article. (a) HalfCheetah-v1 (17-D observation and 6-D action). (b) Walker2d-v1 (17-D observation and 6-D action). (c) Hopper-v1 (11-D observation and 3-D action). (d) LunarLanderContinuous-v2 (8-D observation and 2-D action). (e) Swimmer-v1 (8-D observation and 2-D action).

It randomly samples a mini-batch of J transitions from $\mathcal{D}$ and calculates the corresponding TD error using the estimated Mellowmax operation introduced in Algorithm 1. The parameters of the critic and actor are updated by gradient descent optimization following (5) and (6), where the Q function is replaced by the action preferences function. The target networks are then smoothly updated with parameter τ. Please note that the Boltzmann sampling policy is only utilized in the training process, and the agent directly decides deterministic actions from $\hat{\mu}$ in the evaluation phase. Although CDPP closely follows the traditional RL approach based on the AC framework, it suffers from additional computational complexity due to the Monte Carlo sampling in both the estimated Mellowmax operator and the Boltzmann sampling policy, which bridges DPP from a discrete action space to a continuous one. Section IV-C analyzes and discusses this issue as the main challenge of applying CDPP in practice.

IV. Experiments

A. Experimental Settings

In this section, the proposed approach CDPP was evaluated by five benchmark control tasks with different complexities developed in OpenAI Gym [38] and MuJoCo [39]: HalfCheetah-v1, Walker2d-v1, Hopper-v1, LunarLanderContinuous-v2, and Swimmer-v1, as shown in Fig. 2. For the baselines, we selected CDPP without using Mellowmax (denoted as CDPP-B), DPPG [5] and its current extension episodic memory AC (EMAC) [40],[1] the baseline value function approach A3C [19], [20],[2] and a current constrained policy gradients approach CVaR PPO (CPPO) [24][3] to sufficiently investigate the advantages of not only the relative entropy regularization but also the Mellowmax in the proposed AC-based approach. The proposed CDPP and DDPG were developed by PaddlePaddle [41] under its RL toolkit PARL.[4] All experiments were conducted on a computational server with Intel Xeon-W2265 CPU, NVIDIA GeForce RTX-3080Ti GPU, 32-GB memory, and Ubuntu 20.04 OS.

The common parameters and network structures of all compared approaches were listed in Table II. The network structures of all baselines were set according to their open-source code. We developed CDPP and CDPP-B following the network structures of DDPG and EMAC with 256 × 256 multiple-layer perceptron (MLP) and rectified linear unit

[1]EMAC was developed following https://github.com/schatty/EMAC.
[2]A3C was developed following https://github.com/ikostrikov/pytorch-a3c.
[3]CPPO was developed following https://github.com/yingchengyang/CPPO.
[4]https://github.com/PaddlePaddle/PARL

TABLE II
COMMON HYPERPARAMETER SETTINGS OF COMPARED APPROACHES

	CDPP/CDPP-B	DDPG	A3C	CPPO	EMAC
Critic Learning Rate	$5 \cdot 10^{-4}$	10^{-3}	10^{-4}	10^{-3}	10^{-3}
Actor Learning Rate	$5 \cdot 10^{-5}$	10^{-4}	10^{-4}	$3 \cdot 10^{-4}$	10^{-3}
Actic and Critic Structure	(256,256)	(256,256)	(256,256,128,128) ⇒LSTM(128)⇒(128)	(64,32)	(256,256)
Target Update Rate (τ)	5×10^{-3}	10^{-3}	1.0	1.0	5×10^{-3}
Batch Size (J)	64	64	/	4000	256
Discount Factor (γ)	0.99	0.99	0.99	0.99	0.99
Memory Size (E)	10^6	10^6	/	$4 \cdot 10^3$	10^6
Warmup Steps (N_{initial})	10^4	10^4	/	/	10^4
Exploration Noise	/	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$	/	/	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$

TABLE III
IMPORTANCE RANKING AND VALUES OF HYPERPARAMETERS FOR CDPP AND CDPP-B ALGORITHMS

Rank	Hyperparameter	Value	Range
1	KL Temperature (η)	0.05	$[5 \times 10^{-4}, 1.0]$
2	Policy Sampling Number	50	$[1, \infty)$
3	Softmax Sampling Number	30	$[1, \infty)$
4	Sampling Noise variance	0.1	$(0, \infty)$
5	Exploration Noise variance	0.1	$(0, \infty)$

(ReLU) activation function in both critic and actor. Other common parameters in CDPP were mainly selected based on DDPG's SOTA performance [18], except for a smaller learning rate and a larger target update rate to fully adapt to the KL regularization term. The specific parameters of baselines were set based on their paper and open-source code. All tunable parameters of CDPP were summarized in Table III with a rank of importance. The most important parameter η controls the KL regularization term, and it was set to $\eta = 0.05$ based on the experiment in Section IV-D. The Monte Carlo samples in both estimated Mellowmax operation (Algorithm 1) and Boltzmann sampling policy (Algorithm 2) followed a Gaussian noise with a variance of 0.1, which is the same as the exploration noise in DDPG and EMAC. Their corresponding sampling numbers were set as 30 and 50 separately to balance the learning capability and computational efficiency. Please note that all parameters of CDPP were selected in a general way to fairly compare with other baselines, and it is possible to further enhance the performance of CDPP on specific tasks with fining-tuned parameters.

