11-2020

Reducing estimation bias via triplet-average deep deterministic policy gradient

Dongming WU
Beijing Institute of Technology

Xingping DONG
Beijing Institute of Technology

Jianbing SHEN
Beijing Institute of Technology

Steven C. H. HOI
Singapore Management University, chhoi@smu.edu.sg

Reducing Estimation Bias via Triplet-Average Deep Deterministic Policy Gradient

Article *in* IEEE Transactions on Neural Networks and Learning Systems · January 2020

DOI: 10.1109/TNNLS.2019.2959129

4 authors, including:

Xingping Dong
Inception Institute of Artificial Intelligence (IIAI)
25 PUBLICATIONS **1,023** CITATIONS

SEE PROFILE

Jianbing Shen
Beijing Institute of Technology
258 PUBLICATIONS **7,558** CITATIONS

SEE PROFILE

Steven C. H. Hoi
Nanyang Technological University
262 PUBLICATIONS **8,651** CITATIONS

SEE PROFILE

Reducing Estimation Bias via Triplet-Average Deep Deterministic Policy Gradient

Dongming Wu, Xingping Dong, Jianbing Shen, *Senior Member, IEEE*, and Steven C. H. Hoi, *Fellow, IEEE*

***Abstract*—The overestimation caused by function approximation is a well-known property in Q-learning algorithms, especially in single critic models, which leads to poor performance in practical tasks. However, the opposite property, underestimation, which often occurs in Q-learning methods with double critics, has been largely left untouched. In this paper, we investigate the underestimation phenomenon in the recent Twin Delay Deep Deterministic actor-critic algorithm and theoretically demonstrate its existence. We also observe that this underestimation bias does indeed hurt performance in various experiments. Taking into account the opposite properties of single critic and double critic methods, we propose a novel Triplet Average Deep Deterministic policy gradient algorithm that takes the weighted action-value of three target critics to reduce the estimation bias. Given the connection between estimation bias and approximation error, we suggest averaging previous target values to reduce per-update error and further improve performance. Extensive empirical results over various continuous control tasks in OpenAI gym show that our approach outperforms the state-of-the-art methods. Our source code is available at *https://github.com/shenjianbing/TADDRL*.**

***Index Terms*—Deep reinforcement learning, estimation bias, triplet networks, averaging technology.**

I. Introduction

In recent years, it has witnessed the huge success of deep reinforcement learning (DRL) in a variety of real-world tasks, such as playing the game of Go [40], [42], after the pioneering work Deep Q Network (DQN) [31] successfully combined the sensory ability of deep neural network function approximation with decision ability of Q-learning. However, there still exist several issues blocking its extension to more tasks with systematic overestimation bias being one well-known problem in value-based reinforcement learning, such as Q-learning [45].

The overestimation phenomenon occurs when the value estimated by a function approximation is larger than the actual value and is observed in Q-learning with a discrete action setting by Thrun *et al.* [45]. The main causes are attributed to the max operator and insufficiently flexible function approximation [45] or noise [20], [49], or more generally inaccurate action values [48]. Since a current action-value is updated using an imprecise estimate of a subsequent state, the overestimation bias is further exaggerated by the nature of temporal difference learning [44], further resulting in the policy becoming suboptimal or even divergence.

Considering the connection between overestimation bias and variance or error at each update, numerous approaches tried to directly minimize the size of errors at each time step by averaging value estimates [2], [3], adding penalizing or correcting terms to the policy [14], [28] and applying smoothed value functions [32]. On the other hand, it is noted that the overestimation bias often occurs in single critic methods, such as DQN [31] and DDPG [29]. Thus, some approaches have tried to use two critics to reduce the overestimation bias for robust performance in both discrete action [20], [48] and continuous control settings [16]. For instance, Double DQN [48] makes unbiased value estimates by decoupling the selection and adoption of the best action using two independent estimators, which has been successfully applied to discrete actions. However, this approach is not effective in avoiding overestimation in continuous domains, such as actor-critic algorithms [16]. To overcome overestimation in the actor-critic setting, the recent Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [16] applied a pair of critic functions for value estimation and took the minimum between these two estimates for target updating. This minimization operation is effective in alleviating the overestimation phenomenon in continuous control settings, but it may cause an underestimation bias at each updating iteration. Although this bias will not be explicitly propagated during updating [16], the inaccurate estimation still negatively affects the performance, which is shown in our experiments. Fujimoto *et al.* afterward introduce a simple solution, Batch-Constrained deep Q-learning (BCQ) [15], for limiting underestimation phenomenon in TD3.

Few previous works pay attention to the underestimation phenomenon occurring in some value-based reinforcement learning approaches, like TD3. In this paper, we focus on the impact of the underestimation phenomenon on the performance. We begin by establishing the property of underestimation bias for the recent actor-critic algorithm, TD3, in a continuous control setting. Theoretically, we demonstrate that TD3 suffers from the underestimation bias even when its two critic functions give an unbiased estimation for the true critic, which is caused by a function approximation error and the minimization operation. We also observe that the underestimation phenomenon occurs and indeed hurts the performance in practical tasks.

As mentioned before, the overestimation phenomenon often

This work was supported in part by the Beijing Natural Science Foundation under Grant 4182056. Specialized Fund for Joint Building Program of Beijing Municipal Education Commission. (Corresponding author: *Jianbing Shen*)

D. Wu is with Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, P. R. China. (Email: wudongming97@gmail.com)

X. Dong, and J. Shen are with Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, P. R. China, and also with the Inception Institute of Artificial Intelligence, Abu Dhabi, UAE. (email: xingping.dong@gmail.com, shenjianbingcg@gmail.com)

S. C. H. Hoi is with the School of Information Systems, Singapore Management University, and Salesforce Research Asia, Singapore. (Email: stevenhoi@gmail.com)

occurs in single critic algorithms and double critics approaches may bring underestimation bias. Can we combine these two opposite biases to achieve a more accurate estimation? To answer this question, we explore a combination of these two estimation biases in actor-critic algorithms for the continuous setting. More specifically, we propose a novel Triplet Average Deep Deterministic policy gradient algorithm (TADD), which takes the weighted action-value of triplet critics and an average Q-value approach for robust target updating. In the framework of triplet critics, we apply twin critics and select the minimum estimation to simulate underestimation and add a single critic to provide the overestimation value. Then, we take the weighted average of these two estimations as the final critic value to update the policy. The triplet critics mechanism successfully reduces the estimation bias, including the overestimation in the single critic method and the underestimation in the twin critics approach, in both theoretical and experimental results. Besides, considering the connection between the estimation bias and noise, caused in particular by high variance, we add an average Q-value method into our single critic to address variance reduction for better value estimation.

The major contributions are summarized as follows:

- We theoretically prove that underestimation bias occurs during the minimization operation between two critics, and demonstrate its negative impact on the twin critics method TD3. To the best of our knowledge, this is the first work to analyze the underestimation phenomenon both theoretically and experimentally.
- A novel triplet critics mechanism is incorporated into the deep deterministic policy gradient algorithm to reduce the estimation bias for both theory and practical tasks.
- To further decrease the estimation bias, we propose an average Q-value method for the actor-critic method to do variance reduction and provide a theoretical analysis to prove its effectiveness.
- Through more accurate action-value estimation, our approach achieves better performance than the state-of-the-art methods on several control tasks from OpenAI gym.

II. Preliminaries

Deep reinforcement learning (DRL), the task of learning the optimal control policy with interactions with environment, had achieved huge development in many areas like playing games [40], [42], computer vision [19], [12], network architecture search [4], [10], [47] and imitation learning [23]. In this work, we formulate the standard reinforcement learning (RL) as a Markov decision process (MDP), defined by a tuple $M = (\mathcal{S}, \mathcal{A}, p, p_0, \mathcal{R}, \gamma)$ that consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition kernel p, an initial state distribution p_0, a reward function $\mathcal{R}$, and a discount factor $\gamma \in [0, 1]$. The goal of reinforcement learning is to find an optimal policy that maximize its expectation of a discounted cumulative reward $R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$, where $r_{t'} = \mathcal{R}(s_{t'}, a_{t'})$. In deterministic policy mapping states to a specific action, action-value function (or Q-function) $Q(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a]$ is used to evaluate the policy, further guiding to learn the optimal policy owning the biggest value, which means that precise estimate value plays an essential role.

When p is unknown, the Q-function can be recursively approximated using the following Bellman equation with a transition (s, a, r, s') (action a is executed at state s, leading to reward r and next state s'):

$$Q(s, a) = r + \gamma \mathbb{E}_{s' \sim S, a' \sim \pi}[Q(s', a')]. \tag{1}$$

However, function approximation, especially when combined with a neural network, i.e. deep reinforcement learning (DRL), is prone to causing estimation variance due to its generalization, and further estimation bias due to the value iteration. In the following, we provide some mathematical background for these algorithms related to our work: DQN, DDPG, and TD3.

