Reduce Overestimation by Kullback-Leibler Divergence Regularized Distributional Actor-Critic

Journal:	*IEEE Transactions on Neural Networks and Learning Systems*
Manuscript ID	TNNLS-2023-P-29282
Manuscript Type:	Regular Paper
Date Submitted by the Author:	27-Jul-2023
Complete List of Authors:	Gong, Mingrong; Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences Yi, Zhengkun; Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, Li, Huiyun; Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, Cui, Yunduan; Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, Wu, Xinyu; Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences; CAS Key Laboratory of Human-Machine-Intelligence Synergic Systems, Shenzhen Institutes of Advanced Technology; Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Keywords:	Reinforcement Learning, Distributional Reinforcement Learning, Kullback-Leibler Divergence Regularization, Overestimation Alleviation

Reduce Overestimation by Kullback-Leibler Divergence Regularized Distributional Actor-Critic

Mingrong Gong, Zhengkun Yi, Huiyun Li, Yunduan Cui, Xinyu Wu

***Abstract*—In this work, we propose a novel distributional learning (RL) approach, Kullback-Leibler Divergence regularized Distributional Actor-Critic (KDAC) to tackle the issue of overestimated value function and sample efficiency in the current RL domain. Clipping the high-value quantiles in the distributional value function, the proposed KDAC efficient reduces the overlarge estimated value function from a distributional perspective without employing additional critic networks. The learning process of both actor and critic networks in KDAC is further accelerated and stabilized by introducing the Kullback-Leibler divergence regularization between the current and previous policies. Evaluated by Several benchmark tasks with different levels of complexity, KDAC demonstrates significant superiority in learning capability, overestimation reduction and sample efficiency compared with various traditional and state-of-the-art RL baselines and expands the potential of DRL in more complicated control scenarios.**

I. Introduction

In recent years, deep reinforcement learning (DRL), which combines the power of both deep neural networks [1] and reinforcement learning (RL) [2], has emerged as a promising approach to artificial intelligence. It explores optimal or sub-optimal policies that outperform human-level performance in challenging tasks such as video games, go, and other board games thanks to its capability of iteratively exploring unknown environments without prior knowledge [3]–[6]. One important direction of DRL is off-policy approaches based on the Actor-Critic (AC) framework. These approaches have shown great potential in many control scenarios with continuous action spaces due to their efficient utilization of historical training samples, making them an important direction of DRL [7], [8]. However, their fatal flaws such as the overestimation bias of the value function and accumulated approximation errors lead to unstable learning procedures, unwanted control behaviors and slow convergence which become major impediments to real-world control implementations like robots and unmanned vehicles as reported by [9]–[11].

This work is supported in part by the National Natural Science Foundation of China under Grant 62103403 and Grant U22A2056; in part by Guangdong Provincial Basic and Applied Basic Research Foundation under Grant 2020B515130004.

Corresponding author: Yunduan Cui (email: cuiyunduan@gmail.com)

Mingrong Gong is with Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China.

Zhengkun Yi, Huiyun Li, Yunduan Cui and Xinyu Wu are with CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, and 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.

Many works attempt to tackle the overestimation of the value function under the AC framework. Soft Actor-Critic (SAC) [12] was proposed to alleviate the overestimation bias by additionally approximating the value function V besides the Q function. Twin Delayed Deep Deterministic Policy Gradient (TD3) [13] introduced two independent critic networks to reduce the overestimation of the value function. Based on TD3, Triplet-Average Deep Deterministic Policy Gradient (TADD) [14] employed triple critics and achieved superior learning performances with less overestimation compared with TD3. Inspired by the distributional RL approaches like C51 [15] and Quantile Regression Deep Q Network (QRDQN) [16] that estimate the value function as random distributions, Truncated Quantile Critics (TQC) [17] ensemble multiple distributional critic networks to alleviate the overestimation bias from a stochastic perspective and enjoyed superior control performances in Mujoco simulation. These approaches attempted to alleviate the overestimation of value function by employing additional neural networks. It turns to not only a heavy computational burden but also an unstable learning procedure. Furthermore, a practical RL approach that simultaneously addresses both overestimation and accumulated approximation errors remains unexplored.

As a potential solution to the accumulated errors in RL using function approximation, Dynamic Policy Programming (DPP) first employed the Kullback-Leibler (KL) divergence between previous and current policies as a regularization term of the value function [18]. This approach has demonstrated superiority in both theory and engineering. The KL divergence regularization in DPP contributes to a superior error dependency where the accumulated errors of value function approximation are implicitly averaged, significantly reducing the upper boundary of accumulated errors in theory [19], [20]. In terms of engineering applications, DPP-based approaches using different function approximators such as kernel functions and deep neural networks have been successfully implemented in a wide range of real-world control scenarios, including robot manipulation [21], [22] and large-scale chemical plant control [23]. These approaches outperform baselines in both sample efficiency and learning stability. Despite the advantages above, the application of DPP and the KL divergence regularization was heavily limited due to their nature of only supporting discrete action space. Their extension to the AC framework remains an open issue.

In this paper, we propose KL Divergence regularized Distributional Actor-Critic (KDAC) to naturally leverage distributional RL [15], [16] and KL divergence regularized RL [18]–[20] under an AC framework. It simultaneously addresses the

TABLE I
COMPARISON OF KDAC AND RELATED APPROACHES

Approach	KL regularization	Actor-Critic	Distributional	Overestimation
DDPG [7]	×	○	×	×
SAC [12], TD3 [13]	×	○	×	○
QRDQN [16]	×	○	○	×
TQC [17]	×	○	○	○
DPP [18]	○	×	×	×
KDAC (ours)	○	○	○	○

overestimation bias of the value function and accumulated approximation errors in current off-policy DRL approaches without employing additional neural networks. Inheriting the distributional perspective from QRDQN [16], KDAC alleviates the overestimation bias in its signal critic network by adaptively truncating the overlarge quantiles according to the learning process. This is done instead of clipping larger values over multiple critics in TD3 [13] and TQC [17]. To reduce the accumulated errors over the learning process, KDAC approximates the value function regularized by KL divergence following DPP [18] in critic and naturally merges it to the AC framework using a Monte-Carlo sampling-based actor networks. Evaluated by several benchmark control tasks, KDAC significantly outperforms state-of-the-art baselines in terms of convergence, sample efficiency and overestimation alleviation while enjoying superior computational efficiency than the related methods employing additional neural networks. With alleviated overestimation and fewer interactions required to converge, KDAC successfully expands the potential of DRL in complicated control scenarios like robots and unmanned vehicles. Based on the features summarized in Table I, the contributions of this paper compared with the related RL approaches are:

1) Proposing a novel and efficient solution to alleviate the overestimated value function in off-policy DRL. KDAC suppresses overlarge value estimations by adaptively truncating the corresponding quantiles of the value function's distribution, It reduces the computational burden of multiple critics employed in previous works.
2) Merging the KL divergence regularization into the AC framework with a distributional value function to stabilize and accelerate the learning process of off-policy DRL while alleviating the overestimation bias. It can be seen as a synchronous extension of DPP in both continuous action space and distributional RL.
3) Evaluating the proposed KDAC through various benchmark control tasks compared with related baselines. Experimental results demonstrate its significant advantages in learning capability, computational efficiency and overestimation alleviation.