B. Evaluation of the Learning Capability

The learning capability of CDPP was first evaluated in this section. All compared approaches were trained in one million time steps and evaluated every 5000 steps. Each evaluation reported the average reward over five episodes without exploration noise. All experimental results were conducted by five independent trials with different random seeds for statistical evidence.

Fig. 3 illustrated the average learning curves of all approaches over five benchmark tasks. The corresponding max average returns with standard deviation were reported in Table IV, where the maximum values were bolded. It is observed that the proposed CDPP with the Mellowmax operator outperformed CDPP-B without the Mellowmax operator and the original DDPG overall most tasks in not only the sample efficiency but also the robustness after quick convergences. This result clearly indicated the superior learning capability in AC structure by introducing the relative entropy regularization into DDPG. In addition, CDPP-B with the Boltzmann operator also outperforms DDPG in most tasks except swimmer, but could not reach the performances of the proposed CDPP with the Mellowmax operator. In the hopper task, CDPP-B achieved the highest average return in the middle of learning but quickly degenerated to the level of DDPG. As a comparison, CDPP successfully maintained high average returns without any oscillation. In other tasks, the learning performances of CDPP-B were all more degraded than CDPP, especially in the swimmer task where the learning capability was even worse than DDPG. Please note that DDPG achieved the highest return but could not maintain it while its convergence was far slower than CDPP. This result indicated the positive effect of the Mellowmax operator in CDPP to avoid the local optimum and improper behavior during the RL process regularized by the relative entropy. CDPP also demonstrated significant superiorities in convergence and average returns compared with other related works, including the baseline value function approach A3C, the constrained policy gradients approach CPPO, and EMAC, which is an extension of DDPG with SOTA performance. A3C only achieved comparable performance in the LunarLander task. CPPO demonstrated overall slow sample efficiency due to its on-policy nature. EMAC outperformed CDPP in the walker task and achieved similar performance in the hopper task, but converged slowly with fewer average returns in other tasks.

In this work, we defined the sample efficiency as the number of interactions with the environment used to complete the learning process, which is important to the implementation of RL in real-world hardware. The sample efficiency of the proposed method was evaluated in Fig. 4, where the bars with different colors present the number of steps used for each approach to reach the lower boundary of the max average returns in Table IV over five tasks. Overall, CDPP utilized far fewer interactions (about 16% than CDPP-B, 33% than DDPG, 68% than A3C, 58% than CPPO, and 40% than EMAC) to

TABLE IV
MAX AVERAGE RETURNS OVER FIVE BENCHMARK TASKS

	CDPP	CDPP-B	DDPG	A3C	CPPO	EMAC
HalfCheetah	**9778.59±403.35**	9159.85±451.23	9316.27±1198.09	640.84±212.37	1875.99±362.97	3867.25±752.85
Walker2d	2530.71±397.55	2281.69±303.12	1879.67±384.23	957.02±101.85	1087.24±82.79	**4153.98±571.65**
Hopper	2681.72±321.86	**2883.29±262.96**	2205.49±308.64	948.86±121.52	1659.03±114.30	2654.66±733.83
LunarLanderContinuous	**225.91±21.71**	223.45±50.09	115.95±106.64	207.14±32.39	7.61±28.00	153.38±113.75
Swimmer	144.43±16.21	134.92±7.92	**148.25±15.44**	44.67±2.14	116.77±10.11	44.74±9.98