A. Deep Q-Network (DQN)

DQN [31] uses a Deep Neural Network (DNN) as function approximation that for a given state s outputting the action-value of policy, $Q(s, a)$. For addressing the instability from the combination between nonlinear function like DNN and Q-learning, DQN introduces two important technologies: experience replay and target network. When the estimate of the optimal action-value, $Q^*(s, a)$, is learned by minimizing the following loss function with respect to the neural network parameters θ according to Bellman equation (1):

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{B}}[(y - Q(s, a; \theta))^2], \tag{2}$$

here $y^i_{s,a} = r + \gamma \max_{a' \in A} Q(s', a'; \theta')$ is target value constructed using the fixed and separate target network with parameters θ' which is copied from network parameters θ for every fixed time-steps to reduce the correlation between target value and online value, and $\mathcal{B}$ is a replay buffer to store and relay experience transitions, which eliminates the correlation of sampled trajectory data.

Although DQN can achieve human-level control in many real-world games like Atari, there still are many imperfect places in it. In order to achieve better performance, plenty of excellent approaches has been proposed to upgrade DQN, such as accurate value estimation [48], [51], recurrent neural network [21], highlight experience replay [36], advanced exploration strategy [6], [39], [46], [21], function regularization [17] and etc. However, these methods favor discrete action, and can't handle with continuous action, a situation largely existing in real-world tasks. Considering this concern, policy gradient methods like Asynchronous Advantage Actor-Critic (A3C) [30], Trust Region Policy Optimization (TRPO) [37], Proximal Policy Optimization (PPO) [38], Soft Actor-Critic (SAC) [18] appeared, and DDPG, an actor-critic method based on the value function and the deterministic policy, also has a major impact for continuous setting due to its simplicity and effectiveness.

B. Deep Deterministic Policy Gradient (DDPG)

Silver *et al.* [41] proposed a Deterministic Policy Gradient method (DPG), which uses deterministic policy instead of the usual stochastic policy in actor-critic algorithm with continuous actions. This simple form is more efficient for action-value estimation in high-dimensional action spaces. Lillicrap *et al.* [29] incorporated a DNN into DPG (DDPG), which uses a learned value estimator critic to train a deterministic policy

actor. The deterministic policy specifies one action with the guidance of the single critic. We denote the parameters of the actor π and critic Q as ϕ and θ, respectively. Similarly, the target actor π' and target critic Q' are denoted as ϕ' and θ'. The critic estimates Q-values and updates its parameters by minimizing the following loss, using collected experience in a replay buffer $\mathcal{B}$:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}[(y - Q(s,a \mid \theta))^2], \tag{3}$$

where $y = r + \gamma Q'(s', \pi'(s'))$ is the target value based on the independent target critic network. This loss function also stems from the Bellman equation (1). The actor is updated in the direction of the critic's action-value gradient by applying the chain rule to the expected return $J(\phi) = \int_{\mathcal{S}} \rho(s)\mathcal{R}(s, \pi_\phi(s))ds$, where $\rho(s)$ is state distribution [41]:

$$\nabla_\phi J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a \mid \theta) \mid_{s=s_i, a=\pi(s_i)} \nabla_\phi \pi(s \mid \phi) \mid_{s_i} . \tag{4}$$

After updating θ and ϕ by using Eq. (3) and (4), DDPG uses a soft target update:

$$\theta' = \tau\theta + (1-\tau)\theta', \quad \phi' = \tau\phi + (1-\tau)\phi', \tag{5}$$

where $\tau \ll 1$ greatly improves the learning stability due to its slow changing.

Following DDPG, additional extensions to the basic algorithm have further increased its performance. Several works have made use of distributed methods [34], [5], [13], [7] and prioritized experience replay [1], [24], [50] to deal with data efficiency. The recent works [35], [26], [25] proposed a combination with the evolutionary strategy for better performance.

C. Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 [16], an extension of DDPG, is also a deep deterministic actor-critic algorithm. TD3 takes a minimum value between a pair of critics, named clipped Double Q-learning, as a target value estimate. Through this method, if one critic is overestimated, the other will be chosen, thereby limiting overestimation phenomenon. This is different from DDPG taking the only critic-value directly. Similar to DDPG, TD3 uses the loss function (3) but $y = r + \gamma \min_{i=1,2} Q_i'(s', \pi'(s'))$, where Q_1' and Q_2' represent two target critics corresponding to a pair of independent critics Q_1 and Q_2. The single actor π is optimized with respect to Q_1 according to function (4). Moreover, TD3 uses delaying policy updates to reduce per-update error and target policy smoothing regularization by adding noise into the target policy to relieve over-fitting.

III. The Underestimation Phenomenon

In this section, we begin with a theoretical analysis of the underestimation phenomenon under the minimization operation between two critics. Then we show its negative effect in a recent actor-critic algorithm TD3, which also has the same minimization operation.

According to the Bellman equation (1) and loss function (3), the minimum between two critic values (or Q-values) over the next states is employed to update two critic functions (or Q-functions) over an experience transition (s, a, r, s') in the actor-critic method, like TD3:

$$Q(s,a) \longleftarrow r + \gamma \min_{i=1,2} Q_i(s', \pi'(s')). \tag{6}$$

To better understand the effect of minimization, we define Q_1^{approx} and Q_2^{approx} as two independent estimate Q-values of two critics, which approximate the hypothetical true value Q^{true}. Due to the noise induced by function approximation, there exists an error term $Y_{s'}^i = Q_i^{approx}(s', \pi'(s')) - Q^{true}(s', \pi'(s'))$ that overestimates Q-values for $i = 1, 2$, where the error can be modeled by an independent and identical uniform distribution in the interval $[-\epsilon, \epsilon]$. The minimization operation over two estimate Q-values further introduces some errors to the left-hand term of equation (6), denoted by Z_s:

$$\begin{aligned} Z_s &= r + \gamma \min_{i=1,2} Q_i^{approx}(s', \pi'(s')) - \big(r + \gamma Q^{true}(s', \pi'(s'))\big) \\ &= \gamma \min\left(Y_{s'}^1, Y_{s'}^2\right). \end{aligned} \tag{7}$$

To analyze the expected error $\mathbb{E}[Z_s]$, we first give Theorem 1 and its proof.

Theorem 1: Let Q^{true} denotes the only true value and assume there are M estimate values Q_i^{approx} to approximate it for $i = 1, \cdots, M$. If the error of each approximation value, denoted as $Y^i = Q_i^{approx} - Q^{true}$, is independently and identically uniformly distributed in the interval $[-\epsilon, \epsilon]$, the average underestimation of the minimum of all approximation values is $\mathbb{E}[\min_{i=1,\cdots,M}(Y^i)] = -\frac{M-1}{M+1}\varepsilon$.

Proof: Since there are M error variables Y^i identically uniformly distributed in $[-\varepsilon, \varepsilon]$ for $i = 1, \cdots, M$, we can denote their same probability density as $f(x)$:

$$f(x) = \begin{cases} \frac{1}{2\varepsilon} & x \in [-\varepsilon, \varepsilon] \\ 0 & x \in else. \end{cases}$$

Then, we can derive that for all variables Y^i:

$$P(Y^i > x) = \begin{cases} 1 & \text{if} \quad x \le -\varepsilon \\ \frac{\varepsilon - x}{2\varepsilon} & \text{if} \quad x \in (-\varepsilon, \varepsilon) \\ 0 & \text{if} \quad x \ge \varepsilon. \end{cases}$$

Because these estimation error variables Y^i are independent and identically distributed, the probability that $\min_{i=1,\cdots,M} Y^i \ge x$ is equal to the probability that $Y^i \ge x$ for all M variables simultaneously and we can get that:

$$\begin{aligned} P(\min_{i=1,\cdots,M} Y^i > x) &= \prod_{i=1,\cdots,M} P(Y^i > x) \\ &= \begin{cases} 1 & \text{if} \quad x \le -\varepsilon \\ \left(\frac{\varepsilon - x}{2\varepsilon}\right)^M & \text{if} \quad x \in (-\varepsilon, \varepsilon) \\ 0 & \text{if} \quad x \ge \varepsilon. \end{cases} \end{aligned}$$

The cumulative distribution function of $\min_{i=1,\cdots,M} Y^i$ is $P(\min_{i=1,\cdots,M} Y^i \le x)$:

$$P(\min_{i=1,\cdots,M} Y^i \le x) = \begin{cases} 0 & \text{if} \quad x \le -\varepsilon \\ 1 - \left(\frac{\varepsilon - x}{2\varepsilon}\right)^M & \text{if} \quad x \in (-\varepsilon, \varepsilon) \\ 1 & \text{if} \quad x \ge \varepsilon. \end{cases}$$

This implies that we can get the probability density of this variable by using the derivative of the cumulative distribution function: $f_{min}(x) = \frac{d}{dx} P(\min_{i=1,\cdots,M} Y^i \le x) =$

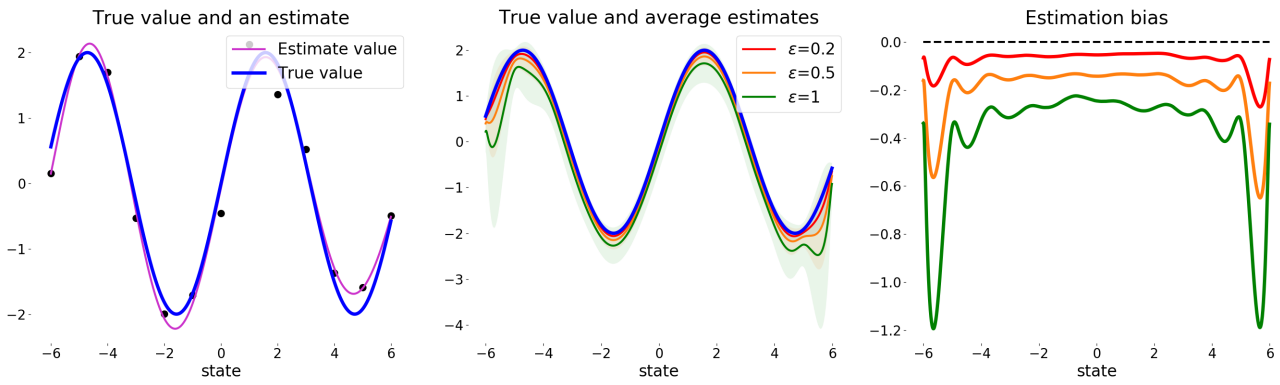


Fig. 1: Simulation of underestimation bias under minimization operation. The left column shows a true value (blue) and an estimate (purple) over some sampled noisy data (black dots). The middle column shows the average values of over one thousand estimations with different error-disturbance bounds ε. The shaded region represents a standard deviation of the average. The right shows the expected estimation bias, where the underestimation bias increases with rising error.