The paper is organized as follows: Section II presents the preliminaries; the proposed approach KDAC is detailed in Section III; Section IV demonstrates the experimental results; Section V gives the conclusions.

II. PRELIMINARIES

A. Markov Decision Process

RL considers the paradigm of an agent interacting with the environment. This process is formulated as Markov Decision Process (MDP) with a 5-tuple $(\mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ denote the spaces of state $\boldsymbol{s}$ and action $\boldsymbol{a}$, respectively, $r(\boldsymbol{s}, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function designed for a specific task, $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ denotes the transition probability, $\gamma \in [0, 1)$ is the discount factor. The control policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ maps state to a probability distribution over the action space. The objective of the RL agent is to optimize its policy to maximize the obtained reward in a long term which is defined as the value function:

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

where $\boldsymbol{s}_0 = \boldsymbol{s}$, $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$, $\boldsymbol{a}_0 = \boldsymbol{a}$, $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$. The maximized long-term reward of specific state $\boldsymbol{s}$ and state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ are presented as optimal value functions V^* and Q^* following Bellman equations:

$$V^*(\boldsymbol{s}) = \max_\pi \sum_{\boldsymbol{a}\in\mathcal{A}} \pi(\boldsymbol{a}|\boldsymbol{s})\Big[r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})V^*(\boldsymbol{s}')\Big], \tag{2}$$

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})V^*(\boldsymbol{s}'). \tag{3}$$

B. Kullback-Leibler Divergence Regularized RL

DPP [18] employed the KL divergence between previous policy $\bar{\pi}$ and current policy π to the value function as a regularization term:

$$\mathbb{D}_{\mathrm{KL}}(\boldsymbol{s}_t) = \sum_{\boldsymbol{a}_t\in\mathcal{A}} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) \log\left(\frac{\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\bar{\pi}(\boldsymbol{a}_t|\boldsymbol{s}_t)}\right). \tag{4}$$

Define parameter η to control the scale of KL divergence, the value function became:

$$V^\pi_{\bar{\pi}}(\boldsymbol{s}) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \big(r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) - \frac{1}{\eta}\mathbb{D}_{\mathrm{KL}}(\boldsymbol{s}_t)\big)\right]. \tag{5}$$

It is convenient to present the regularized Q function as action preferences function [2] in DPP:

$$\Psi_{t+1}(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{\eta}\log\bar{\pi}_t(\boldsymbol{a}|\boldsymbol{s}) + r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in\mathcal{S}} \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})V^\pi_{\bar{\pi},t}(\boldsymbol{s}'). \tag{6}$$

The update of value function and policy from iteration t to $t + 1$ was obtained following [24], [25]:

$$V^\pi_{\bar{\pi},t+1}(\boldsymbol{s}) = \frac{1}{\eta}\log\sum_{\boldsymbol{a}\in\mathcal{A}} \exp\big(\eta \cdot \Psi_t(\boldsymbol{s}, \boldsymbol{a})\big), \tag{7}$$

$$\bar{\pi}_{t+1}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp\big(\eta \cdot \Psi_t(\boldsymbol{s}, \boldsymbol{a})\big)}{\sum_{\boldsymbol{a}'\in\mathcal{A}} \exp\big(\eta \cdot \Psi_t(\boldsymbol{s}, \boldsymbol{a}')\big)}. \tag{8}$$

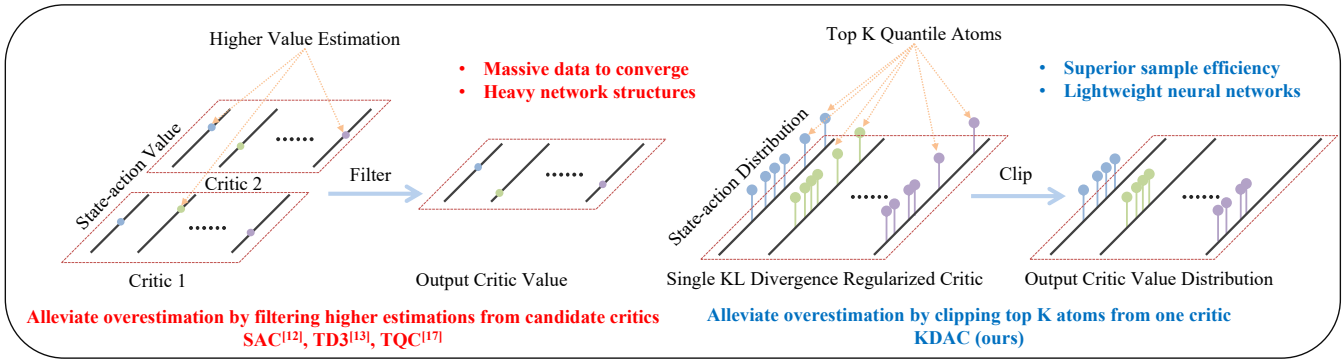


Fig. 1. Principle of the proposed KDAC in alleviating the overestimation by single Critic from a distributional perspective.

Given sample $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, the action preferences function Ψ can be updated by inserting Eqs. (7) and (8) into Eq. (6):

$$\Psi_{t+1}(\boldsymbol{s}, \boldsymbol{a}) = \Psi_t(\boldsymbol{s}, \boldsymbol{a}) - \mathcal{L}_\eta \Psi_t(\boldsymbol{s}) + r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \mathcal{L}_\eta \Psi_t(\boldsymbol{s}'). \tag{9}$$

In practical, $\mathcal{L}_\eta \Psi(\boldsymbol{s}) = \frac{1}{\eta} \log\left(\sum_{\boldsymbol{a}\in\mathcal{A}} \exp\left(\eta \Psi(\boldsymbol{s}, \boldsymbol{a})\right)\right)$ can be replaced by a more tractable Boltzmann softmax operator $\mathcal{B}_\eta \Psi(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\mathcal{A}} \frac{\exp(\eta\Psi(\boldsymbol{s},\boldsymbol{a}))\Psi(\boldsymbol{s},\boldsymbol{a})}{\sum_{\boldsymbol{a}'\in\mathcal{A}} \exp(\eta\Psi(\boldsymbol{s},\boldsymbol{a}'))}$ according to [18].

