# OPPORTUNISTIC ACTOR-CRITIC (OPAC) WITH CLIPPED TRIPLE Q-LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Despite being the most successful model-free deep reinforcement learning (RL) algorithms in recent years, Soft Actor-Critic (SAC) and Twin Delayed Deep Deterministic Policy Gradient (TD3) have their respective downsides–TD3 performs well in simple tasks, while SAC does so in relatively complicated ones. However, they also suffer from underestimation due to Clipped Double Q-learning, i.e., taking a minimum of two Q-values. This paper introduces Opportunistic Actor-Critic (OPAC), an ensemble model-free deep RL algorithm that performs well in simple and complex tasks. OPAC combines the features of TD3 and SAC under one roof to retain their respective benefits. It also employs three critics and considers taking the mean of the smallest two Q-values for updating the shared target, dubbed Clipped Triple Q-learning. Our analytical results establish that Clipped Triple Q-learning incurs less underestimation than Clipped Double Q-learning. Furthermore, we have systematically evaluated OPAC in MuJoCo environments, and the empirical results indicate that OPAC attains higher average rewards than the current baselines.

## 1 INTRODUCTION

Model-free deep reinforcement learning (RL) has been enormously successful in many fields, like gaming (Mnih et al., 2013; Silver et al., 2016; Alonso et al., 2021), robotic control (Gu et al., 2017; Haarnoja et al., 2018a), decision-making (Chen et al., 2019), and many more. However, despite achieving overwhelming performance in these tasks, applying model-free deep RL in continuous domains confronts several significant obstacles. These include over or under-estimation bias, sample complexity, convergence brittleness concerning the hyper-parameters, inefficient exploration, and sub-optimal policies.

Getting accurate estimates of the Q-values is essential in model-free deep RL, as it enables the actor to learn a robust policy. Actor-critic architectures form the base of the most popular and successful model-free methods. Konda & Tsitsiklis (1999) presented a class of actor-critic algorithms with provable convergence properties. Nevertheless, one glaring drawback of traditional actor-critic methods is that they suffer from an overestimation bias while estimating the Q-values of the critics. For this reason, Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) based algorithms fail to deliver promising results on continuous control domains. Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) addressed the issue of overestimation in actor-critic methods. It employed Clipped Double Q-learning, delayed policy updates, and target policy smoothing. These techniques also contributed significantly to the stability of TD3. It is an off-policy algorithm that trains a deterministic actor and learns from past samples with an experience replay memory. However, TD3 underestimates the Q-values due to the usage of Clipped Double Q-learning (Wu et al., 2020; He & Hou, 2020). In other words, TD3 considers the minimum of two Q-values while computing the shared target, leading to underestimation. Furthermore, TD3 requires careful hyper-parameter tuning to converge to the optimal policy.

To combat the convergence brittleness of model-free deep RL, the maximum entropy framework was incorporated in Soft Actor-Critic (SAC) (Haarnoja et al., 2018b;c). The maximum entropy framework has been used in several areas e.g. Ziebart et al. (2008); Todorov (2008); Toussaint (2009); Rawlik et al. (2013). Like TD3, SAC also utilizes Clipped Double Q-learning and suffers from underestimation. But as it also aims to maximize entropy, it ultimately achieves substantially higher

rewards. SAC outperformed prior state-of-the-art methods (including TD3) in terms of performance and sample efficiency. However, due to the Gaussian nature of its policy, SAC struggles with poor exploration, resulting in borderline performance in relatively simpler continuous control environments (Ciosek et al., 2019).