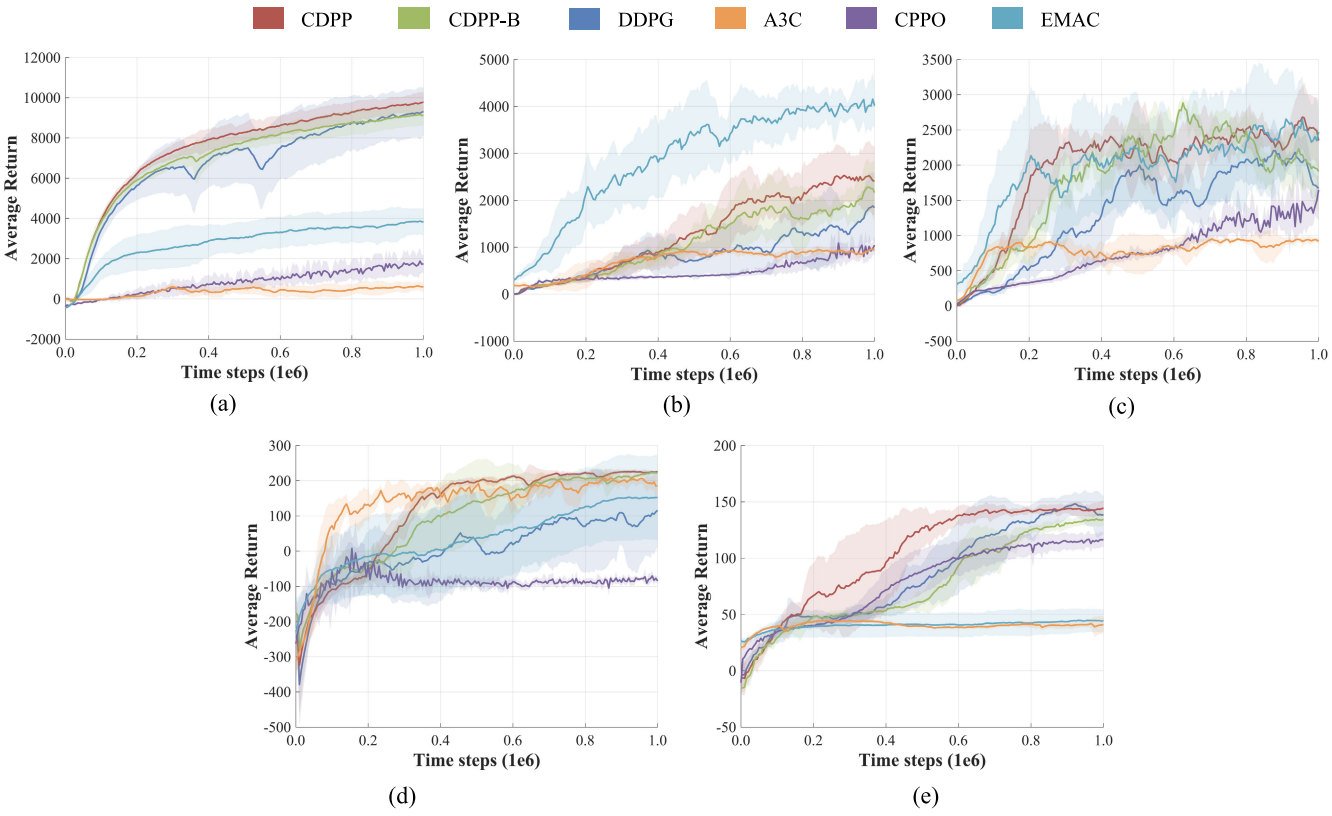


Fig. 3. Learning curves over five benchmark tasks. The shaded region represents the standard deviation of the average evaluation over five independent trials. Curves are uniformly smoothed for visual clarity. (a) HalfCheetah-v1. (b) Walker2d-v1. (c) Hopper-v1. (d) LunarLanderContinuous-v2. (e) Swimmer-v1.

reach the low boundary of performances over all five tasks compared with other approaches. This result indicated the superior sample efficiency of CDPP, which is consistent with the results in its previous works with discrete action [29], [30], [32] and clearly showed the positive impact of the KL divergence in practical RL.

C. Implementation on Real-Robot-Based Simulation

In this section, CDPP was compared with other baselines in a real-robot-based simulation robo-gym [42] to evaluate its potential in more complex hardware. We selected the end-effector position task using a UR5 robot arm and obstacle avoiding task using a Mir100 mobile robot, as shown in Fig. 5. CDPP-B was not compared in this section due to its suboptimal learning performances in previous results.

The learning curves of average returns in two tasks were demonstrated in Fig. 6. Except for EMAC, which achieved superior learning performance to CDPP in the end-effector position task, CDPP significantly outperformed all comparison methods in terms of sampling efficiency and max average returns in both UR5 and Mir100 robots. This result indicated the potential of the proposed method in real-robot systems.

To analyze the additional computational burden of the Monte Carlo sampling in CDPP, the average time (including decision-making and networks update) of all approaches over every 50 000 step was compared in Fig. 7. The additional computational burden of CDPP was acceptable with a suitable setting of Monte Carlo sampling number in these two tasks. In the obstacle avoiding task with 20-D state and 2-D action, CDPP achieved a close computational time to DDPG and A3C. Although it was about 37% slower than CPPO, it outperformed EMAC with over 21% less computational time. Turning to the end-effector position task with 27-D state and 5-D action, CDPP worked a bit slower than DDPG and CPPO with 7% and 4% more time. A3C achieved the best computational efficiency in this task because of its asynchronous nature, while the SOTA approach EMAC had a heavy computational burden.