C. Distributional RL

The distributional RL was first proposed by [15]. It leverages the state-value distribution $Z^\pi(\boldsymbol{s}, \boldsymbol{a})$ under policy π:

$$Z(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma Z(\boldsymbol{s}', \boldsymbol{a}'), \tag{10}$$

$$Q(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[Z(\boldsymbol{s}, \boldsymbol{a})]. \tag{11}$$

Following [26], the distributional RL approximates $Z^\pi(\boldsymbol{s}, \boldsymbol{a})$ in two directions. The categorical distributional RL like C51 [15] directly estimates the target distribution by N fixed atoms that are uniformly distributed in the interval $[V_{min}, V_{max}]$, the approximated Q function determines the corresponding probability of each atom. Quantile distribution RL was proposed for superior flexibility. QRDQN [16] learns the target distribution by a set of quantiles with fixed probabilities and variable locations:

$$Z(\boldsymbol{s}, \boldsymbol{a}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\theta_i(\boldsymbol{s},\boldsymbol{a})}, \tag{12}$$

where $\delta_{\theta_i(\boldsymbol{s},\boldsymbol{a})}$ denotes a Dirac at the i-th output of an parameterized model $\theta_i(\boldsymbol{s}, \boldsymbol{a}) \in \mathbb{R}$.

Given sample $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, the next optimal action $\boldsymbol{a}^*$ and the distributional Bellman target were calculated following:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}'} \frac{1}{N} \sum_{j=1}^{N} \theta_i(\boldsymbol{s}', \boldsymbol{a}'), \tag{13}$$

$$\mathcal{T}\theta_j = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \theta_i(\boldsymbol{s}', \boldsymbol{a}^*), \quad j = 1, ..., N. \tag{14}$$

The parameterized model was updated by stochastic gradient descent approaches on the following loss based on Quantile Regression and Huber Loss [16], [27]:

$$\mathcal{L}_\kappa^H(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = \sum_{i=1}^{N} \mathbb{E}_j \left[\rho_{\hat{\tau}_i}^\kappa \left(\mathcal{T}\theta_j - \theta_i\left(\boldsymbol{s}, \boldsymbol{a}\right)\right)\right], \tag{15}$$

$$\rho_{\hat{\tau}}^\kappa(u) = |\tau - \delta_{\{u<0\}}| \mathcal{L}_\kappa(u), \tag{16}$$

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{for } |u| \le \kappa \\ \kappa\left(|u| - \frac{1}{2}\kappa\right), & \text{otherwise} \end{cases} \tag{17}$$

where the quantile midpoint is defined as $\hat{\tau}_i = \frac{\tau_{i-1}+\tau_i}{2}$ for $1 \le i \le N$, κ is the Huber-loss parameter.

III. Approach

A. Motivation of KDAC

In this section, we detailed the proposed KDAC which naturally leveraged distributional RL and KL divergence regularized RL within an AC framework to simultaneously reduce the overestimation bias and accumulated approximation error. The principle of our method was demonstrated in Fig. 1. Compared to mainstream approaches like TD3 [13], SAC [12] and TADD [14] that alleviate the overestimation bias by employing multiple critic networks, KDAC reduces overlarge values by adaptively clipping the high-value quantiles of one critic from a distributional perspective inspired by DQDQN [16]. The KL divergence regularization in DPP [18] is further integrated into the AC framework of KDAC to not only alleviate the overestimation of the max operator in a single critic by the Boltzmann softmax operator as reported by [28]–[30], but also stabilize the actor's policy update from an adaptively clipped distributional value function. Tackling the issue of overestimation, KDAC benefits in not only superior sample efficiency since only one critic will be learned but also a lightweight network structure with fewer computational burdens.

B. Reduce Overestimation Bias by Clipped Quantile

Define the actor network with parameters as $\pi(\boldsymbol{s}; \boldsymbol{\phi})$, the critic network with parameters as $\theta(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\psi})$, the corresponding target networks as $\pi(\boldsymbol{s}; \boldsymbol{\phi}^-)$ and $\theta(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\psi}^-)$. The distributional value function and its Bellman operator become:

$$Z(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\psi}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\theta_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})}, \tag{18}$$

$$\mathcal{T}Z(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\psi}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \mathbb{E}\left[Z\left(\boldsymbol{s}', \pi(\boldsymbol{s}'; \boldsymbol{\phi}); \boldsymbol{\psi}\right)\right]. \tag{19}$$

Dislike the traditional approach TD3 [13] that reduces the overestimation by selecting the smaller value from two critics:

$$Q\left(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}\right) \longleftarrow r(\boldsymbol{s},\boldsymbol{a}) + \gamma \min_{i=1,2} Q_i\left(\boldsymbol{s}', \pi\left(\boldsymbol{s}';\boldsymbol{\phi}\right);\boldsymbol{\psi}\right). \quad (20)$$

KDAC alleviates the overlarge distributional value function by clipping the top K large quantiles following:

$$Z_{\text{clip}}\left(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}\right) = \frac{1}{N-K}\sum_{i=1}^{N-K} \delta_{\vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})} \quad (21)$$

where $\vec{\theta}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})$ indicates the critic with sorted N outputs from small to large values. It is obvious that the overestimation of value distribution would be alleviated since:

$$Q_{\text{clip}}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}) = \mathbb{E}[Z_{\text{clip}}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})] \leq \mathbb{E}[Z(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})] = Q(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}). \quad (22)$$

The critic outputs high values only when most quantiles are located in large values. In practice, we propose an adaptive clipping strategy to dynamically change K over the learning process following:

$$K = \left\lfloor \max\left(K_{\text{start}} - \frac{K_{\text{start}} - K_{\text{end}}}{T \cdot d_{\text{decay}}} \cdot t, K_{\text{end}}\right) \cdot N \right\rfloor \quad (23)$$

where t and T are the steps and total steps of learning. It controls K to smoothly decrease from $\lfloor K_{\text{start}} \cdot N \rfloor$ to $\lfloor K_{\text{end}} \cdot N \rfloor$ within $T \cdot d_{\text{decay}}$ steps.

In summary, the clipped quantiles of KDAC dynamically suppress the overestimation in the distributional value function by preventing K network outputs with high values from updating and making decisions. It effectively alleviates the impact of overestimation on the exploratory and learning behaviors in RL without introducing additional network structure and computational burden.