Although TD3 and SAC are two of the most successful model-free deep RL algorithms, they fail to deliver consistently in a wide range of continuous control environments. For instance, TD3 performs best in simple tasks, but SAC does well in relatively complicated ones. So, can we combine the features of TD3 and SAC to get an algorithm that performs well in easy and complex tasks? In this context, we introduce Opportunistic Actor-Critic (OPAC), an ensemble model-free deep RL algorithm that puts the central features of TD3 and SAC under one roof to retain their respective benefits. It is an off-policy algorithm that trains a stochastic actor. Also, OPAC uses three critics and considers the mean of the smallest two Q-values to update the shared target. We call this strategy Clipped Triple Q-learning, and our analytical results show that it incurs less underestimation than Clipped Double Q-learning, i.e., taking the minimum of two Q-values. As per our knowledge, taking the mean of the smallest two Q-values is a novel target-update strategy, and no prior works exist on this. We have systematically evaluated OPAC on MuJoCo (Todorov et al., 2012) continuous control tasks interfaced through OpenAI Gym (Brockman et al., 2016). The empirical results demonstrate that OPAC consistently yields higher average rewards over time than current baselines.

## 2 Related Works

Several algorithms deal with controlling the estimation bias in actor-critic-based architectures. These primarily differ in terms of the number of critics used and the target-update rule. Triplet-Average Deep Deterministic Policy Gradient (TADD) (Wu et al., 2020) uses three critics and takes a convex combination of the minimum of the first two Q-values and the third Q-value. Additionally, it uses the arithmetic mean of the third Q-value of the last $K$ time steps to reduce variance. A shortcoming of this approach is its dependence on the convex combination coefficient $\beta$, which is an environment-dependent hyper-parameter and is responsible for balancing the over-estimation and under-estimation bias. Saglam et al. (2021) also uses three critics, but its target-update rule is a modification of the Double Q-learning (van Hasselt et al., 2015) algorithm. It first takes the maximum of two Q-values and then compares it with a third Q-value to determine the minimum. Unlike TADD, there is no environment-specific hyper-parameter in this work. Weighted Delayed Deep Deterministic Policy Gradient (WD3) (He & Hou, 2020) employs two critics and reduces the underestimation by employing a weighted smooth update mechanism.

There also exist approaches that obtain better function estimates by not focusing on the number of critics or the target-update rule. Duan et al. (2020) shows that the estimation bias is mitigated by learning a distribution function of state-action returns instead of directly learning the Q-values. Softmax Deep Double Deterministic Policy Gradients (SD3) (Pan et al., 2020) uses the Boltzmann-Softmax operator for value function estimation. It can not only lessen the overestimation of DDPG but also smooth the optimization landscape. Another approach is in Lyu et al. (2022) where the prospect of having double actors and regularized critics is explored. Double actors possess a bias alleviation property, and regularized critics control the large difference in value estimation between two independent critics.

## 3 Preliminaries

In this section, we briefly go over the fundamentals of reinforcement learning and maximum entropy reinforcement learning. The provided mathematical notations have been used throughout.

### 3.1 Reinforcement Learning

Markov Decision Processes (MDPs) are most commonly defined by the 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p$ represents the transition function, and $r$ represents the reward function. The transition function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ signifies the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$.

The goal in an MDP is to find an optimal policy. Policy is a function $\pi$ that specifies the action $\pi(s)$ to choose when in state $s$. It is often expressed as a conditional probability distribution over the actions given the states. Reinforcement learning considers an agent interacting with its environment to learn a behavior that maximizes the accumulated rewards. The agent in RL could be thought of as the decision-maker in MDPs and the environment could be thought of as the setting on which the MDP is defined. Thus, RL can be viewed as a policy search in an MDP. The standard RL objective is the expected sum of rewards given by $\sum_{t=0}^{\infty} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t)]$. The goal in RL is to learn a policy $\pi(a_t \mid s_t)$ that maximizes this objective.

### 3.2 Maximum Entropy Reinforcement Learning

Entropy is a measure of unpredictability of a random variable. Suppose $x$ is a random variable with probability mass or density function $\mathcal{P}$. The entropy $\mathcal{H}$ of $x$ is computed by $\mathcal{H}(\mathcal{P}) = \mathbb{E}_{x\sim\mathcal{P}}[-\log \mathcal{P}(x)]$.