$\frac{M}{2\varepsilon}(\frac{\varepsilon-x}{2\varepsilon})^{M-1}$ for $x \in (-\varepsilon, \varepsilon)$. Now we write expectation as an integral in terms of the probability density:

$$\begin{aligned} E[\min_{i=1,\cdots,M} Y^i] &= \int_{-\varepsilon}^{\varepsilon} x f_{min}(x)dx \\ &= \int_{-\varepsilon}^{\varepsilon} x \frac{M}{2\varepsilon}(\frac{\varepsilon - x}{2\varepsilon})^{M-1}dx \\ &= -M\int_{1}^{0}(\varepsilon - 2\varepsilon y)y^{M-1}dy \text{ for } y = \frac{\varepsilon - x}{2\varepsilon} \\ &= -\frac{M-1}{M+1}\varepsilon. \end{aligned}$$

Thus, we prove theorem 1. ■

Considering Theorem 1 for $M = 2$, we can easily derive the expected error $\mathbb{E}[Z_s]$ due to the minimization operation between two critics, like TD3:

$$\mathbb{E}[Z_s] = -\frac{1}{3}\gamma\epsilon < 0. \tag{8}$$

Thus, even if the approximations are unbiased, the minimization operation can easily introduce a negative expected bias, i.e. $\mathbb{E}[Z_s] < 0$, at each update, which can be called underestimation phenomenon.

To sum up, the inaccurate function approximation and minimization operation between two critics result in underestimation bias. We consider a simple fitting process to simulate underestimation bias with the true values for a continuous state space, $Q^\pi(s, \pi^{approx}(s))$, with a fixed approximate actor $\pi^{approx}(s)$. For convenience, we denote $Q^\pi(s, \pi^{approx}(s)) = Q^\pi(s)$. The real values are shown in the left column of Figure 1, $Q^\pi(s) = 2\sin(x)$ for instance. Considering the impact of inaccurate function approximation for sampled trajectory, we directly sample 13 noisy data (black dots) from true values as alternatives, where the noise is drawn from an uniform distribution bounded in $[-\varepsilon, \varepsilon]$. Then, we fit them as estimate values by an 8-degree polynomial. We take a minimum of two fitting data at each evaluation and average one thousand fitting results as the final estimation result, as shown in the middle column. The results in the right column show that the bias is negative, indicating an underestimation happening, which means that the minimum of inaccurate values can cause underestimation. We also investigate the effect of different error-disturbance intervals by changing the value of ε.

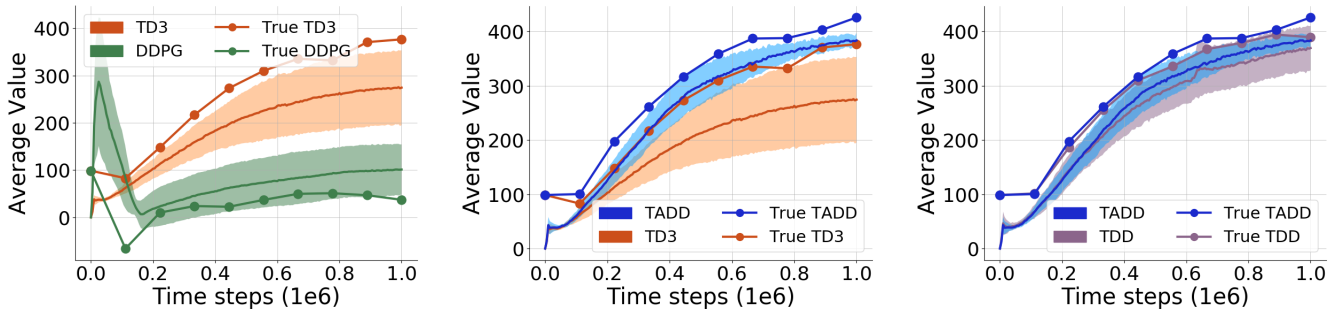
a) Does this theoretical underestimation occur in practice for TD3: The results on the OpenAI gym environments Ant-v2 [9] suggest that underestimation occurs in TD3. In Figure 2(a), we plot the average estimate value of TD3 without a target policy smoothing regularization operation, starting from 10 initial states and compare it to the true value. The true value is the discounted cumulative reward based on the current policy and states. The underestimation bias of TD3 is very clear, and our proposed TADD, described in the following section, obviously reduces the underestimation bias in practice (see Figure 2(b)). More importantly, the true values of TADD are higher than TD3, indicating that our method improves performance by mitigating underestimation [48]. In addition, we also show that the single critic method DDPG produces overestimation in Ant-v2, which means in the same environment, DDPG tends to suffer from overestimation, while underestimation occurs for TD3. Thus, we can combine these two opposite properties to design our new algorithm.

IV. Triplet Average Deep Deterministic Policy Gradient

Based on the above observation, we propose a framework of triplet critics, to reduce the estimation bias, and prove its effectiveness in both theory and practice. Then, based on the connection between estimation bias and high variance, we present an average Q-value method to further reduce bias.

A. Triplet critics

As a single critic algorithm, DDPG produces an overestimation bias, while, as mentioned above, TD3 brings an underestimation bias due to the minimization operation between double critics. Intuitively, we can combine these two opposite biases to achieve a more accurate estimation. Building on TD3 and DDPG, we form a novel algorithm, named Triplet Deep Deterministic policy gradient algorithm (TDD), which uses the



(a) TD3 vs DDPG (b) TADD vs TD3 (c) TADD vs TDD

Fig. 2: Illustration of estimation bias for TD3, DDPG, TADD, and TDD, on environment Ant-v2 over one million time steps. The marked lines are true values and the non-marked are estimate values. The shaded region represents a standard deviation of the averaged evaluation over 10 trials.

weighted minimum Q-value between a pair of critics and the Q-value of a regular critic to obtain the target update:

$$y = r+\gamma\{\beta \min_{i=1,2} Q'_i(s', \pi_{\phi'}(s'))+(1-\beta)Q'_3(s', \pi_{\phi'}(s')\} \quad (9)$$

where $\beta \in (0,1)$ is the weight of double critics. During implementation, we use a single actor optimized with respect to Q_1 and set the same target value y for Q_1, Q_2 and Q_3. At each update, Q_i is assumed to be an unbiased estimation with bound ϵ for $i = 1, 2$, and the expected estimation bias of double critics is $\mathbb{E}[Z_s^{double}] = -\frac{1}{3}\gamma\epsilon$, as TD3. Note that a single critic method like DDPG always favors overestimation. Thus, we can assume Q_3 is overestimated with the same hypothesis bias of DDPG $\mathbb{E}[Z_s^{single}] = \mathbb{E}[Z_s^{DDPG}] = \lambda > 0$. Due to the weighting operation in our TDD, our expected estimation bias becomes $\mathbb{E}[Z_s^{TDD}] = -\frac{1}{3}\beta\gamma\epsilon + (1-\beta)\lambda$.

Based on above assumptions, we can find our estimation bias is larger than TD3 and less than DDPG:

$$\mathbb{E}[Z_s^{TD3}] < \mathbb{E}[Z_s^{TDD}] < \mathbb{E}[Z_s^{DDPG}], \quad (10)$$

which means that our method TDD effectively reduces the estimation bias compared with TD3 or DDPG. When $\beta = \frac{3\lambda}{\gamma\epsilon+3\lambda}$, $\mathbb{E}[Z_s^{TDD}] = 0$. This indicates our approach can make unbiased estimation with some reasonable assumptions and a good choice of weight. The experiments in the next section show that the larger weight β can complete it in most cases.