D. Evaluation of the Relative Entropy Regularization

In this section, the effect of parameter η in CDPP that controls the regularization of relative entropy was evaluated in

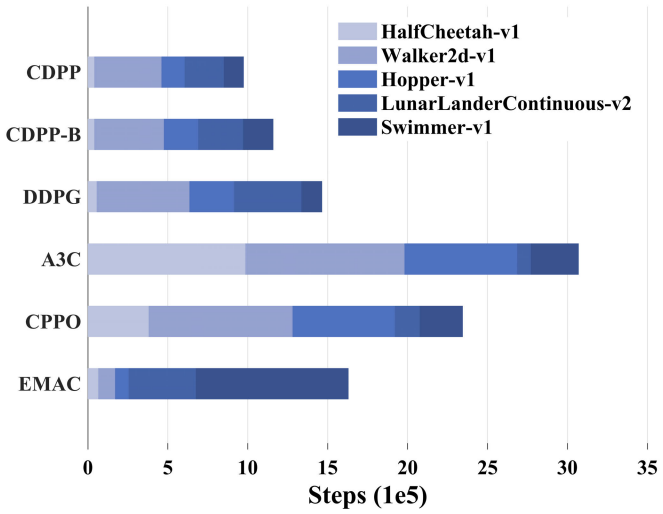
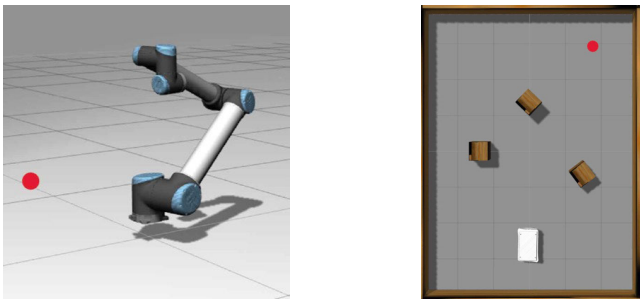


Fig. 4. Number of interactions used by compared approaches to finish different tasks.



(a) (b)

Fig. 5. Real-robot-based tasks. (a) End-effector position task using a UR5 robot arm (27-D observation and 5-D action). (b) Obstacle avoiding task using a Mir100 mobile robot (20-D observation and 2-D action).

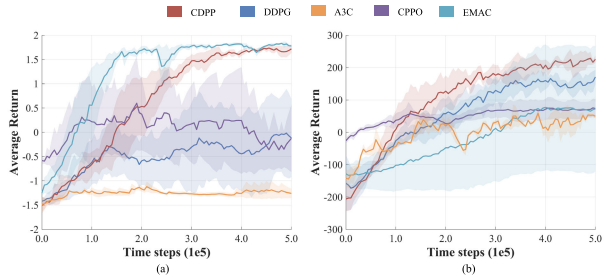


Fig. 6. Learning curves of two real-robot-based simulation tasks. The shaded region represents the standard deviation of the average evaluation over five independent trials. Curves are uniformly smoothed for visual clarity. (a) EndEffectorPositioningURSim-v0. (b) ObstacleAvoidanceMir 100Rob-v0.

the Halfcheetah task by setting $\eta = [0.15, 0.05, 0.005, 0.0005]$ and compared with DDPG. Other experimental settings followed Section IV-A. From the experimental result demonstrated in Fig. 8, it is clearly observed that all candidate CDPP outperformed DDPG regardless of η in the early and middle periods of training with superior stability. On the other hand, both $\eta = 0.15$ and $\eta = 0.0005$ resulted in poor learning capability after time step 6×10^5: the overlarge η turned to insufficient constrained policies with less sample efficiency, while the oversmall η could slow down the convergence by learning over conservative policies that are far from the optimal one. This result was consistent with related works of the relative entropy regularized RL [29], [30]. As a comparison,

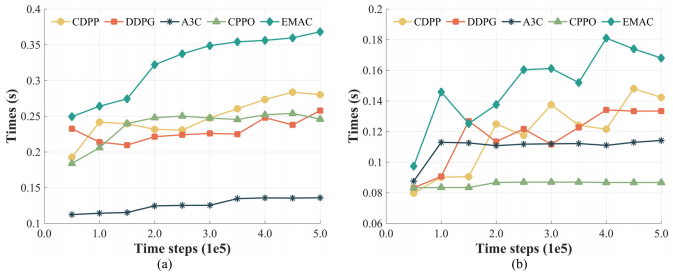


Fig. 7. Average calculation time of two real-robot-based simulation tasks. (a) EndEffectorPositioningURSim-v0. (b) ObstacleAvoidanceMir100Rob-v0.