The maximum entropy objective extends the normal RL objective by including an entropy term, so that the optimal policy also strives to maximize its entropy at each visited state, $\pi^* = \arg\max_\pi \sum_{t=0}^{\infty} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))]$. Here $\alpha$ is the temperature parameter that determines the relative importance of the entropy term versus the reward. It controls the stochastic nature of the optimal policy.

## 4 Features of OPAC

Before we delve into the mathematics of OPAC, let us discuss its constituent features taken from TD3 and SAC. It is to provide an intuition of the advantages of the chosen features. First, we consider the features from SAC that are present in OPAC.

- **Maximum Entropy Framework**: The concept of maximizing entropy alongside the expected sum of rewards is central to this framework. Here the policy is rewarded for broadening its boundaries while rejecting sub-optimal actions. The policy can also capture a wide range of near-optimal behavior, assigning the same probabilities to actions that appear equally desirable. Maximizing the entropy boosts learning speed compared to approaches that optimize the standard RL objective. Another justification for using this framework is to balance the slight underestimation incurred in Clipped Triple Q-learning with the expected entropy value.
- **Automatic Entropy Adjustment**: SAC was shown to be brittle with respect to the temperature parameter $\alpha$. Fixing the value of $\alpha$ in the maximum entropy framework is not a good approach. Determining the ideal value of $\alpha$ is a complex undertaking. Furthermore, the entropy might fluctuate unpredictably across tasks and during training as the policy improves. Therefore, $\alpha$ in OPAC is automatically adjusted similarly to SAC.

Now we list the features borrowed from TD3 in OPAC and their reason for inclusion.

- **Target Networks**: In deep RL, target networks help stabilize training (Mnih et al., 2015). Deep neural networks are great at approximating functions, but take a long time to converge since they require many gradient adjustments. Target networks provide a clear goal in the learning process and allow for higher training data acquisition in such situations. Like TD3, OPAC leverages target networks for both actors and critics.
- **Delayed Policy Updates**: Updating the actor at a lesser frequency than the critic reduces the risk of error before performing a policy update. The core idea is to hold back updating the policy until the error in the critic's Q-value estimation is as small as possible. Correlation between the critics is also regularized in this way.
- **Addition of Noise**: At any instance, an RL-tuple is defined as $(s, a, s', r)$ where $s$ is the current state, $a$ is the playable action from state $s$, $s'$ is the new-state reached, and $r$ is the reward obtained. We can obtain $a'$, i.e., the playable action from next-state $s'$ by feeding it into the target policy. Adding a small amount of Gaussian noise to $a'$ facilitates exploration.

Only a unique feature of OPAC is left to discuss - Clipped Triple Q-learning. Section 6 is entirely devoted to it.

**Algorithm 1:** Opportunistic Actor-Critic (OPAC)

Initialize policy parameters $\theta$, Q-function parameters $\phi_1$, $\phi_2$, $\phi_3$ and an empty replay buffer $\mathcal{R}$
Initialize target network parameters $\theta' \leftarrow \theta$, $\phi'_i \leftarrow \phi_i$ $(i = 1, 2, 3)$
Initialize the replay buffer $\mathcal{R}$
**for** $t = 1, \ldots, T$ **do**
  Select action with exploration noise: $a \sim \pi_\theta(\cdot \mid s) + \eta$, where $\eta \sim \mathcal{N}(0, 0.1)$
  After observing the reward $r$ and new state $s'$, store the transition tuple $(s, a, s', r)$ in $\mathcal{R}$
  Sample a batch of transitions $\mathcal{B} = \{(s, a, s', r)\}$ from $\mathcal{R}$
  Compute next action: $a' \leftarrow \tilde{a} + \eta$, where $\tilde{a} \sim \pi_{\theta'}(\cdot \mid s')$ and $\eta \sim \mathcal{N}(0, \delta)$
  Compute the target: $y(r, s') \leftarrow r + \gamma \left( \operatorname{mst}_{i=1,2,3} Q_{\phi'_i}(s', a') - \alpha \log \pi_{\theta'}(a' \mid s') \right)$
  Update Q-functions: $\nabla_{\phi_i} \frac{1}{|\mathcal{B}|} \sum \left( Q_{\phi_i}(s, a) - y(r, s') \right)^2$
  **if** $t \bmod D = 0$ **then**
    Update the policy: $\nabla_\theta \frac{1}{|\mathcal{B}|} \sum (Q_{\phi_1}(s, \pi_\theta(s)) - \alpha \log \pi_\theta(\pi_\theta(s) \mid s))$
    Update the target networks and tune $\alpha$:

$$\theta' \leftarrow \tau\theta' + (1 - \tau)\theta$$
$$\phi'_i \leftarrow \tau\phi'_i + (1 - \tau)\phi_i$$
$$\alpha \leftarrow \alpha - \lambda \nabla_\alpha J(\alpha)$$

  **end**
**end**

## 5 Opportunistic Actor-Critic

OPAC is derived from Soft Policy Iteration which was introduced in Haarnoja et al. (2018c;b). Since SAC inspires our derivation, both have many similarities, and we will skip the repetitive details. Deep networks must approximate the soft Q-function and the policy to translate the Soft Policy Iteration into a continuous setting. We alternate between optimizing the soft Q-function and policy network by stochastic gradient descent to build our algorithm of OPAC. Let $Q_\phi$ and $\pi_\theta$ denote the soft Q-function and the policy with parameters $\phi$ and $\theta$ respectively. $\phi$ is trained to minimize the soft Bellman-error as follows

$$J_Q(\phi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{R}} \left[ \frac{1}{2} \left( Q_\phi(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V_{\phi'}(s_{t+1})]) \right)^2 \right], \tag{1}$$

where $\mathcal{R}$ is the replay buffer. The value function parameters $V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)]$, are put in Equation 1 and optimized by stochastic gradient descent:

$$\nabla_\phi J_Q(\phi) = \nabla_\phi Q_\phi(s_t, a_t) \big( Q_\phi(s_t, a_t) - (r(s_t, a_t) + \gamma Q_{\phi'}(s_{t+1}, a_{t+1}) - \alpha \log \pi_{\theta'}(a_{t+1} \mid s_{t+1})) \big), \tag{2}$$

where $\phi'$ and $\theta'$ denote the parameters of the target Q-function and that of target policy respectively. $\theta$ can be learned by minimizing the KL-divergence in the policy improvement equation (given in Haarnoja et al. (2018c)) as follows

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{R}}[\mathbb{E}_{a_t \sim \pi_\theta(\cdot \mid s_t)}[\alpha \log \pi_\theta(a_t \mid s_t) - Q_\phi(s_t, a_t)]]. \tag{3}$$

Therefore we need to compute, $\nabla_\theta \mathbb{E}_{a_t \sim \pi_\theta(\cdot \mid s_t)}[\alpha \log \pi_\theta(a_t \mid s_t) - Q_\phi(s_t, a_t)]$. As $Q_\phi(s_t, a_t)$ does not directly depend on $\theta$, its gradient cannot be computed over $\theta$. Since the policy is a Gaussian distribution with parameters $\mu_\theta$ and $\sigma_\theta$, we have

$$a_t = \mu_\theta(s_t) + \kappa_t \sigma_\theta(s_t), \text{ where } a_t \sim \pi_\theta(\cdot \mid s_t) \text{ and } \kappa_t \sim \mathcal{N}(0, 1). \tag{4}$$

It is convenient to use the re-parameterization trick for obtaining the gradient. Setting $a_t = f_\theta(\kappa_t, s_t) = \mu_\theta(s_t) + \kappa_t \sigma_\theta(s_t)$ and combining Equation 3 and Equation 4, we get

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{R}}[\mathbb{E}_{\kappa_t \sim \mathcal{N}(0,1)}[\alpha \log \pi_\theta(f_\theta(\kappa_t, s_t) \mid s_t) - Q_\phi(s_t, f_\theta(\kappa_t, s_t))]], \tag{5}$$

whose gradient w.r.t $\theta$ can be obtained by

$$\nabla_\theta J_\pi(\theta) = \nabla_\theta \log \pi_\theta(a_t \mid s_t) + (\nabla_{a_t} \log \pi_\theta(a_t \mid s_t) - \nabla_{a_t} Q_\phi(s_t, a_t)) \nabla_\theta f_\theta(\kappa_t, s_t). \tag{6}$$