In the following, to confirm the convergence of our proposed method, TDD, we also provide a proof of convergence in the finite MDP setting, as Theorem 2. For accelerating our proof, we first give the known certified theory Lemma 1, and after that, show the Theorem 2 and its proof. First of all, referring to the clipped Double Q-learning of TD3, we denote three critics of TDD as weighted Triplet Q-learning. In a version of our weighted Triplet Q-learning for a finite MDP setting, we maintain three tabular value estimates Q^A, Q^B, Q^C. At each time step, we update the value estimates with respect to the target value y and learning rate $\alpha_t(s,a)$:

$$Q^A(s,a) = Q^A(s,a) + \alpha_t(s,a)(y - Q^A(s,a))$$
$$Q^B(s,a) = Q^B(s,a) + \alpha_t(s,a)(y - Q^B(s,a))$$
$$Q^C(s,a) = Q^C(s,a) + \alpha_t(s,a)(y - Q^C(s,a)).$$

The target value y is set by selecting the optimal actions a^* based on the next states s' according to loss function (9):

$$a^* = \arg\max_a Q^A(s', a),$$
$$y = r+\gamma\left[\beta \min(Q^A(s', a^*), Q^B(s', a^*))+(1-\beta)Q^C(s', a^*)\right].$$

In this proof, we borrow heavily from the proofs of convergence of SARSA [43], Double Q-learning [20] and TD3 [16], which are actually in the same line. The proof of Lemma 1 builds on the work of Bertsekas [8] and was proposed by Singh *et al.* [43].

Lemma 1: Consider a stochastic process (ζ_t, Δ_t, F_t), $t \geq 0$, where $\zeta_t, \Delta_t, F_t : X \longrightarrow \mathbb{R}$ satisfy the equation:

$$\Delta_{t+1}(x_t) = (1 - \zeta_t(x_t))\Delta_t(x_t) + \zeta_t(x_t)F_t(x_t),$$

where $x_t \in X$ and $t = 0, 1, 2, \cdots$. Let P_t be a sequence of increasing $\sigma-$fields such that ζ_0 and Δ_0 are P_0-measurable and ζ_t, Δ_t and F_{t-1} are P_t-measurable, $t = 1, 2, 3, \cdots$. Assume that the following hold:

1) The set X is finite.
2) $\zeta_t(x_t) \in [0,1]$, $\sum_t \zeta_t(x_t) = \infty$, $\sum_t(\zeta_t(x_t))^2 < \infty$ with probability 1 and $\forall x \neq x_t : \zeta(x) = 0$.
3) $||\mathbb{E}[F_t|P_t]|| < \kappa||\Delta_t|| + c_t$, where $\kappa \in [0,1)$ and c_t converges to 0 with probability 1.
4) $\mathrm{Var}[F_t(x_t)|P_t] < K(1 + \kappa||\Delta_t||^2)$, where K is some constant.

where $||\cdot||$ denotes the maximum norm. Then Δ_t converges to 0 with probability 1.

Theorem 2: Given the following conditions:

1) Each state action pair is sampled an infinite number of times.
2) The MDP is finite.
3) $\gamma \in [0,1)$.
4) Q-values are stored in a lookup table.
5) Q^A, Q^B and Q^C receive an infinite number of updates.
6) The learning rates $\alpha_t(s,a) \in [0,1]$, $\sum_t a_t(s,a) = \infty, \sum_t(a_t(s,a))^2 < \infty$ with probability 1 and $\alpha_t(s,a) = 0, \forall(s,a) \neq (s_t, a_t)$.
7) $\mathrm{Var}[r(s,a)] < \infty, \forall s, a$.

Then, the weighted Triplet Q-learning will converge to the optimal value function Q^*, as defined by the Bellman optimality equation, with probability 1.

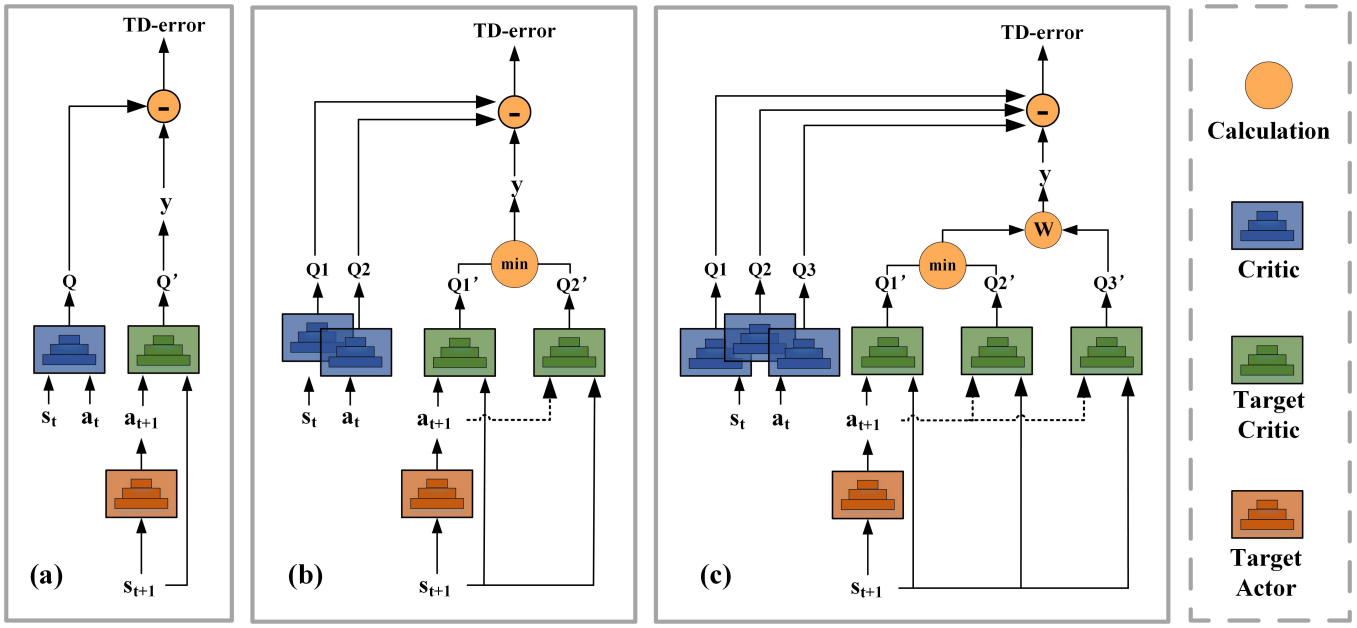


Fig. 3: Visualization of forward propagation process for (a) DDPG, (b) TD3 and (c) TDD. The target critics (green) are separate and used to calculate the Q-value of the next states and actions for substantial target value. The target value y is calculated with different operators (yellow and circled), where 'min' means a minimum of two values, 'W' means weighting operation and '-' means subtraction.

Proof: We apply Lemma 1 with $P_t = \{Q_0^A, Q_0^B, Q_0^C, s_0, a_0, \alpha_0, r_1, s_1, \cdot, s_t, a_t\}, X = S \times A, \Delta_t = Q_t^A - Q^*, \zeta_t = \alpha_t$. Notice that, the condition 1, 2, 4 of Lemma 1 hold with respect to the conditions 2,6,7 of Theorem 2. Next, we will find out whether the condition 3 of Lemma 1 holds.

Defining $a^* = \arg\max_a Q_t^A(s_{t+1}, a)$, we have that:

$$\begin{aligned}
&\Delta_{t+1}(s_t, a_t) \\
&= (1 - \alpha_t(s_t, a_t))(Q_t^A(s_t, a_t) - Q^*(s_t, a_t)) \\
&\quad + \alpha_t(s_t, a_t)\left\{r_t + \gamma\left[\beta \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*))\right.\right. \\
&\qquad\qquad \left.\left. + \ (1-\beta) Q_t^C(s_{t+1}, a^*)\right] - Q^*(s_t, a_t)\right\} \\
&= (1 - \alpha_t(s_t, a_t))\Delta_t(s_t, a_t) + \alpha_t(s_t, a_t) F_t(s_t, a_t),
\end{aligned}$$

where $F_t(s_t, a_t)$ is defined as:

$$\begin{aligned}
F_t(s_t, a_t) &= r_t + \gamma\left[\beta \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*))\right. \\
&\qquad \left. + (1-\beta) Q_t^C(s_{t+1}, a^*)\right] - Q^*(s_t, a_t) \\
&= r_t + \gamma\left[\beta \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*))\right. \\
&\qquad \left. + \ (1-\beta) Q_t^C(s_{t+1}, a^*)\right] - Q^*(s_t, a_t) \\
&\qquad + \gamma Q_t^A(s_{t+1}, a^*) - \gamma Q_t^A(s_{t+1}, a^*) \\
&= F_t^Q(s_t, a_t) + c_t,
\end{aligned}$$

where $F_t^Q(s_t, a_t) = r_t + \gamma Q_t^A(s_{t+1}, a^*) - Q^*(s_t, a_t)$ is the value of F_t under normal Q-learning and $c_t = \gamma\left[\beta \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) + (1-\beta) Q_t^C(s_{t+1}, a^*)\right] - \gamma Q_t^A(s_{t+1}, a^*)$. It is well-known that $\mathbb{E}[F_t^Q | P_t] \leq \gamma ||\Delta_t||$, so, to apply condition 3 of the Lemma 1, we need identify the convergence of c_t.