C. Alleviate Accumulated Error under AC Framework

KDPP integrates the KL divergence regularization to further reduce the overestimation in the single critic and stabilize the policy update with the clipped value distribution. Let Z_{clip} present the action preferences function is a distribution form:

$$\Psi(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}\left[Z_{\text{clip}}\left(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}\right)\right] = \mathbb{E}\left[\frac{1}{N-K}\sum_{i=1}^{N-K} \delta_{\vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})}\right]. \quad (24)$$

For discrete action space, the Bellman target with quantiles $i = 1, ..., N-K$ can be obtained following Eq. (9) with Boltzmann softmax operator:

$$\mathcal{T}\vec{\theta}_i = \vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}^-) - \mathcal{B}_\eta\vec{\theta}_i(\boldsymbol{s}) + r(\boldsymbol{s},\boldsymbol{a}) + \gamma\mathcal{B}_\eta\vec{\theta}_i(\boldsymbol{s}'). \quad (25)$$

However, Eq. (25) is intractable in the AC framework with continuous actions due to the softmax operations over the entire action space.

KDAC tackles this issue by a Monte Carlo sampling on a local area over the continuous action space. For a given state-action pair $(\boldsymbol{s},\boldsymbol{a})$, an Monte Carlo action set with n actions $\hat{\mathcal{A}} = [\boldsymbol{a}^1, ..., \boldsymbol{a}^n]$ is sampled with a Gaussian noise:

$$\boldsymbol{a}^j = \boldsymbol{a} + \epsilon^j, \epsilon^j \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2), \text{ for } j = 1, ..., n \quad (26)$$

Algorithm 1: KL Divergence regularized Distributional Actor-Critic (KDAC)

Initialize the parameters of actor and critic networks and their target networks $\boldsymbol{\phi}, \boldsymbol{\phi}^-, \boldsymbol{\psi}, \boldsymbol{\psi}^-$
Prepare learning rates for critic and actor $\lambda_\theta, \lambda_\pi$, and smooth update parameter ξ
Initialize memory reply buffer $\mathcal{D}$
for $t = 1$ *to* T **do**
  # Interaction phase
  In state $\boldsymbol{s}_t$, decide action $\boldsymbol{a}_t = \pi(\boldsymbol{s}_t;\boldsymbol{\phi}^-)$
  Execute action $\boldsymbol{a}_t$, obtain reward $r(\boldsymbol{s}_t,\boldsymbol{a}_t)$ and observe the next state $\boldsymbol{s}_{t+1}$
  Store $(\boldsymbol{s}_t,\boldsymbol{a}_t,r(\boldsymbol{s}_t,\boldsymbol{a}_t),\boldsymbol{s}_{t+1})$ to sample buffer $\mathcal{D}$
  # Update phase
  Random sample a mini batch of M tuples from $\mathcal{D}$
  for *each sample* $(\boldsymbol{s},\boldsymbol{a},r(\boldsymbol{s},\boldsymbol{a}),\boldsymbol{s}')$ *in* $\mathcal{D}$ **do**
    $\boldsymbol{a}' \leftarrow \pi(\boldsymbol{s};\boldsymbol{\phi}^-)$
    Monte Carlo sampling $\hat{\mathcal{A}}, \hat{\mathcal{A}}'$ following Eq. (26)
    # Update critic network
    $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \lambda_\theta\nabla_{\boldsymbol{\psi}}J_\theta(\boldsymbol{\psi})$ following Eq. (28)
    # Update actor network
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \lambda_\pi\nabla_{\boldsymbol{\phi}}J_\pi(\boldsymbol{\phi})$ following Eq. (29)
    # Update target networks
    $\boldsymbol{\psi}^- \leftarrow \xi\boldsymbol{\psi} + (1-\xi)\boldsymbol{\psi}^-$
    $\boldsymbol{\phi}^- \leftarrow \xi\boldsymbol{\phi} + (1-\xi)\boldsymbol{\phi}^-$
  Dynamically calculate K following Eq. (23)
return $\theta(\cdot,\cdot;\boldsymbol{\psi}), \pi(\cdot;\boldsymbol{\phi})$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are selected for specific tasks. Sampling $\hat{\mathcal{A}}$ near action $\boldsymbol{a}$, $\mathcal{B}_\eta\vec{\theta}_i(\boldsymbol{s})$ in Eq. (25) is approximated as:

$$\mathcal{B}_\eta\vec{\theta}_i(\boldsymbol{s}) \approx \sum_{\boldsymbol{a}\in\hat{\mathcal{A}}} \frac{\exp\left(\eta\vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}^-)\right)\vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi}^-)}{\sum_{\bar{\boldsymbol{a}}\in\hat{\mathcal{A}}}\exp\left(\eta\vec{\theta}_i(\boldsymbol{s},\bar{\boldsymbol{a}};\boldsymbol{\psi}^-)\right)}. \quad (27)$$

$\mathcal{B}_\eta\vec{\theta}_i(\boldsymbol{s}')$ is calculated in the same way where the action set $\hat{\mathcal{A}}'$ is sampled near the optimal action at the next step decided by the target actor network $\boldsymbol{a}' = \pi(\boldsymbol{s};\boldsymbol{\phi}^-)$.

Given a sample $(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$, the critic network is updated following the Huber-loss function below:

$$J_\theta(\boldsymbol{\psi}) = \frac{1}{(N-K)^2}\sum_{i=1}^{N-K}\sum_{j=1}^{N-K}\rho_{\hat{\tau}_i}^{\kappa}\left(\mathcal{T}\vec{\theta}_j - \vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})\right). \quad (28)$$

The actor network is then updated based on the critic:

$$J_\pi(\boldsymbol{\phi}) = \nabla_{\boldsymbol{a}}\frac{1}{N-K}\sum_{i=1}^{N-K}\vec{\theta}_i(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\psi})\nabla_{\boldsymbol{\phi}}\pi(\boldsymbol{s};\boldsymbol{\phi})|_{\boldsymbol{a}=\pi(\boldsymbol{s};\boldsymbol{\phi})}. \quad (29)$$

D. Summary of Algorithm

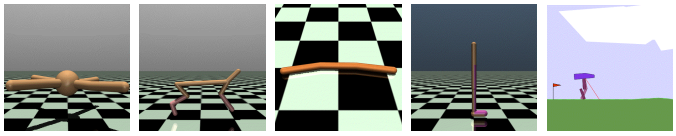
The learning process of the proposed KDAC is summarized in Algorithm 1. The critic and action networks are initialized with random parameters $\boldsymbol{\psi}, \boldsymbol{\phi}$. Those weights are copied to the target networks as $\boldsymbol{\psi}^-, \boldsymbol{\phi}^-$ to stabilize the learning procedure. Since Gaussian noise has been introduced to the Monte Carlo sampling to estimate the Q function with the continuous action