An off-policy algorithm for OPAC is now ready with a fixed $\alpha$. We have discussed the need for automating $\alpha$ and will do it the same way as SAC. The standard maximum entropy learning problem

for OPAC can be reformulated as: while aiming to maximize the expected return, the policy should also satisfy a minimum entropy constraint,

$$\max_{\pi_0,\ldots,\pi_T} \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t)\right], \text{ s.t. } \forall t, \mathcal{H}(\pi_t) \geq \mathcal{H}_0, \tag{7}$$

where $\mathcal{H}_0$ is a predefined minimum policy entropy threshold. The optimal value of $\alpha$ can be learned by minimizing the objective function $J(\alpha) = \mathbb{E}_{a_t \sim \pi_t(\cdot|s_t)}[-\alpha \log \pi_t(a_t \mid s_t) - \alpha \mathcal{H}_0]$ (see Haarnoja et al. (2018c) for more details).

The formal algorithm of OPAC is listed in Algorithm 1. When computing the shared target, the $\mathrm{mst}(\cdot)$ operator denotes the usage of Clipped Triple Q-learning. Actors are updated by gradient ascent once every $D$ iteration, while the critics are updated by stochastic gradient descent in every iteration. Like TD3, OPAC considers the output of the first critic-model while updating the policy. The target networks are updated by Polyak-averaging once every $D$ iteration. The algorithm runs for a total of $T$ time steps, and $J(\alpha)$ is the objective function for learning $\alpha$.

## 6 Clipped Triple Q-learning

Single-critic algorithms witness overestimation, while using double-critics often brings about underestimation. The main idea behind employing three critics in OPAC is to combine these opposite biases to achieve better estimates. TD3 and SAC utilize Clipped Double Q-learning, i.e., consider the minimum of the two Q-values while computing the shared target. On the contrary, OPAC uses a novel target-update strategy–taking the mean of the smallest two Q-values. We name this strategy Clipped Triple Q-learning. This approach is undoubtedly pessimistic, but not as much as taking the minimum of all the Q-values.

### 6.1 Proof of Convergence of Clipped Triple Q-learning

Clipped Triple Q-learning is presented formally in Theorem 1. The technique used to prove Theorem 1 has been used for showing the convergence of many prior Q-learning based algorithms like Double Q-learning (Hasselt, 2010), Clipped Double Q-learning (Fujimoto et al., 2018), and TADD (Wu et al., 2020).

**Theorem 1** (Clipped Triple Q-learning)**.** *Consider a finite MDP where,*

*(i) Each state-action pair $(s, a)$ is sampled an infinite number of times.*

*(ii) Q-values are stored in a lookup table.*

*(iii) $Q^A$, $Q^B$, and $Q^C$ receive an infinite number of updates.*

*For $\gamma \in [0, 1)$ let the following conditions hold:*

*(iv) The learning rates satisfy the Robbins-Monro conditions: $\alpha_t(s, a) \in [0, 1]$, $\sum_t \alpha_t(s, a) = \infty$, $\sum_t (\alpha_t(s, a))^2 < \infty$ with probability 1, and $\alpha_t(s, a) = 0$, for all $(s, a) \neq (s_t, a_t)$.*