We rewrite c_t as:

$$\begin{aligned}
c_t &= \gamma\left[\beta \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*))\right. \\
&\qquad \left. + (1-\beta) Q_t^C(s_{t+1}, a^*)\right] - \gamma Q_t^A(s_{t+1}, a^*) \\
&= \underbrace{\gamma\beta\left[\min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) - Q_t^A(s_{t+1}, a^*)\right]}_{c_{t1}} \\
&\qquad + \underbrace{\gamma(1-\beta)\left[Q_t^C(s_{t+1}, a^*) - Q_t^A(s_{t+1}, a^*)\right]}_{c_{t2}} \\
&= c_{t1} + c_{t2}.
\end{aligned}$$

According to the analysis of TD3, we have the convergence of c_{t1}, and we need to further identify the convergence of c_{t2}. Let $y = \beta \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) + (1-\beta) Q_t^C(s_{t+1}, a^*)$ and $\Delta_t^{CA}(s_t, a_t) = Q_t^C(s_t, a_t) - Q_t^A(s_t, a_t)$:

$$\begin{aligned}
&\Delta_{t+1}^{CA}(s_t, a_t) \\
&= \Delta_t^{CA}(s_t, a_t) + \alpha_t(s_t, a_t)(y - Q_t^C(s_t, a_t) - (y - Q_t^A(s_t, a_t))) \\
&= \Delta_t^{CA}(s_t, a_t) + \alpha_t(s_t, a_t)(Q_t^A(s_t, a_t) - Q_t^C(s_t, a_t)) \\
&= (1 - \alpha_t(s_t, a_t))\Delta_t^{CA}(s_t, a_t).
\end{aligned}$$

It is very clear that $\Delta_{t+1}^{CA}(s_t, a_t)$ will converge to 0, further implying that c_{t2} and c_t will converge to 0 with probability 1, when condition 3 of Lemma 1 holds. Therefore, we can get that $Q^A(s_t, a_t)$ converges to $Q^*(s_t, a_t)$. Using similar methods, we can get the convergence of both $Q^B(s_t, a_t)$ and $Q^C(s_t, a_t)$. Thus, we prove Theorem 2. ■

To better understand the differences between the three algorithms, we visualize several forward propagation processes for DDPG, TD3 and TDD in Figure 3. S_t, a_t and s_{t+1} are the data sampled from a replay buffer, and used to calculate the Q-value and Temporal Difference error (TD-error). For concision, we don't display the process of back propagation.

B. Variance Reduction by Average Q-value

Although the double critics method induces underestimation bias, it actually decreases the estimation noise, especially that caused by high variance, at each update. However, the high variance of the single critic in the simple combination causes a significant impact on our entire estimation bias during value iteration, as shown in Figure 2(c). To reduce the per-update variance for further controlling the magnitude of the bias, we propose an average Q-value method for the single critic. During practical learning, target approximation error is denoted as estimation noise Z_t at time step t, determined after minimizing the loss function (3) for an individual update:

$$Z_t = Q(s,a;\theta_t) - y_t = Q(s,a;\theta_t) - \left(r + \gamma Q(s',a';\theta'_t)\right), \tag{11}$$

where $\theta'_t = \tau^t\theta_0 + \sum_{j=1}^{t}\tau^{t-j}(1-\tau)\theta_j$ is the weighted internal parameter of previous networks according to the soft target update in equation (5). The target approximation error mainly comes from two sources: the sub-optimum of θ_i, resulting from the local minimum of deep neural networks, and the generality error from not obtaining infinite state-action pairs when sampling. Next, for simplicity, we assume the Q-value over the next states as:

$$Q(s',a';\theta'_t) = \tau^t Q(s',a';\theta_0) + \sum_{j=1}^{t}\tau^{t-j}Q(s',a';\theta_j) + D_t, \tag{12}$$

where we directly use weighted Q-value of networks with previous parameter to replace the Q-value of a network with weighted internal parameter, and the deviation between them is denoted as D_t. Employing functions (11) and (12), we obtain the Q-value, Q^{single}, when the agent begins at state s_0, and terminates at state s_N:

$$\begin{aligned}
&Q^{single}(s_0,a;\theta_t) = Z_t + r + \gamma Q(s_1,a;\theta'_t)\\
&= Z_t + r + \gamma\Bigg[\tau^t Q(s_1,a;\theta_0)\\
&\qquad + \sum_{j_1=1}^{t}\tau^{t-j_1}(1-\tau)Q(s_1,a;\theta_{j_1}) + D_t\Bigg]\\
&= \underbrace{\underbrace{Z_t + r}_{U_0} + \underbrace{\gamma\tau^t Q(s_1,a;\theta_0)}_{V_0} + \underbrace{\gamma D_t}_{W_0}}_{A_0}\\
&\qquad + \gamma\sum_{j_1=1}^{t}\tau^{t-j_1}(1-\tau)Q(s_1,a;\theta_{j_1})\\
&= A_0 + \gamma\sum_{j_1=1}^{t}\tau^{t-j_1}(1-\tau)\Bigg[Z^{j_1} + r + \gamma\Bigg(\tau^{j_1}Q(s_2,a;\theta_0)\\
&\qquad + \sum_{j_2=1}^{j_1}\tau^{j_1-j_2}(1-\tau)Q(s_2,a;\theta_{j_2}) + D_{j_1}\Bigg)\Bigg]\\
&= A_0 + \underbrace{\gamma\sum_{j_1=1}^{t}\tau^{t-j_1}(1-\tau)(Z_{j_1}+r)}_{U_1}\\
&\qquad + \underbrace{\gamma^2\sum_{j_1=1}^{t}\tau^{t}(1-\tau)Q(s_2,a;\theta_0)}_{V_1}\\
&\qquad + \underbrace{\gamma^2\sum_{j_1=1}^{t}\tau^{t-j_1}(1-\tau)D_{j_1}}_{W_1}\\
&\qquad + \gamma^2\sum_{j_1=1}^{t}\sum_{j_2=1}^{j_1}\tau^{t-j_2}(1-\tau)^2Q(s_2,a;\theta_{j_2})\\
&= A_0 + A_1 + \underbrace{\gamma^2\sum_{j_1=1}^{t}\sum_{j_2=1}^{j_1}\tau^{t-j_2}(1-\tau)^2(Z_{j_2}+r)}_{U_2}\\
&\qquad + \underbrace{\gamma^3\sum_{j_1=1}^{t}\sum_{j_2=1}^{j_1}\tau^{t}(1-\tau)^2Q(s_3,a;\theta_0)}_{V_2}\\
&\qquad + \underbrace{\gamma^3\sum_{j_1=1}^{t}\sum_{j_2=1}^{j_1}\tau^{t-j_2}(1-\tau)^2D_{j_2}}_{W_2}\\
&\qquad + \gamma^3\sum_{j_1=1}^{t}\sum_{j_2=1}^{j_1}\sum_{j_3=1}^{j_2}\tau^{t-j_3}(1-\tau)^3Q(s_3,a;\theta_{j_3})\\
&= \sum_{n=0}^{N}A_n = \sum_{n=0}^{N}\left(U_n + V_n + W_n\right),
\end{aligned} \tag{13}$$

where

$$\begin{aligned}
U_n &= \gamma^n\left(\sum\right)\tau^{t-j_n}(1-\tau)^n(Z_{j_n}+r),\\
V_n &= \gamma^{n+1}\left(\sum\right)\tau^{t}(1-\tau)^n(Q(s_{n+1},a;\theta_0)),\\
W_n &= \gamma^{n+1}\left(\sum\right)\tau^{t-j_n}(1-\tau)^n D_{j_n}.
\end{aligned}$$

If $n > 0$:

$$\left(\sum\right) = \sum_{j_1=1}^{t}\sum_{j_2=1}^{j_1}\cdots\sum_{j_n=1}^{j_{n-1}},$$

else $n = 0$:

$$\left(\sum\right) = 1,\ j_0 = 0.$$

We assume that Z_t is a zero-mean random variable with variance $Var[Z_t] = \delta^2$ and $E[r] = 0, Var[r] = 0$ for simplicity. Thus, we can easily obtain the variance of a single

Q-value under soft target update:

$$Var[Q^{single}(s_0,a;\theta_t)] = \sum_{n=0}^{N}\left\{\gamma^{2n}\underline{\left(\sum\right)\tau^{2(t-j_n)}}(1-\tau)^{2n}\delta^2+\underline{Var[W_n]}\right\}.$$

Our new method, named average Q-value, averages K previous target Q-values:

$$Q(s',a';\theta'_t)=\frac{1}{K}\sum_{k=1}^{K}Q(s',a';\theta'_{t+1-k}). \tag{14}$$