TABLE II
COMMON HYPERPARAMETER SETTINGS OF COMPARED APPROACHES

	KDAC	**DDPG**	**TD3**	**SAC**	**TQC**	**SDPG**
Critic Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Actor Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Actor and Critic Structure	(256,256)	(256,256)	(256,256)	(256,256)	(256,256)	(256,256)
Target Update Rate (ξ)	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}
Batch Size (M)	256	256	256	256	256	256
Discount Factor (γ)	0.99	0.99	0.99	0.99	0.99	0.99
Memory Size	10^6	10^6	10^6	10^6	10^6	10^6
Warmup Steps	10^4	10^4	10^4	10^4	10^4	10^4
Exploration Noise	/	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$	/	/	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$
Atom Number (N)	200	/	/	/	200	200

TABLE III
SPECIFIC HYPERPARAMETERS OF KDAC

	η	K_{start}	K_{end}	d_{decay}	ϵ
Ant	1.0	0.1	0.01	0.4	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$
HalfCheetah	0.5	0.1	0.01	0.4	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$
Swimmer	0.5	0.1	0.01	0.4	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$
Walker2d	1.0	0.2	0.01	0.1	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$
BipedalWalker	0.5	0.1	0.01	0.1	$\mathcal{N}(\mathbf{0}, \mathbf{0.1})$



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

Fig. 2. Benchmark tasks for evaluation: (a) Ant-v3 (111-dimensional observation including contact forces, 8-dimensional action), (b) HalfCheetah-v3 (17-dimensional observation, 6-dimensional action), (c) Swimmer-v3 (8-dimensional observation, 2-dimensional action), (d) Walker2d-v3 (17-dimensional observation, 6-dimensional action), (e) BipedalWalker-v3 (24-dimensional observation, 4-dimensional action)

space, no additional exploration noise is required for decision-making in KDAC. At each step t, KDAC first conducts the interaction phase. It decides an optimal action $\boldsymbol{a}_t$ according to its target actor network $\pi(\boldsymbol{s}_t; \boldsymbol{\phi}^-)$. After operating this action, the agent obtains the reward and observes the next step state. The transition tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1})$ is stored to the memory reply buffer $\mathcal{D}$. KDAC then moves to the training phase when the buffer size is larger than the mini-batch size M. It randomly selects M samples from $\mathcal{D}$. For each selected tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1})$ from the mini-batch, KDAC calculates the potential action for the next step state $\boldsymbol{a}' = \pi(\boldsymbol{s}; \boldsymbol{\phi}^-)$ and then collects the actions set $\hat{\mathcal{A}}, \hat{\mathcal{A}}'$ using Monte Carlo sampling following Eq. (26) to estimate Eq. (25) in the continuous action space. The parameters of critic and actor networks are updated by gradient descent optimization following Eqs. (28) and (29) with learning rates $\lambda_\theta, \lambda_\pi$. The target networks are smoothly updated with a smooth parameter ξ. After the update phase, KDAC dynamically changes the number of clipped quantiles K following Eq. (23) and moves to the next step.

IV. EXPERIMENTS

A. Experimental Settings

In this section, KDAC was evaluated by five benchmark control tasks in OpenAI Gym [31] and MuJoCo [32] with different difficulty and complexity: Ant-v3, HalfCheetah-v3, Swimmer-v3, Walker2d-v3, and BipedalWalker-v3 as demonstrated in Fig. 2. For the compared baselines, DDPG [7], TD3 [13] and SAC [12] were selected as the widely implemented related methods. As the state-of-the-art distributional RL baselines, we further selected TQC [17] that combined SAC and distributional RL and employed five critics to alleviate the issue of overestimation and Sample-based Distributional Policy Gradient (SDPG) [33] which enhanced the performances of DDPG from a distributional perspective. KDAC was developed by PyTorch [34]. All experiments in this section were conducted on a computational server with Intel Xeon-W2265 CPU, NVIDIA GeForce RTX-3090 GPU, 32 GB memory and Ubuntu 20.04 OS. The experimental results were summarized through five independent trials with different random seeds for statistical evidence.

Table II listed the common hyperparameters of all compared approaches. All actors and critics shared the same neural network structure with $(256, 256)$ hidden neurons and were updated with learning rate $3 \cdot 10^{-4}$. The batch size and memory buffer size were set to 256 and 10^6, the target update rate $\xi = 0.005$, and the discount factor $\gamma = 0.99$. The warmup process was conducted with 10^4 random samples for all approaches. The exploration noise of TD3, SAC and SDPG was set as $\mathcal{N}(\mathbf{0}, \mathbf{0.1})$. For all distributional RL approaches, the number of atoms was set as 200. The tunable hyperparameters of KDAC for each benchmark task were summarized in Table III. The KL regularization term was controlled by η, the dynamically changing K of the clipped quantile in Eq. (23) was determined by K_{start}, K_{end}, and d_{decay}. The Gaussian noise added to Monte Carlo action set $\hat{\mathcal{A}}$ in Eq. (26) was set as $\mathcal{N}(\mathbf{0}, \mathbf{0.1})$. The number of Monte Carlo sampling for all tasks was set as $n = 50$.

B. Evaluation of Learning Capability

This section first evaluated the learning capability of the proposed KDAC. The learning curves of KDAC and other baselines over five benchmark tasks were compared in Fig. 3.