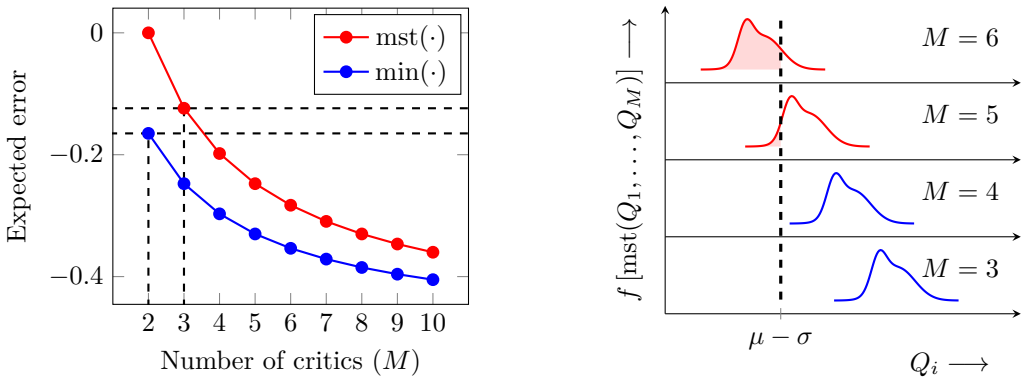
*(v) $\mathrm{Var}[r(s, a)] < \infty$, for all $(s, a)$.*

*The optimal action-value function $Q^*$ is as defined by the Bellman optimality equation. If an MDP satisfies the above conditions then Clipped Triple Q-learning converges to $Q^*$ with probability 1.*

Conditions (iv) (Robbins & Monro, 1951) and (v) also occur in the theorem of Clipped Double Q-learning (Fujimoto et al., 2018). The proof of Theorem 1 has been deferred to Section A.1 of the Appendix. We now present an analysis of the estimation error of Clipped Triple Q-learning which establishes that Clipped Triple Q-learning incurs less underestimation than Clipped Double Q-learning.

### 6.2 Expected Error of Clipped Triple Q-learning

Define $Q_1, Q_2, \ldots, Q_M$ as independent Q-value estimates of $M$ critics. The critics are parameterized by $\phi_i$'s for all $i = 1, \ldots, M$. Let $\mathrm{mst}(Q_1, \ldots, Q_M)$ or $\mathrm{mst}(\cdot)$ denote computing the mean of




(a) (b)

Figure 1: (a) A comparison of the expected underestimation between two target-update strategies: $\min(\cdot)$ and $\text{mst}(\cdot)$. The dashed horizontal lines indicate that the expected underestimation of $\text{mst}(Q_1, Q_2, Q_3)$ (in red dot) is less than that of $\min(Q_1, Q_2)$ (in blue dot). Here $\gamma = 0.99$ and $\epsilon = 0.5$. (b) The distribution of $\text{mst}(Q_1, \ldots, Q_M)$ for randomly sampled numbers from an interval. The red filled region, $f\left[\text{mst}(Q_1, \ldots, Q_M) < \mu - \sigma\right]$, indicates the probability that $\text{mst}(Q_1, \ldots, Q_M)$ violates the lower bound $\mu - \sigma$ ($\mu$ and $\sigma$ denote the mean and the standard deviation of the Q-values respectively). Only $\text{mst}(Q_1, Q_2, Q_3)$ and $\text{mst}(Q_1, \ldots, Q_4)$ are lower bounded by $\mu - \sigma$.

the smallest two Q-values in a total of $M$ Q-values. Any algorithm that employs $\text{mst}(\cdot)$ will have its shared target calculated as

$$y(r, s') \leftarrow r + \gamma \underset{i=1,\ldots,M}{\text{mst}} Q_{\phi_i'}(s', \pi_{\theta'}(s')), \tag{8}$$

$\phi_i'$ and $\theta'$ denote the parameters of the target networks for the critic and actor, respectively. Let $Q^*$ be the hypothetical true Q-function the critics attempt to approximate. Due to the noise induced by function approximation, there exists an error term $Y_i = Q_{\phi_i'}(s', \pi_{\theta'}(s')) - Q^*(s', \pi_{\theta'}(s'))$ for all $i = 1, \ldots, M$. One important assumption can be made without loss of generality: all the $Y_i$'s are independent, identical, and uniformly distributed over an interval $[-\epsilon, \epsilon]$. The error induced by the $\text{mst}(\cdot)$ operation in Equation 8 can be written as,