We use the average Q-value based on the previous target critic networks in the above equation instead of the original Q-function in single critic method. In the following, we can prove the effectiveness of average Q-value in variance reduction. Similar to the above analysis and employing the above three functions (11), (14), (12) in order, we can calculate the new Q-value, $Q^{average}$, agent starting at s_0 and terminating at s_N as before, based on the average Q-value:

$$\begin{aligned}
&Q^{average}(s_0,a;\theta_t)\\
&=Z_t+r+\gamma Q(s_1,a;\theta'_t)\\
&=Z_t+r+\gamma\frac{1}{K}\sum_{k=1}^{K}Q(s_1,a;\theta'_{t+1-k_1})\\
&=Z_t+r+\frac{\gamma}{K}\sum_{k_1=1}^{K}\Bigg[\tau^{t+1-k_1}Q(s_1,a;\theta_0)\\
&\qquad+\sum_{j_1=1}^{t+1-j_1}\tau^{t+1-k_1-j_1}(1-\tau)Q(s_1,a;\theta_{j_1})+D_t\Bigg]\\
&=\underbrace{\underbrace{Z_t+r}_{U'_0}+\underbrace{\frac{\gamma}{K}\sum_{k_1=1}^{K}\tau^{t+1-k_1}Q(s_1,a;\theta_0)}_{V'_0}+\underbrace{\frac{\gamma}{K}\sum_{k_1=1}^{K}D_t}_{W'_0}}_{A'_0}\\
&\qquad+\frac{\gamma}{K}\sum_{k_1=1}^{K}\sum_{j_1=1}^{t+1-k_1}\tau^{t+1-k_1-j_1}(1-\tau)Q(s_1,a;\theta_{j_1})\\
&=A'_0+\frac{\gamma}{K}\sum_{k_1=1}^{K}\\
&\quad\bullet\sum_{j_1=1}^{t+1-k_1}\tau^{t+1-k_1-j_1}(1-\tau)\left[Z_{j_1}+r+\gamma Q(s_2,a;\theta'_{j_1})\right]\\
&=A'_0+\underbrace{\frac{\gamma}{K}\sum_{k_1=1}^{K}\sum_{j_1=1}^{t+1-j_1}\tau^{t+1-k_1-j_1}(1-\tau)(Z_{j_1}+r)}_{U'_1}\\
&\quad+\underbrace{\frac{\gamma^2}{K^2}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\sum_{j_1=1}^{t+1-k_1}\tau^{t+2-k_1-k_2}(1-\tau)Q(s_2,a;\theta_0)}_{V'_1}\\
&\quad+\underbrace{\frac{\gamma^2}{K^2}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\sum_{j_1=1}^{t+1-k_1}\tau^{t+1-k_1-j_1}(1-\tau)D_{j_1}}_{W'_1}\\
&\quad+\frac{\gamma^2}{K^2}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\sum_{j_1=1}^{t+1-k_1}\\
&\qquad\bullet\sum_{j_2=1}^{j_1+1-k_2}\tau^{t+2-k_1-k_2-j_2}(1-\tau)^2Q(s_3,a;\theta_{y_3})\\
&=\sum_{n=0}^{N}A'_n=\sum_{n=0}^{N}\left(U'_n+V'_n+W'_n\right),
\end{aligned} \tag{15}$$

where

$$\begin{aligned}
U'_n&=\frac{\gamma^n}{K^n}\left[\sum\right]\tau^{t+n-j_n-k_1-\cdots-k_n}(1-\tau)^n(Z_{j_n}+r),\\
V'_n&=\frac{\gamma^{n+1}}{K^{n+1}}\left[\sum\right]\\
&\quad\bullet\sum_{k_{n+1}=1}^{K}\tau^{t+n+1-k_1-\cdots-k_{n+1}}(1-\tau)^nQ(s_{n+1},a;\theta_0),\\
W'_n&=\frac{\gamma^{n+1}}{K^{n+1}}\left[\sum\right]\sum_{k_{n+1}=1}^{K}\tau^{t+n-j_n-k_1-\cdots-k_n}(1-\tau)^nD_{j_n}.
\end{aligned}$$

If $n>0$:

$$\left[\sum\right]=\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\cdots\sum_{k_n=1}^{K}\sum_{j_1=1}^{t+1-k_1}\sum_{j_2=1}^{j_1+1-k_2}\cdots\sum_{j_n=1}^{j_{n-1}+1-k_n},$$

else $n=0$:

$$\left[\sum\right]=1,\ j_0=0.$$

We can also easily derive the variance of average Q-value:

$$Var[Q^{average}(s_0,a;\theta_t)] = \sum_{n=0}^{N}\left\{\gamma^{2n}\underline{B_{K,n}}(1-\tau)^{2n}\delta^2+\underline{Var[W'_n]}\right\},$$

where $B_{K,n}=\frac{1}{K^{2n}}\sum_{k_1=1}^{K}\cdots\sum_{k_n=1}^{K}\sum_{j_1=1}^{t+1-k_1}\sum_{j_2=1}^{j_1+1-k_2}\cdots\sum_{j_n=1}^{j_{n-1}+1-k_n}\tau^{2(t+n-j_n-k_1-\cdots-k_n)}$ if $n=1,2,\cdots,N$, else $B_{K,n}=\tau^{2t}$ when $n=0$.

We underline the difference between $Var[Q^{single}(s_0,a;\theta_t)]$ and $Var[Q^{average}(s_0,a;\theta_t)]$. For a more convenient comparison, we set the difference as $C_n=(\sum)\tau^{2(t-j_n)}$ in $Var[Q^{single}(s_0,a;\theta_t)]$. When $n=0,K=1$, it is obvious that $B_{K,n}=C_n=\tau^{2t}$. When $n=1,K>1$, we can calculate C_1 and $B_{K,1}$:

$$\begin{aligned}
C_1&=\sum_{j_1=1}^{t}\tau^{2(t-j_1)}=\frac{\tau^{2(t-1)}(1-\tau^{-2t})}{1-\tau^{-2}}\\
&=\frac{1}{1-\tau^{-2}}\left[\tau^{2(t-1)}-\tau^{-2}\right].
\end{aligned} \tag{16}$$

Algorithm 1 TADD

Input: Observation state $\mathcal{S}$
Output: Action $\mathcal{A}$
Randomly initialize critic networks Q_{θ_1}, Q_{θ_2}, Q_{θ_3} and actor networks π_ϕ with parameters $\theta_1, \theta_2, \theta_3, \phi$
Initialize target networks $\theta_1' \longleftarrow \theta_1, \theta_2' \longleftarrow \theta_2, \theta_3' \longleftarrow \theta_3, \phi' = \phi$
Initialize experience replay buffer $\mathcal{B}$
Initialize target critic network replay buffer $\mathcal{C}$ with K previous networks $[Q_{\theta_3'}]^1, \cdots, [Q_{\theta_3'}]^K$
for episode = 1, M **do**
  Receive initial observation state s_1
  for t = 1,T **do**
    Select action $a_t = \pi(s_t) + \epsilon$ according to the current policy and exploration noise $\epsilon \in \mathcal{N}(0, \sigma)$
    Execute action a_t and observe reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
    Sample a random minibatch of N transitions (s, a, r, s') from $\mathcal{B}$
    Set $a' = \pi_{\phi'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \widetilde{\sigma}), -c, c)$
    Set $Q_1' = \min_{i=1,2} Q_{\theta_i'}(s', a')$
    Set $Q_2' = \frac{1}{K} \sum_k [Q_{\theta_3'}(s', a')]^k$
    Set $y = r + \gamma [\beta Q_1' + (1-\beta) Q_2']$
    Update critics θ_i by minimizing the loss: $L = \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$
    if t mod d **then**
      Update the actor ϕ using deterministic policy gradient:

$$\bigtriangledown_\phi J(\phi) = \frac{1}{N} \sum \bigtriangledown_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \bigtriangledown_\phi \pi_\phi(s)$$

      Update target networks:

$$\theta_i' = \tau \theta_i + (1-\tau)\theta_i', \quad \phi' = \tau \phi + (1-\tau)\phi'$$

      Update $\mathcal{C}$ by popping up the oldest and filling the new target critic network
    end if
  end for
end for

$$\begin{aligned} B_{K,1} &= \frac{1}{K^2} \sum_{k_1=1}^{K} \sum_{j_1}^{t+1-k_1} \tau^{2(t+1-j_1-k_1)} \\ &= \frac{1}{K^2} \sum_{k_1=1}^{K} \frac{\tau^{2(t-k_1)} - \tau^{-2}}{1-\tau^{-2}} \\ &= \frac{1}{K^2(1-\tau^{-2})} \left[\frac{\tau^{2(t-1)}(1-\tau^{-2K})}{1-\tau^{-2}} - K\tau^{-2} \right] \\ &= \frac{1}{(1-\tau^{-2})} \left[\underbrace{\frac{(1-\tau^{-2K})}{(1-\tau^{-2})K^2}}_{>1} \tau^{2(t-1)} \underbrace{-\frac{1}{K}}_{>-1} \tau^{-2} \right] \\ &< \frac{1}{1-\tau^{-2}} \left[\tau^{2(t-1)} - \tau^{-2} \right] = C_n, \end{aligned} \tag{17}$$

where $1 - \tau^{-2}$ is negative.

Employing a mathematical induction method, we assume that $B_{K,n} < C_n$ for $n > 1$. Therefore, we can derive that:

$$C_{n+1} = \sum_{j_{n+1}=1}^{j_n} \tau^{2(j_n - j_{n+1})} C_n. \tag{18}$$

$$B_{K,n+1} = \frac{1}{K^2} \sum_{k_{n+1}=1}^{K} \sum_{j_{n+1}=1}^{j_n+1-k_{n+1}} \tau^{2(j_n+1-j_{n+1}-k_{n+1})} B_{K,n}. \tag{19}$$

The coefficients between (16) and (18), (17) and (19) are similar, and, more importantly, the last term of (19) is less than (18), i.e. $B_{K,n} < C_n$. Then, referring to the calculation of functions (16) and (17), we have $B_{K,n+1} < C_{n+1}$. Therefore, we can easily get $B_{K,n} < C_n$ based on the mathematical induction method for $n > 0, K > 1$. We assume that the error and variance of D_t is constant across updates. Due to the similarity between coefficients of both U_n and W_n, we can also derive that $Var[W_n'] < Var[W_n]$ by using the same method for $n > 0, K > 1$. Finally, we have

$$Var[Q^{average}(s_0, a; \theta_t)] < Var[Q^{single}(s_0, a; \theta_t)].$$

Thus, we prove that the method of average Q-value can reduce variance of single critic.