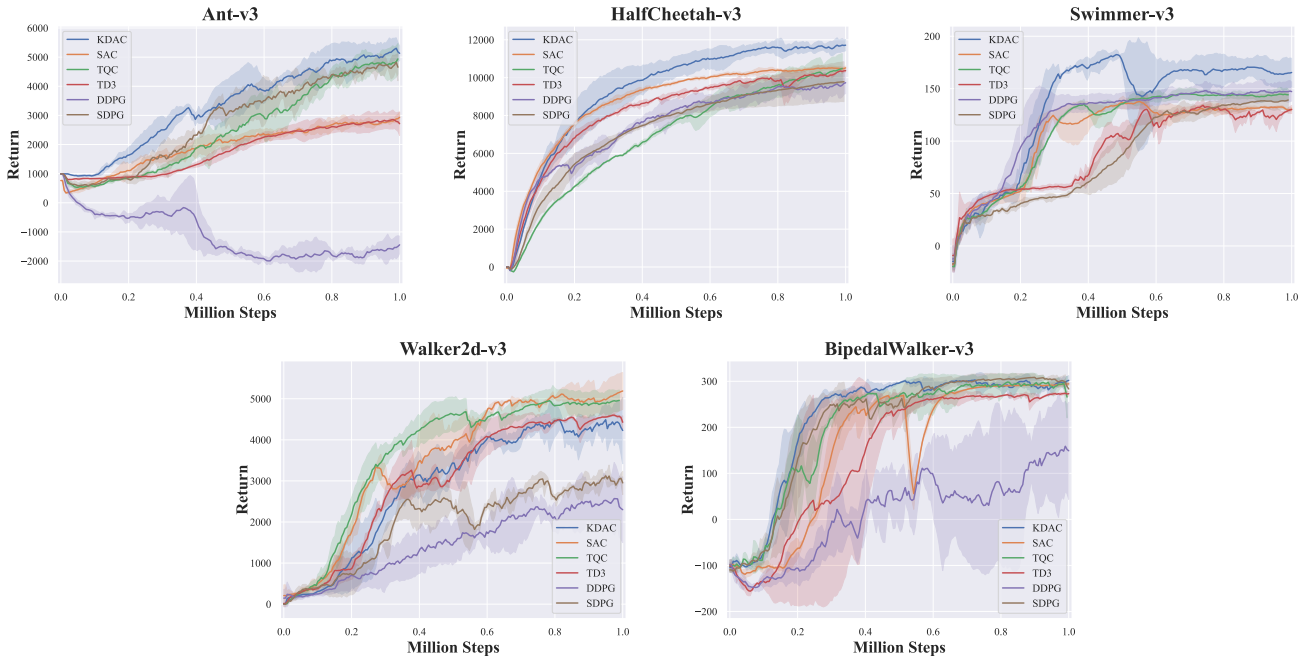


Fig. 3. Learning curves over five benchmark tasks. Curves are uniformly smoothed for visual clarity. The shaded region represents the corresponding standard deviation.

TABLE IV
MAXIMUM AVERAGE RETURNS OVER FIVE BENCHMARK TASKS

	KDAC	DDPG	TD3	SAC	TQC	SDPG
Ant	**5539.21 ± 272.88**	190.24 ± 311.72	3006.27 ± 363.93	3191.01 ± 321.84	5444.66 ± 354.15	5309.97 ± 46.00
HalfCheetah	**12139.58 ± 582.62**	10249.92 ± 310.13	10672.05 ± 47.75	10700.08 ± 265.19	10649.40 ± 311.72	9860.26 ± 1069.59
Swimmer	**190.050 ± 4.03**	152.69 ± 19.08	143.68 ± 26.54	149.67 ± 4.75	149.53 ± 14.17	142.61 ± 9.61
Walker2d	4824.71 ± 624.19	3231.36 ± 1217.11	4721.97 ± 25.46	**5393.00 ± 324.31**	5099.49 ± 341.76	3747.07 ± 236.38
BipedalWalker	**317.58 ± 1.34**	245.16 ± 35.97	279.14 ± 1.62	296.64 ± 5.57	312.4 ± 0.74	311.30 ± 5.78

Table IV demonstrated the maximum average returns (after the warmup process) of each compared method, the numbers in red color indicated the best performances in each task.

In the Ant task, traditional methods like SAC and TDC were unable to learn an effective control policy, while DDPG even produced a very poor control strategy. As a comparison, KDAC significantly outperformed the latest distributional RL methods TQC and SDPG in both convergence speed and final average return. It achieved the highest maximum average return which is 4.32% higher than SDPG and 1.74% higher than TQC. KDAC outperformed all baselines in the HalfCheetah task. Its maximum average return was 23.11% higher than SDPG and 13.99% higher than TQC. Although TQC had a spike in average return at the end of the learning process, it had the worst convergence performance. SDPG only achieved a close performance to DDPG, which was inferior to that of all other approaches. In the Swimmer task, the proposed KDAC outperforms all baselines in terms of convergence speed and final return. It quickly converged to an average return that surpasses the second-highest value of TQC within 0.3 million steps and finally enjoyed 33.27% increase in the maximum average return than TQC. In the Walker2d task, deterministic policy-based RL approaches such as DDPG tend to perform poorly. However, KDAC outperformed SDPG and DDPG due to its combination of KL divergence regularization and distributional value function. Although it achieved a close control performance to TD3, it was unable to surpass TQC and SAC. In terms of the maximum average return, KDAC outperforms SDPG by 28.76%, but lags behind TQC by 5.4%. In the BipedalWalker task, we observed a result similar to that of the Ant, HalfCheetah, and Swimmer tasks. KDAC quickly converged to the highest average returns over all baselines. The experimental results presented above demonstrated that KDAC has an overall advantage in terms of convergence speed and maximum average return compared with other baselines.

C. Evaluation of Reducing Overestimation

This subsection evaluated the impact of KDAC in reducing overestimation. Fig. 4 compared the estimated and corresponding true values over one million steps of training in Ant and Walker2d tasks for all compared approaches. In the Ant task, KDAC successfully alleviated the estimation error of the value function throughout the learning process. However, DDPG which does not address the issue of overestimation, struggled to accurately approximate the value function in the early stages of training and failed to explore states with high long-

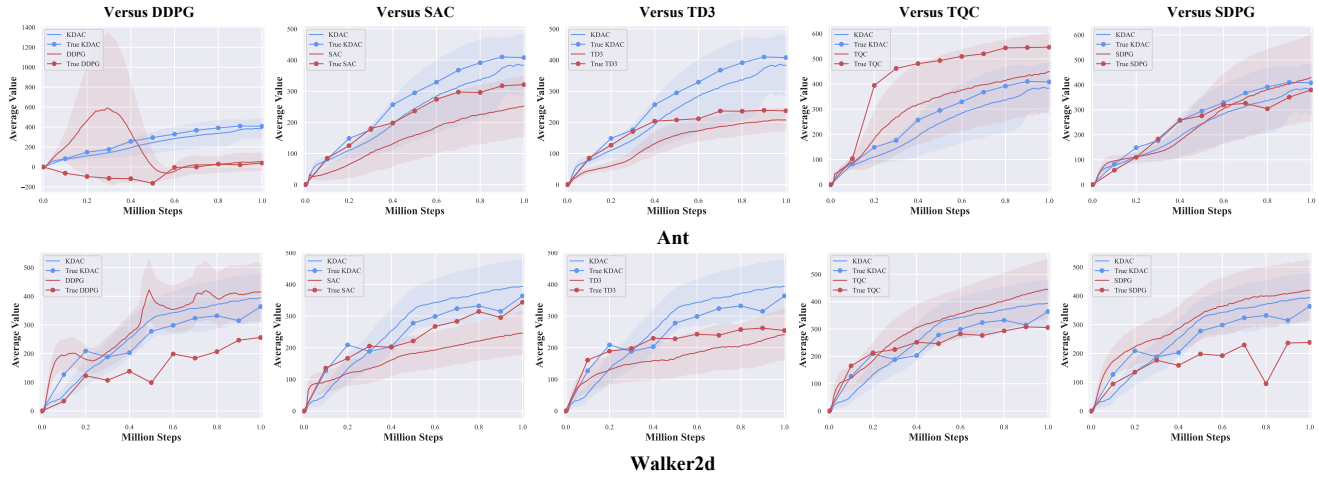


Fig. 4. Average estimation of the value function and the true value of KDAC and baselines in Ant and Walker2d tasks over one million steps.

term returns. SAC, TD3, and TQC, which employ additional critics to filter larger estimated values, successfully reduced overestimation but suffered from larger estimation errors than KDAC. These approaches heavily underestimated the value function, negatively impacting the learning procedure. While SDPG achieved a close estimation error to KDAC at the beginning of learning, it eventually overestimated the value function with a larger standard deviation.