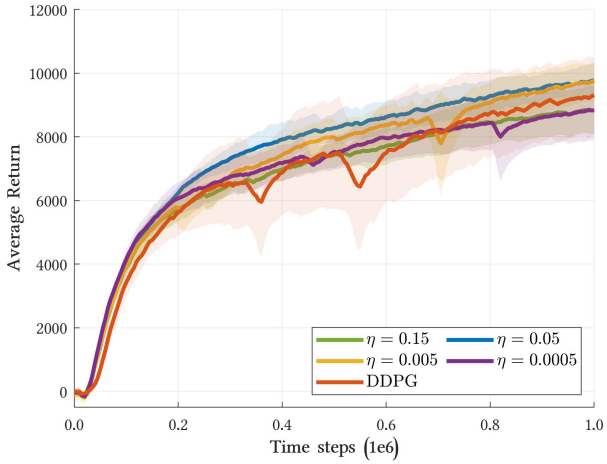


Fig. 8. Learning curves of the CDPP with different values of η compared with DDPG in the Halfcheetah task. The shaded region represents the standard deviation of the average evaluation over five trials. Curves are smoothed uniformly for visual clarity.

the proper settings of $\eta = [0.05, 0.005]$ could balance the penalty items between the overlarge policy update and the speed of convergence. This result indicated the importance of properly selecting the parameter, and $\eta = 0.05$ was selected for all other experiments based on its superior converge speed and average return in the final step.

E. Evaluation of the Smooth Policy Update

In this section, the smooth policy update in CDPP was investigated. We first explored the effect of the relative entropy regularization in the sampling behavior of CDPP under the Walker2d task with the comparison of DDPG. The samples collected by both CDPP and DDPG during $(0, 10]$, $(30, 40]$, $(60, 70]$, and $(90, 100]$ 1000 steps in one trial with the same random seed were visualized by t-distributed stochastic neighbor embedding (t-SNE) [43] in Fig. 9, where the colors from blue to red presented the corresponding value function (Ψ and Q) with low to high values.

At the early stage of learning ($t \leq 10^4$), DPPG drove a more general exploration among the whole state space, while CDPP was stuck in relatively low value states due to the policy constrained by the relative entropy. However, compared with the unfocused exploratory behavior of DDPG with the increase of time step from 30k to 40k, CDPP accelerated its learning by smoothly exploring near the previously explored area and successfully obtained relatively high value

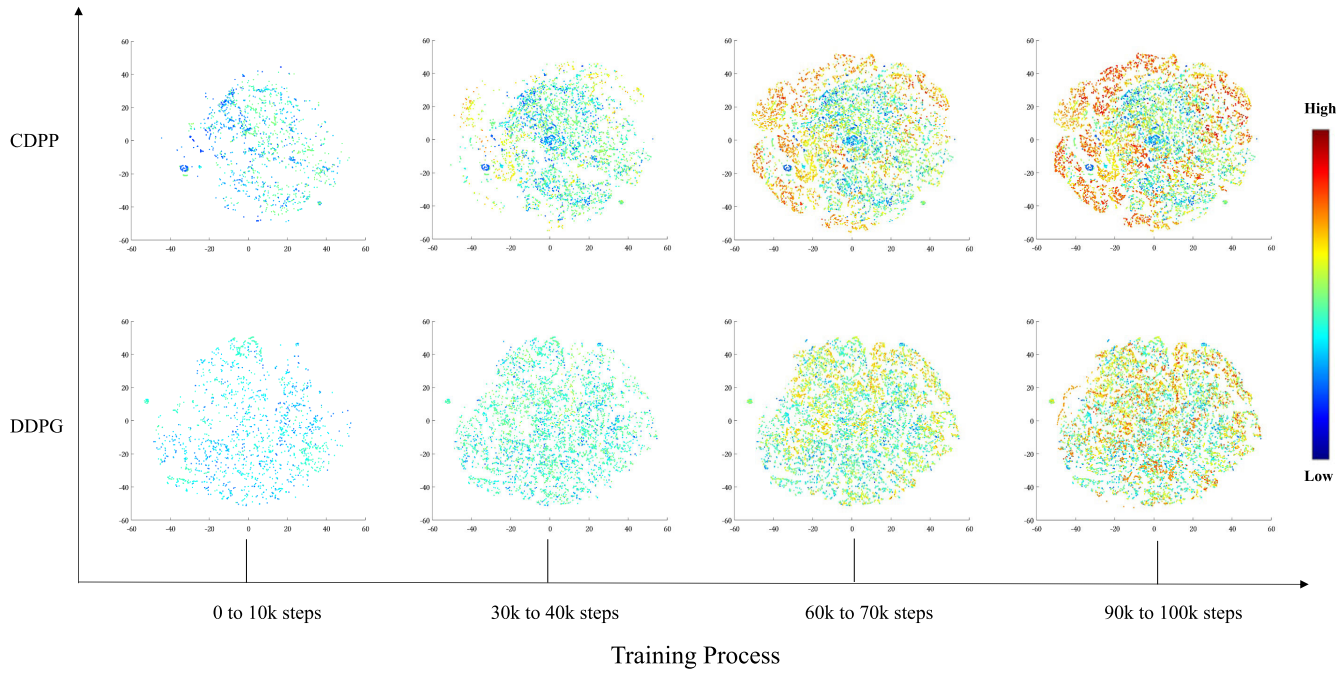


Fig. 9. Visualized samples of CDPP and DDPG during the learning procedure. The color from blue to red represents the value function from small to large.