Intuitively, the variance reduction method is similar to Averaged-DQN [2], which also averages the previous target action-values. However, there are two main differences: Firstly, our method is applied to deep deterministic actor-critic algorithms rather than simple DQN. Secondly, the soft target update is analyzed in the process of iteration, which involves the neural network itself.

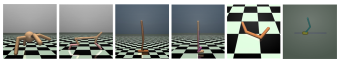
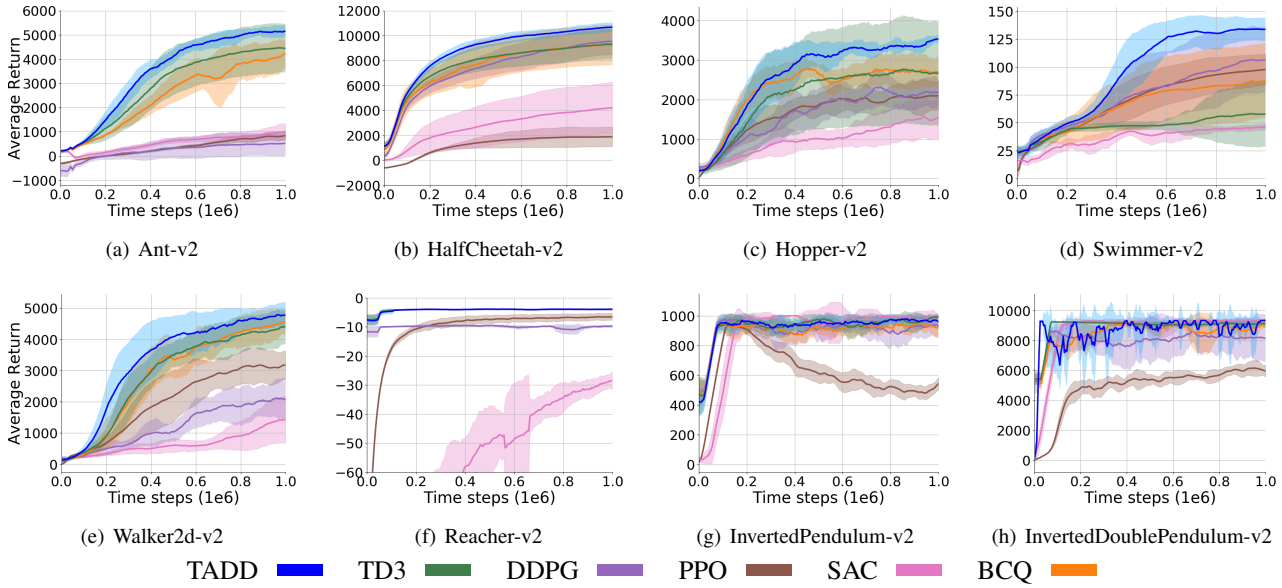


Fig. 4: Example MuJoCo environments. In order from the left: Ant-v2, HalfCheetah-v2, Hopper-v2, Walker2d-v2, Swimmer-v2, InvertedDoublePendulum-v2.

TABLE I: Quantitative descriptions of MuJoCo environments for version 2 (v2). The first row represents the state dimension and the second represents the action dimension, where both of them are continuous in each dimension.

Environments (v2)	Ant	HalfCheetah	Hopper	Swimmer	Walker2d	Reacher	InvertedPendulum	InvertedDoublePendulum
State Dimension	111	17	11	8	17	11	4	11
Action Dimension	8	6	3	2	6	2	1	1



(a) Ant-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Swimmer-v2

(e) Walker2d-v2 (f) Reacher-v2 (g) InvertedPendulum-v2 (h) InvertedDoublePendulum-v2

Fig. 5: Qualitative performances on the OpenAI gym continuous control tasks. The results are obtained by running our TADD, TD3, DDPG, PPO, and SAC over 10 trials. The darker lines show the median over the seeds. The shaded region represents a standard deviation of the averaged evaluation.

V. Experimental Results and Analysis

We present a Triplet-Average Deep Deterministic policy gradient algorithm (TADD), building on DDPG and TD3 by applying the modification described above for estimation bias reduction. Specifically, at each update, we calculate the average Q-value of a single critic and weight it with the minimum of the other two critics as target value. In addition, we make use of delaying policy update and target policy smoothing proposed by Fujimoto *et al.* [16] for increasing stability and performance of our algorithm. Here, we give more details of TADD and summary it in Algorithm 1.

A. Implementation Details

To evaluate our algorithm, we measure its performance on MuJoCo continuous tasks from OpenAI gym [9]. These tasks provide both a low dimensional state vector describing joint angles and position in Table I (including action vector), and a high dimensional pixel rendering and examples in Figure 4, where we only use the state vector as inputs. Moreover, we directly employ the default reward function and environments settings without any modification for fair comparison.

Due to the recent concern about reproducibility [22], [33] and effective comparison [11], we run our experiments over 10 different random seeds and the network initialization. For network architecture, TADD uses a two-layer fully connected neural network of 400 and 300 hidden neurons, with rectified linear units (ReLU) between each layer, for both actor and critic, and a final tanh unit following the output of the actor. Like TD3, both state and action are inputted into the first layer of the critic. To minimize the loss of algorithms, the Adam optimizer [27] is used to update both network parameters on a mini-batch of 100 uniformly sampled transitions with a learning rating of 10^{-3}. The actor and both target critic networks are updated every $d = 2$ iterations for delayed policy update, with $\tau = 0.005$.

For the trade-off of exploration and exploitation, we use the off-policy exploration policy, adding Gaussian noise $\mathcal{N}(0, 0.1)$ to each executed action. Additionally, we maintain exploring without training before 10000 time steps for better collecting experience. In target policy smoothing, the Gaussian noise $\mathcal{N}(0, 0.2)$ is added to each action from the target actor network, clipped to maximum action in terms of each task.

To better balance the overestimation and underestimation

TABLE II: Average return for last 5 evaluations over 10 different random seeds. The second column show the results of our method without the averaged operation (TDD). The last columns present the results obtained by running our TADD, TD3, DDPG, PPO, and SAC. Best results are in bold.

Environment	TADD	TDD	TD3	DDPG	PPO	SAC	BCQ
Ant-v2	**5178.36±387.72**	5057.01	4469.44	475.06	840.08	1003.61	4381.48
HalfCheetah-v2	**10696.31±445.12**	10191.83	9340.67	9538.28	1875.94	4242.29	9169.74
Hopper-v2	**3634.19±160.33**	3463.93	2770.74	2239.08	2099.52	1544.36	2725.24
Swimmer-v2	**133.47±11.89**	128.21	58.07	104.91	97.70	46.10	87.31
Walker2d-v2	**4622.27±819.72**	4511.20	4414.28	2111.87	3182.05	1420.07	4619.18
Reacher-v2	**-3.97±0.48**	-3.99	-3.99	-10.16	-6.53	-28.11	-3.98
InvertedPendulum-v2	**1000.00±0.00**	**1000.00**	990.33	959.72	541.52	**1000.00**	840.30
InvertedDoublePendulum-v2	**9337.41±15.02**	9224.01	9141.94	5872.40	5984.99	9313.54	9069.82

biases over different environments, we set $\beta = 0.2$ for Swimmer-v2 and $\beta = 0.95$ for the remaining environments. We average the K target action-values to reduce the approximation error for an individual update, with $K = 2$. While a larger K would result in a larger benefit with respect to the accumulation of error, a lower change of target value can improve the stability, especially for continuous action. For fair comparison, we set other hyperparameters as the same as TD3 [16].

B. Evaluation

We compare our algorithm against other state-of-the-art algorithms such as DDPG [29], PPO [38], TD3 [16], SAC [18] and BCQ [15](the first two codes are from OpenAI's baselines repository and the last two are published or suggested by their authors). In fact, BCQ is proposed for off-policy reinforcement learning without exploration similar to imitation learning, with a convex combination between minimum and maximum values of two critics as target value. Here, we train the re-tuned version of BCQ, also called BCQ, by replacing the target value of TD3 with the combination value mentioned above and using the default weighting parameter published by the original BCQ's codes. We run each algorithm with one million time steps and take a test every 5000 steps. In each test, the summation of all rewards in each episode without exploration noise and discount factor is viewed as a return, and every policy is evaluated with the average return over 10 episodes for reducing the uncertainty of environment. We eventually demonstrate the average return and its standard deviation over 10 trails or seeds.