$$\begin{aligned} Z^M_{\text{mst}} &= r + \gamma \underset{i=1,\ldots,M}{\text{mst}} Q_{\phi_i'}(s', \pi_{\theta'}(s')) - (r + \gamma\, Q^*(s', \pi_{\theta'}(s'))) \\ &= \gamma \left( \underset{i=1,\ldots,M}{\text{mst}} Q_{\phi_i'}(s', \pi_{\theta'}(s')) - Q^*(s', \pi_{\theta'}(s')) \right) \\ &= \gamma \left( \frac{Y_{(1)} + Y_{(2)}}{2} \right). \end{aligned} \tag{9}$$

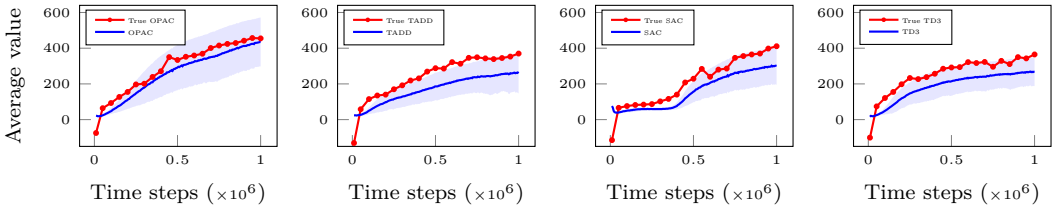
$Y_{(1)}$ and $Y_{(2)}$ in Equation 9 denote the 1-th and 2-th order statistic respectively of the $Y_i$'s. We need to calculate the expected error $\mathbb{E}[Z^M_{\text{mst}}]$ which is written as

$$\mathbb{E}\left[Z^M_{\text{mst}}\right] = \gamma\, \mathbb{E}\left[\frac{Y_{(1)} + Y_{(2)}}{2}\right] = \frac{\gamma}{2}\left(\mathbb{E}\left[Y_{(1)}\right] + \mathbb{E}\left[Y_{(2)}\right]\right). \tag{10}$$

Theorem 2 helps us analyze the expected error, and its proof is in Section A.2 of the Appendix.

**Theorem 2.** *Assume that every $Y_i = Q_{\phi_i'}(s', \pi_{\theta'}(s')) - Q^*(s', \pi_{\theta'}(s'))$ for $i = 1, \ldots, M$, is independently and identically uniformly distributed in the interval $[-\epsilon, \epsilon]$. Then the average underestimation of the $\text{mst}(\cdot)$ operation of all the $Y_i$'s is $\mathbb{E}[Z^M_{\text{mst}}] = \frac{2-M}{M+1}\epsilon\gamma$.*

Let $\min(\cdot)$ denote computing the minimum of all the $M$ Q-values. According to Wu et al. (2020), the expected underestimation of $\min(\cdot)$ is $\mathbb{E}[Z^M_{\min}] = -\frac{M-1}{M+1}\epsilon\gamma$. From Theorem 2 it is evident that $\mathbb{E}[Z^M_{\min}] < \mathbb{E}[Z^M_{\text{mst}}]$ (for $M \geq 2$), i.e., $\text{mst}(\cdot)$ is a better target-update strategy than $\min(\cdot)$. Figure 1a clearly illustrates this fact. Putting $M = 3$ in $\mathbb{E}[Z^M_{\text{mst}}]$ and $M = 2$ in $\mathbb{E}[Z^M_{\min}]$ we have: $\mathbb{E}[Z^3_{\text{mst}}] =$




(a) OPAC (b) TADD (c) SAC (d) TD3

Figure 2: Measuring the bias in the value estimates of OPAC, TADD, SAC, and TD3 on the Ant-v2 environment of MuJoCo over 1 million time steps. The red dotted line indicates the trajectory of the true Q-values, while the blue line corresponds to the predicted Q-values.