In the Walker2d task, KDAC could not outperform SAC, TQC, and TD3. However, it successfully constrained the range of estimation error to an acceptable level that was fully covered by the standard deviation. In comparison, the double Q learning-based approaches such as SAC and TD3 underestimated the value function while maintaining similar estimation errors to KDAC. TQC demonstrated fewer estimation errors than KDAC in the first 0.2 million steps but quickly turned to larger overestimation errors. Compared to DDPG and SDPG which employed only one critic, KDAC showed a clear advantage in alleviating the overestimation of the value function.

The experimental results clearly indicate the advantages of KDAC in reducing the overestimation of the value function. By employing only one critic, the proposed method successfully outperforms state-of-the-art baselines with fewer estimation errors in tasks with superiority. Even in tasks where KDAC's learning performance was lagging behind, it still achieved a close estimation accuracy than the baselines employing multiple critic networks.

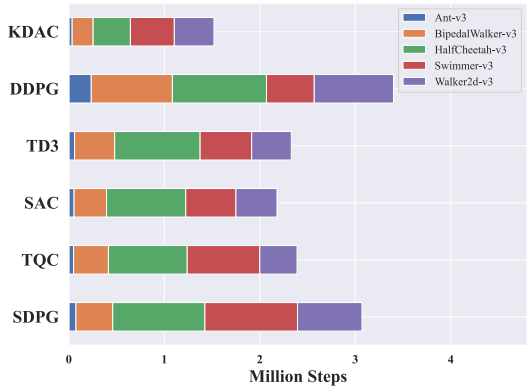


Fig. 5. Number of interactions used by compared approaches to finish different benchmark tasks.

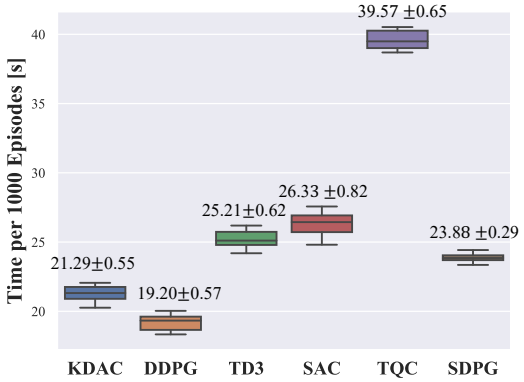


Fig. 6. Average calculation time over 1000 episodes in the Ant task.

D. Evaluation of Sample Efficiency

Defining the sample efficiency which is crucial to the implementation of RL in real-world hardware as the number of interactions used by each approach to reach the lower boundary of the maximum average returns in Table IV over five benchmark tasks, we evaluated KDAC's sample efficiency in Fig. 5. Overall, the proposed method significantly outperformed all traditional baselines across five benchmark tasks, with improvements of 55.29%, 34.76%, and 30.28% compared to DDPG, TD3, and SAC respectively. Compared with the state-of-the-art distributional RL approaches, KDAC comprehensively outperformed SDPG in all benchmark tasks with 50.49% fewer number of interaction. Although KDAC had 6.41% more iterations on the Walker2d task, it still had an overall advantage with 36.4% reduced interaction. This result demonstrated the superior sample efficiency of the proposed

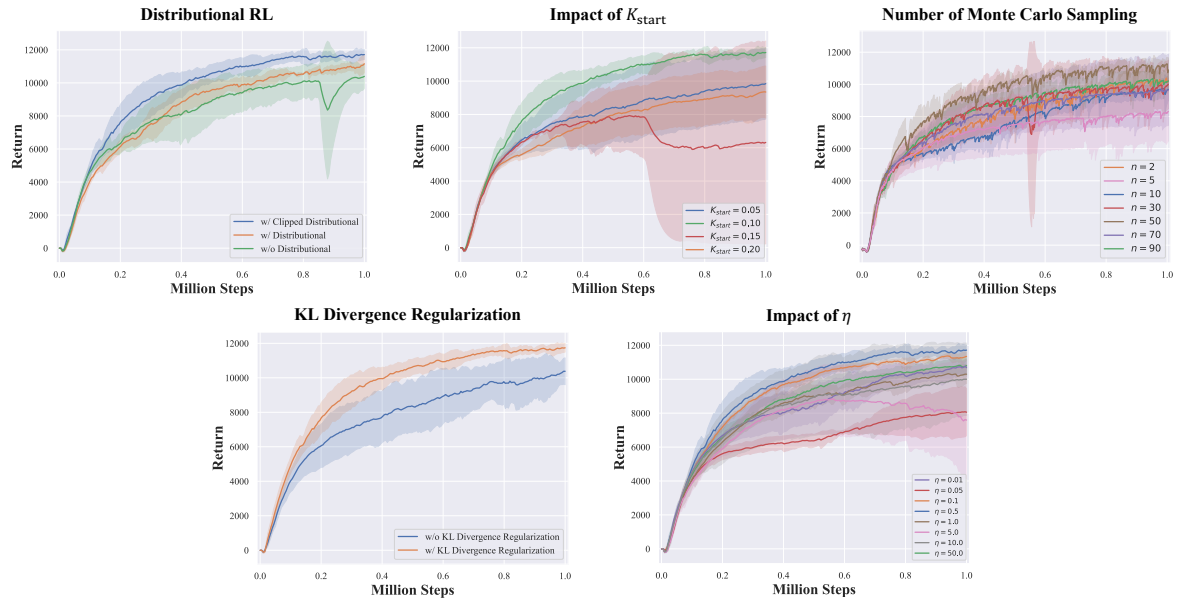


Fig. 7. Ablation test of KDAC in the HalfCheetah task in terms of the distributional RL, KL divergence regularization and impacts of related hyperparameters.

method and highlighted its potential in real-world applications with expensive sampling costs like robot control.

E. Evaluation of Computational Efficiency

In this subsection, we evaluated the computational efficiency of the proposed method. According to the average calculation time over 1000 episodes of all compared approaches in the Ant task illustrated in Fig. 6, KDAC required 22.29 seconds to process 1000 episodes which were slightly slower than DDPG's 19.20 seconds but outperformed SDPG with 10.85% fewer calculation time. Compared with SAC and TD3 which employ one additional critic network in the double Q learning framework, KDAC reduced the average calculation time by 19.14% and 15.55%, respectively. In terms of the most computationally heavy approach TQC which employed five critic networks to alleviate the issue of overestimation, the proposed method achieved a significant superiority in computational efficiency with 46.2% fewer calculation times. Inheriting the efficient network structure from DDPG, KDAC enjoyed obvious advantages over other baselines in computational efficiency. It suffered an additional computational burden from the softmax operator and Monte Carlo sampling which is far less than employing additional critics following SAC, TD3 and TQC.