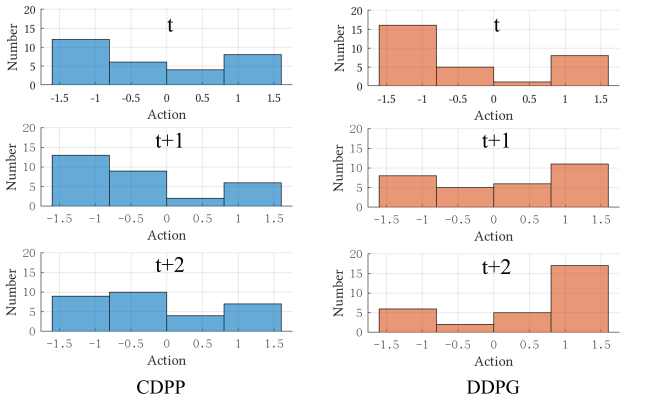


Fig. 10. Update of action distributions in one dimension using CDPP and DDPG over three consecutive time steps in the hopper task.

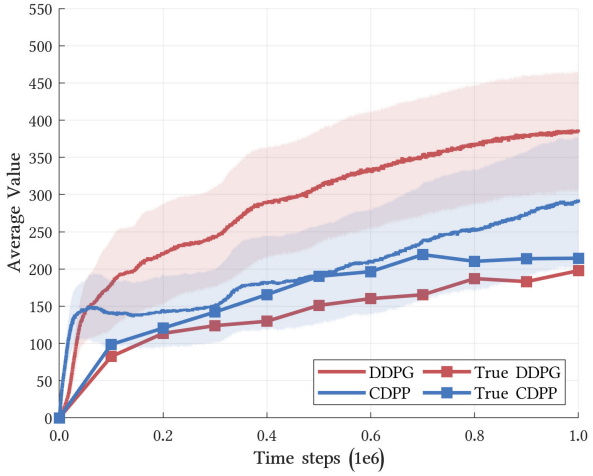


Fig. 11. Overestimation of the value function in DDPG and CDPP in the Walker2d task over 1 million time steps.

states in local areas. In the following learning procedure, DPPG collected relatively low value states without focusing on the local areas of high reward. As a comparison, CDPP efficiently explored the state space with high value function with the smoothness constrained by the previous policies and, therefore, enjoyed a better learning capability than DPPG: over 25% improved converged return and 40% faster convergence according to the learning curve of the Walker2d task in Fig. 8. This result demonstrated the superiority of the proposed CDPP in a unique learning process: it efficiently explored the high dimensional state space and collected more samples with high rewards, which contributed to better sample efficiency and learning capability compared with the baseline approaches.

Next, we evaluated the smoothness of policy updates by one case study of the policy's action distributions over three consecutive time steps in the hopper task. The distributions were calculated based on 30 observations randomly sampled from the current memory buffer. For simplicity, only one dimension of the action was analyzed. According to the statistical histograms shown in Fig. 10, it is clearly observed that the average distribution of action in CDPP enjoyed a relatively smooth change within three consecutive time steps, while the one in DPPG was more drastically updated. This observation coincided with the positive effect of the relative entropy in exploration behavior from the view of action distributions and further indicated the benefit of the smooth policy update in CDPP compared with the original DDPG.

F. Evaluation of the Overestimation

The last experiment focused on the overestimation in the AC framework reported by Fujimoto et al. [18]. Fig. 11 illustrated the approximated value functions and the corresponding true values of both CDPP and DPPG in the Walker2d task following the same hyperparameters introduced in Section IV-A. Following [18], the approximated value functions were estimated based on 10 000 states, and the true value functions

were estimated by averaging the discounted returns over 1000 episodes under the current learned policies. The sampled states in approximated value and the start states in true value were all randomly sampled from the current memory buffer.

We can observe that the overestimation of the value function was significantly alleviated in CDPP under the regularization of the relative entropy. Starting from a bit large value, the overestimation of CDPP was quickly reduced during the learning and finally maintained on a limited scale. As a comparison, DDPG achieved a deteriorating overestimation over the learning process with a far larger mismatch between the approximated and true values than CDPP. This result clearly demonstrated the advantage of CDPP in tackling the overestimation issue in DDPG and validated the theoretical contribution of the relative entropy regularized RL [27] in averaging the approximated error of value function under the AC structure.