Figure 5 shows the learning curves with smoothing for better visualization and Table II represents the average returns of the last 5 evaluations. The results in Figure 5 and Table II show that the average return of TADD is higher than all other algorithms in Ant-v2, HalfCheetah-v2, Hopper-v2, Swimmer-v2, Walker2d-v2, and achieves the maximum value in remaining environments due to their score bounding. TADD promotes largely compared to TD3 in Swimmer-v2, also happening in BCQ, which implies that mitigating underestimation bias has a huge impact on performance increasing. Moreover, we observe that the standard deviation of TADD becomes smaller, perhaps resulting from the introduction of an additional critic. Owing to the total three critics, TADD can prevent the local optimums of a single initialized neural network, further reducing the impact of different network initialization and becoming more robust. Overall, Figure 5 and Table II show that TADD outperforms or matches all other algorithms in the final performance.

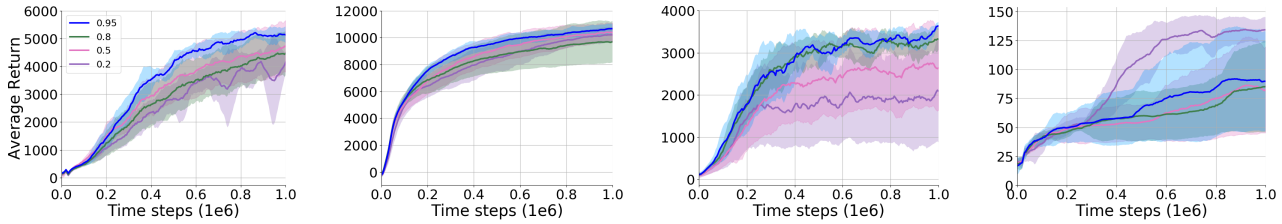
In addition to the final reward performance, we also evaluate the computational cost with floating-point operations (FLOPs) and running time. Essentially, our TADD adds two more same fully connected networks into TD3, one from triplet critics and the other from average Q-value. However, it does not bring too many FLOPs due to the simplicity of these networks. Moreover, we run TADD and TD3 in the same environment at the same time, with Nvidia GeForce GTX 1080Ti. TADD needs only one more hour to run, which is normal and also happens in the comparison between TD3 and DDPG. In short, our proposed TADD does not cost too much computation.

C. Ablation Studies

We perform ablation studies for evaluating the effect of the weighting and averaging techniques. We provide results for the intermediate state TDD. As shown in the second column of Table II, the TDD with only a weighting technique achieves better performance than TD3 and DDPG in most tasks. After adding the averaging technique into TDD, i.e. TADD, the performance is further improved, which implies that the average Q-value can promote performance through variance reduction. On the other hand, Figure 6 presents the results under different weighting parameter β. As shown in Figure 6, while the significance of the weighting parameter varies task to task, larger β corresponds to the better performance in most tasks, indicating the underestimation bias is smaller than the overestimation. However, the Swimmer-v2 is only opposite in all tasks due to the larger underestimation bias from double critics, which also explains why TD3 performs worse than DDPG in Figure 5(d). Therefore, employing a proper weight of β for reducing estimation bias plays a critical role in improving performance.

VI. Conclusion

Estimation bias has been identified as a key challenge in the value-based problem. In this work, we have shown, both in theory and in practice, the existence of estimation bias, and especially underestimation, in deep deterministic actor-critic algorithms. To reduce this bias, we proposed a Triplet Average Deep Deterministic policy gradient algorithm (TADD) comprising a structure of triplet critics and an average Q-value technique. The former is applied to balance overestimation and underestimation bias and the latter taking into account the connection between bias and per-update variance is used to address high variance. We have demonstrated the effectiveness of



(a) Ant-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Swimmer-v2

Fig. 6: Qualitative performances on the OpenAI gym continuous control tasks under different weighting parameter $\beta = 0.95, 0.8, 0.5, 0.2$. The darker lines show the median over the seeds. The shaded region represents a standard deviation of the averaged evaluation.

TADD through theoretical analysis and extensive experiments. More specifically, the results on numerous OpenAI continuous control tasks have shown that our algorithm achieves superior performance over the other state-of-the-art methods.

REFERENCES

[1] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[2] O. Anschel, N. Baram, and N. Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*, 2017.

[3] K. Asadi, C. Allen, M. Roderick, A.-r. Mohamed, G. Konidaris, M. Littman, and B. U. Amazon. Mean actor critic. *stat*, 1050:1, 2017.

[4] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. *International Conference on Learning Representations*, 2017.

[5] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

[6] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[7] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[8] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. Dynamic programming and optimal control. volume 1. Athena scientific Belmont, MA, 1995.

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

[10] H. Cai, L. Zhu, and S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. *International Conference on Learning Representations*, 2019.

[11] C. Colas, O. Sigaud, and P.-Y. Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

[12] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, and F. Porikli, Dynamical hyperparameter optimization via deep reinforcement learning in tracking, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, doi://10.1109/TPAMI.2019.2956703, 2019

[13] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *International Conference on Machine Learning*, 2018.

[14] R. Fox, A. Pakman, and N. Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[15] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *International Conference on Machine Learning*, 2019.

[16] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

[17] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1352–1361, 2017.

[18] 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*, 2018.

[19] X. Dong, J. Shen, D. Wu, K. Guo, X. Jin, and F. Porikli. Quadruplet network with one-shot learning for fast visual object tracking. *IEEE Trans. on Image Processing*, 28(7):3516–3527, 2019.

[20] H. V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[21] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[22] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.

[25] S. Khadka, S. Majumdar, S. Miret, E. Tumer, T. Nassar, Z. Dwiel, Y. Liu, and K. Tumer. Collaborative evolutionary reinforcement learning. *International Conference on Machine Learning*, 2019.

[26] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1188–1200, 2018.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[28] D. Lee, B. Defourny, and W. B. Powell. Bias-corrected q-learning to control max-operator bias in q-learning. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 93–99, 2013.

[29] 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*, 2016.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[31] 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*, 518(7540):529, 2015.

[32] O. Nachum, M. Norouzi, G. Tucker, and D. Schuurmans. Smoothed action value functions for learning gaussian policies. In *International Conference on Machine Learning*, 2018.

[33] P. Nagarajan, G. Warnell, and P. Stone. Deterministic implementations for reproducibility in deep reinforcement learning. In *Thirty-Third AAAI Conference on Artificial Intelligence Workshop*, 2019.

[34] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

[35] A. Pourchot and O. Sigaud. Cem-rl: Combining evolutionary and gradient-based methods for policy search. *International Conference on Learning Representations*, 2019.
[36] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
[37] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.
[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[39] J. Shen, X. Tang, X. Dong, and L. Shao. Visual object tracking by hierarchical attention siamese network. *IEEE Trans. on Cybernetics*, doi://10.1109/TCYB.2019.2936503, 2019.
[40] 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*, 529(7587):484, 2016.
[41] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.
[42] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.
[43] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.
[44] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
[45] S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.
[46] W. Wang, J. Shen, and H. Ling. A deep network solution for attention and aesthetics aware photo cropping. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1531–1544, 2019.
[47] M. Ye, X. Lan, Z. Wang, and P. C. Yuen. Bi-directional center-constrained top-ranking for visible thermal person re-identification. *IEEE Trans. on Information Forensics and Security*, vol. 15, pp. 407–419, 2020.
[48] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[49] H. P. van Hasselt. *Insights in reinforcement learning*. Hado van Hasselt, 2011.
[50] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *International Conference on Learning Representations*, 2017.
[51] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. Dueling network architectures for deep reinforcement learning. *International Conference on Machine Learning*, 2016.
[52] Z. Zhang, Z. Pan, and M. J. Kochenderfer. Weighted double q-learning. In *International Joint Conference on Artificial Intelligence*, pages 3455–3461, 2017.

Dongming Wu is currently working toward the Ph.D. degree in the School of Computer Science, Beijing Institute of Technology, Beijing, China. He received the B.S. degree in Computing Science from Beijing Institute of Technology in 2019. His current research interests include deep reinforcement learning.

Xingping Dong received his Ph.D. degree from Beijing Institute of Technology in 2019. From 2016 to 2018, he was a joint Ph.D. student in College of Engineering and Computer Science, Australian National University. He is currently a Research Scientist at Inception Institute of Artificial Intelligence, UAE. His current research interests include deep learning and object tracking.

Jianbing Shen (M'11-SM'12) is currently acting as the Lead Scientist at the Inception Institute of Artificial Intelligence, Abu Dhabi, UAE. He is also an adjunct Professor with the School of Computer Science, Beijing Institute of Technology, China. He has published more than 100 top journal and conference papers with Google Scholar Citations 5800 times, and ten papers are selected as the ESI Hightly Cited or ESI Hot Papers. His current research interests include deep learning, computer vision, autonomous driving, deep reinforcement learning, and intelligent systems. He is an Associate Editor of *IEEE Trans. on Image Processing*, *IEEE Trans. on Neural Networks and Learning Systems*, and other journals.

Steven C. H. Hoi is an Associate Professor of the School of Information Systems, Singapore Management University, Singapore. Prior to joining SMU, he was an Associate Professor with Nanyang Technological University, Singapore. His research interests are machine learning and data mining and their applications to multimedia information retrieval (image and video retrieval), social media and web mining, and computational finance, *et al.*, and he has published over 150 refereed papers in top conferences and journals in these related areas. Currently he is the Editor-in-Chief of *Neurocomputing*, Associate Editor of *IEEE TPAMI*.