F. Ablation Studies

In this subsection, we analyzed the impact of distributional RL, KL divergence regularization, and the related hyperparameters in KDAC through the HalfCheetah task in Fig. 7 as an ablation test. The top-left subfigure demonstrated the improvement in learning capability achieved by employing the distributional value function and the proposed strategy of clipping quantiles. In terms of the average returns of the last 5000 steps, the proposed method with a distributional value function outperformed the one without a distributional value function with 6.29% improvement. The integration of clipped quantile in the distributional value function further resulted in 12.12% higher average returns while enjoying a faster convergence. This result indicated the effectiveness of both the distributional value function and the clipped quantile in KDAC.

The top-middle subfigure demonstrated the impact of K_{start} in Eq. (23) to the learning performances. It was clearly observed that K_{start} significantly affected the learning behavior of KDAC in the HalfCheetah task. This result demonstrated the importance of selecting appropriate K_{start} for superior control performance in specific tasks. The impact of different numbers of Monte Carlo sampling was studied in the top-right subfigure. With a limited sample number like $n = 2$, KDAC could also learn to a close average return compared to SAC, TQC and TD3. However, the control capability significantly deteriorated when $n = 5$, which was even worse than the one of DDPG. Although the larger number of Monte Carlo sampling usually contributed to superior converge velocity and average return, the best result came from $n = 50$ which was generally set for all benchmarks.

The learning performances with and without the KL divergence regularization were compared in the bottom-left subfigure. It is clearly observed the KL divergence regularization contributed to not only faster convergence but also higher average returns with lower standard deviation in KDAC. Without the smooth update constrained by the KL divergence regularization, KDAC could not outperform other baselines with multiple critic networks by only one critic in obtaining long-term rewards and reducing overestimation. The bottom-right subfigure summarized the impact of parameter η that controls the KL divergence regularization. A small η turned into a strong penalty to the overlarge policy update which may slow down the convergence. Meanwhile, a large η resulted in a

less effective regularization. With an increasing η from 0.01 to 0.5, the learning performances of KDAC continued to improve except for the impaired control behaviors when $\eta = 0.05$. The best learning performance came from $\eta = 0.5$ which can be treated as a good balance between the convergence velocity and the smooth policy update. With the further increase in η, the control performance of KDAC continued to deteriorate and finally achieved a close control performance to KDAC without KL divergence regularization when $\eta = 50$.

V. Conclusions

This paper proposes a novel distributional RL approach KDAC, which naturally leverages distributional RL and KL divergence regularized RL under the AC framework to reduce overestimation of the value function in the RL domain. Compared to related baselines that employ multiple critic networks to alleviate overestimation, KDAC efficiently clips the overlarge estimated value function from a distributional perspective using a single critic. Additionally, KDAC incorporates KL divergence regularization into the learning of both actor and critic networks through a softmax operator based on Monte Carlo sampling, which contributes to superior sample efficiency in practice. Evaluated on five benchmark control tasks with varying levels of complexity, KDAC demonstrates superiority in learning capability, overestimation reduction, sample efficiency, and calculation efficiency compared to various baselines including traditional and state-of-the-art RL approaches. These results suggest KDAC's potential in efficiently learning various control tasks with fewer interactions while significantly reducing overestimation of the value function as an emerging practical RL approaches.

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*. MIT Press, 2018.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[6] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning.," in *International Conference on Learning Representations (ICLR)*, 2016.

[8] Y. Cheng, L. Huang, C. L. P. Chen, and X. Wang, "Robust actor-critic with relative entropy regulating actor," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2022.

[9] G. Peng, C. L. P. Chen, and C. Yang, "Neural networks enhanced optimal admittance control of robot–environment interaction using reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4551–4561, 2022.

[10] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. de Freitas, "Reinforcement learning for mobile robotics exploration: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[11] M. Chen and G. He, "Efficient and stable off-policy training via behavior-aware evolutionary learning," in *Conference on Robot Learning*, pp. 482–491, PMLR, 2023.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (ICML)*, 2018.

[14] D. Wu, X. Dong, J. Shen, and S. C. H. Hoi, "Reducing estimation bias via triplet-average deep deterministic policy gradient," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4933–4945, 2020.

[15] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning*, pp. 449–458, PMLR, 2017.

[16] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[17] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *International Conference on Machine Learning*, pp. 5556–5566, PMLR, 2020.

[18] M. G. Azar, V. Gómez, and H. J. Kappen, "Dynamic policy programming," *The Journal of Machine Learning Research (JMLR)*, vol. 13, no. 1, pp. 3207–3245, 2012.

[19] T. Kozuno, E. Uchibe, and K. Doya, "Theoretical analysis of efficiency and robustness of softmax and gap-increasing operators in reinforcement learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[20] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[21] Y. Cui, T. Matsubara, and K. Sugimoto, "Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states," *Neural Networks*, vol. 94, pp. 13–23, 2017.

[22] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robotics and Autonomous Systems*, vol. 112, pp. 72–83, 2019.

[23] L. Zhu, Y. Cui, G. Takami, H. Kanokogi, and T. Matsubara, "Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process," *Control Engineering Practice*, vol. 97, p. 104331, 2020.

[24] "Path integrals and symmetry breaking for optimal control theory," *Journal of statistical mechanics: theory and experiment*, vol. 2005, no. 11, p. P11011, 2005.

[25] E. Todorov, "Linearly-solvable markov decision problems," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

[26] M. Rowland, R. Dadashi, S. Kumar, R. Munos, M. G. Bellemare, and W. Dabney, "Statistics and samples in distributional reinforcement learning," in *International Conference on Machine Learning*, pp. 5528–5536, PMLR, 2019.

[27] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518, 1992.

[28] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," *Advances in neural information processing systems*, vol. 30, 2017.

[29] Z. Song, R. Parr, and L. Carin, "Revisiting the softmax bellman operator: New benefits and new perspective," in *International conference on machine learning*, pp. 5916–5925, PMLR, 2019.

[30] L. Pan, Q. Cai, and L. Huang, "Softmax deep double deterministic policy gradients," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11767–11777, 2020.

[31] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[32] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[33] R. Singh, K. Lee, and Y. Chen, "Sample-based distributional policy gradient," in *Learning for Dynamics and Control Conference*, pp. 676–688, PMLR, 2022.

[34] A. Paszke and et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems (NIPS)*, pp. 8024–8035, 2019.