G. Advantage and Limitation of CDPP

Based on the studies in Sections IV-D–IV-F, the superiority of CDPP in the max average return and sample efficiency mainly came from the KL-divergence regularization. In terms of policy update, the KL-divergence regularization turned to smoother action distributions during the learning process, as shown in Fig. 10. It contributed to a more efficient exploration in Fig. 9 to quickly find high reward samples with fewer number of interaction. In terms of the overestimation of the value function, Fig. 11 demonstrated a significant reduction of the approximation error. It is consistent with the theoretical study that the relative entropy regularization would implicitly average the error of the approximated value function [27]. All these results indicated that the benefit of the relative entropy regularization was kept after translating to the AC framework with continuous action space through the Monte Carlo sampling.

For the limitation of the proposed method, although CDPP has several advantages compared with most baselines, such as DDPG, it falls behind the SOTA methods, such as EMAC in the walker and end-effector position tasks. This result indicates that CDPP may not achieve superior performance than other approaches in all control tasks. It is also observed that an improper value of the constraint parameter η can lead to a degradation of learning capability. The computational burden with a large number of Monte Carlo samplings could further limit its implementation in more challenging scenarios.

V. Conclusion

In this article, a novel RL approach, CDPP, was proposed to tackle the issues of the sample efficiency and learning stability in RL with continuous action space by extending the relative entropy regularized RL from the value function-based approach to the AC framework-based one. The proposed method successfully estimated the intractable Boltzmann softmax operation over continuous actions in the critic by Monte Carlo sampling and explored the benefit of employing the Mellowmax operator in convergence capability. A Boltzmann sampling policy was also designed in CDPP to naturally guide the exploration of actors following the relative entropy regularized critic. Evaluated by several benchmark and real-robot-based simulation tasks with different complexities, CDPP demonstrated its significant superiority in both learning stability and sample efficiency compared with related baseline approaches. The positive effect of CDPP in efficiently exploring the high-dimensional state–action space, smoothly updating the action distribution of the policy, and alleviating the overestimation of the approximated value function was studied and illustrated. All these results indicated the potential of CDPP as an emerging stable and sample-efficient RL approach for various control problems with continuous action space. Our future work will mainly focus on addressing the current limitations of CDPP to meet the requirements of real-time robot control applications. In terms of parameter selection, we will explore an effective training method to adaptively change η according to the learning performance. It is also interesting to reduce the computational burden of Monte Carlo sampling from the perspective of the distributional value function.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[3] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[4] 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.
[5] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor–critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[8] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[9] M. L. Koga, V. Freire, and A. H. R. Costa, "Stochastic abstract policies: Generalizing knowledge to improve reinforcement learning," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 77–88, Jan. 2015.
[10] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 777–787, Mar. 2019.
[11] W. Shi, S. Song, C. Wu, and C. L. P. Chen, "Multi pseudo Q-learning-based deterministic policy gradient for tracking control of autonomous underwater vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3534–3546, Dec. 2019.
[12] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.
[13] M. A. Lee et al., "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8943–8950.
[14] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Proc. Conf. Robot Learn. (CoRL)*, 2021, pp. 188–204.
[15] Y. Yu, J. Tang, J. Huang, X. Zhang, D. K. C. So, and K.-K. Wong, "Multi-objective optimization for UAV-assisted wireless powered IoT networks based on extended DDPG algorithm," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6361–6374, Sep. 2021.
[16] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3389–3396.
[17] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Proc. Conf. Robot Learn. (CoRL)*, 2017, pp. 262–270.

[18] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor–critic methods," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1587–1596.
[19] S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding Atari agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1792–1801.
[20] A. B. Labao, M. A. M. Martija, and P. C. Naval, "A3C-GS: Adaptive moment gradient sharing with locks for asynchronous actor–critic agents," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1162–1176, Mar. 2021.
[21] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor–critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6584–6598, Nov. 2022.
[22] F. Li et al., "Improving exploration in actor–critic with weakly pessimistic value estimation and optimistic policy optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 28, 2022, doi: 10.1109/TNNLS.2022.3215596.
[23] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization," in *Uncertainty in Artificial Intelligence*. Proceedings of Machine Learning Research (PMLR), 2020, pp. 113–122.
[24] C. Ying, X. Zhou, H. Su, D. Yan, N. Chen, and J. Zhu, "Towards safe reinforcement learning via constraining conditional value-at-risk," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 3673–3680.
[25] W. Meng, Q. Zheng, Y. Shi, and G. Pan, "An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 5, pp. 2223–2235, May 2022.
[26] M. G. Azar, V. Gómez, and H. J. Kappen, "Dynamic policy programming," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 3207–3245, 2012.
[27] T. Kozuno, E. Uchibe, and K. Doya, "Theoretical analysis of efficiency and robustness of softmax and gap-increasing operators in reinforcement learning," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2019, pp. 2995–3003.
[28] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist, "Leverage the average: An analysis of KL regularization in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 12163–12174.
[29] Y. Cui, T. Matsubara, and K. Sugimoto, "Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states," *Neural Netw.*, vol. 94, pp. 13–23, Oct. 2017.
[30] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robot. Auto. Syst.*, vol. 112, pp. 72–83, Feb. 2019.
[31] L. Zhu, Y. Cui, G. Takami, H. Kanokogi, and T. Matsubara, "Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process," *Control Eng. Pract.*, vol. 97, Apr. 2020, Art. no. 104331.
[32] L. Zhu, G. Takami, M. Kawahara, H. Kanokogi, and T. Matsubara, "Alleviating parameter-tuning burden in reinforcement learning for large-scale process control," *Comput. Chem. Eng.*, vol. 158, Feb. 2022, Art. no. 107658.
[33] Y. Cheng, L. Huang, C. L. P. Chen, and X. Wang, "Robust actor–critic with relative entropy regulating actor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9054–9063, Nov. 2023.
[34] K. Asadi and M. L. Littman, "An alternative softmax operator for reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 243–252.
[35] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6287–6296.
[36] S. Wang, Y. Pu, S. Yang, X. Yao, and B. Li, "Boltzmann exploration for deterministic policy optimization," in *Proc. Int. Conf. Neural Inf. Process. (ICONIP)*. Cham, Switzerland: Springer, 2020, pp. 214–222.
[37] E. Todorov, "Linearly-solvable Markov decision problems," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2006, pp. 1369–1376.
[38] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.
[39] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[40] I. Kuznetsov and A. Filchenkov, "Solving continuous control with episodic memory," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 2651–2657.
[41] Y. Ma, D. Yu, T. Wu, and H. Wang, "PaddlePaddle: An open-source deep learning platform from industrial practice," *Frontiers Data Domputing*, vol. 1, no. 1, pp. 105–115, 2019.
[42] M. Lucchi, F. Zindler, S. Mühlbacher-Karrer, and H. Pichler, "Robogym—An open source toolkit for distributed deep reinforcement learning on real and simulated robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5364–5371.
[43] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 1–12, 2008.

Zhiwei Shang received the B.S. degree in hydrology and water resources engineering from Sichuan University, Chengdu, China, in 2020, and the M.S. degree in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2023.

He is currently a Research Assistant at The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests mainly include reinforcement learning and intelligent control.

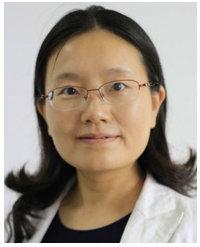
Renxing Li received the B.S. degree in electrical engineering and automation from Tongji University, Shanghai, China, in 2018, and the M.S. degree in software engineering from the University of Science and Technology of China, Hefei, China, in 2023.

She is currently a Research Assistant with the Singapore University of Technology and Design (SUTD), Singapore. Her research interests mainly include reinforcement learning and machine learning.

Chunhua Zheng (Member, IEEE) received the M.S. degree in mechanical engineering from the Harbin Institute of Technology, Harbin, China, in 2007, and the Ph.D. degree in mechanical engineering from Seoul National University, Seoul, South Korea, in 2012.

She is currently an Associate Professor of the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. Her working field includes the energy management of new energy vehicles and the fuel cell system management.

Dr. Zheng works as an Editorial Board Member of *International Journal of Precision Engineering and Manufacturing-Green Technology* and an Editor of *International Journal of Automotive Technology*.



Huiyun Li (Senior Member, IEEE) received the M.Eng. degree in electronic engineering from Nanyang Technological University, Singapore, in 2001, and the Ph.D. degree from the University of Cambridge, Cambridge, U.K., in 2006.

She is currently a Professor at the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, and The Chinese University of Hong Kong, Hong Kong. Her research interests include automobile electronics and autonomous vehicles.

Yunduan Cui (Member, IEEE) received the B.E. degree in electronic engineering from Xidian University, Xi'an, China, in 2012, the M.E. degree in computer science from Doshisha University, Kyoto, Japan, in 2014, and the Ph.D. degree in computer science from the Nara Institute of Science and Technology, Ikoma, Japan, in 2017.

From 2017 to 2020, he was a Post-Doctoral Researcher with the Nara Institute of Science and Technology. He is currently an Associate Professor at the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His research interests focus on machine learning, especially reinforcement learning-driven unmanned systems.