$-\frac{1}{4}\epsilon\gamma$ and $\mathbb{E}[Z^2_{\text{min}}] = -\frac{1}{3}\epsilon\gamma$. Therefore, it is clear that the expected underestimation of Clipped Triple Q-learning ($\mathbb{E}[Z^3_{\text{mst}}]$) is less than Clipped Double Q-learning ($\mathbb{E}[Z^2_{\text{min}}]$).

The minimization operation performed by Clipped Double Q-learning is equivalent to computing the lower confidence bound of the Q-values with the coefficient of the uncertainty term being one (Ciosek et al., 2019). Experimental evaluations suggest $\text{mst}(Q_1, \ldots, Q_M)$ is bounded below by $\mu - \sigma$ only when $M \leq 4$, as depicted in Figure 1b. Here $\mu$ and $\sigma$ denote the mean and the standard deviation of the Q-values respectively.

## 7 Estimation Bias of OPAC

According to Algorithm 1, the shared target in OPAC is computed via

$$y(r, s') \leftarrow r + \gamma \left( \underset{i=1,2,3}{\text{mst}} Q_{\phi'_i}(s', a') - \alpha \log \pi_{\theta'}(a' \mid s') \right). \tag{11}$$

Let $\mathbb{E}[Z_{\text{OPAC}}]$ denote the expected error in the Q-value estimation of OPAC. Here, $Z_{\text{OPAC}}$ is the error induced in Equation 11 due to function approximation. Using the calculations of Section 6.2, we can write $Z_{\text{OPAC}}$ as

$$\begin{aligned} Z_{\text{OPAC}} &= \gamma \left( \underset{i=1,2,3}{\text{mst}} Q_{\phi'_i}(s', a') - Q^*(s', \pi_{\theta'}(s')) - \alpha \log \pi_{\theta'}(a' \mid s') \right) \\ &= \gamma \left( Z^3_{\text{mst}} - \alpha \log \pi_{\theta'}(a' \mid s') \right). \end{aligned}$$

Taking expectation on both sides we have

$$\mathbb{E}[Z_{\text{OPAC}}] = \mathbb{E}[Z^3_{\text{mst}}] - (\alpha\gamma)\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')]. \tag{12}$$

In order to get $\mathbb{E}[Z_{\text{OPAC}}]$, we only need to calculate $\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')]$ since we already know that $\mathbb{E}[Z^3_{\text{mst}}] = -\frac{1}{4}\epsilon\gamma$. The value of $\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')]$ is found to be $-\frac{1}{2}\log\left(2\pi(\sigma^2_{\theta'} + \delta)\right) - \frac{1}{2}$. Complete calculations have been deferred to Section A.3 of the Appendix. Substituting the values of $\mathbb{E}[Z^3_{\text{mst}}]$ and $\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')]$ in Equation 12 we have

$$\mathbb{E}[Z_{\text{OPAC}}] = -\frac{1}{4}\epsilon\gamma + \frac{\alpha\gamma}{2} \cdot \log\left(2\pi(\sigma^2_{\theta'} + \delta)\right) + \frac{\alpha\gamma}{2}. \tag{13}$$

Just like $\mathbb{E}[Z_{\text{OPAC}}]$, we can also calculate the expected error in the Q-value estimation of SAC, TD3, and TADD, i.e., $\mathbb{E}[Z_{\text{SAC}}]$, $\mathbb{E}[Z_{\text{TD3}}]$, and $\mathbb{E}[Z_{\text{TADD}}]$. These are shown in Section A.5 of the Appendix.

Figure 2 demonstrates the expected error incurred byOPAC, TADD, SAC, and TD3 while estimating the Q-values. The plots compare an estimate of the true Q-value with the average value estimate of over $10,000$ states. Beginning from the states sampled from the replay buffer, the true Q-value is estimated using the average discounted return over $50$ episodes following the current policy. There are $21$ red dots in each of the plots, which signify the mean of the true Q-values of those $100$ states sampled from the replay buffer every $50,000$ time step. It is seen that OPAC estimates the Q-values more accurately than others which implies that the expected value of the entropy, i.e., $\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')]$ counters the slight underestimation of Clipped Triple Q-learning.

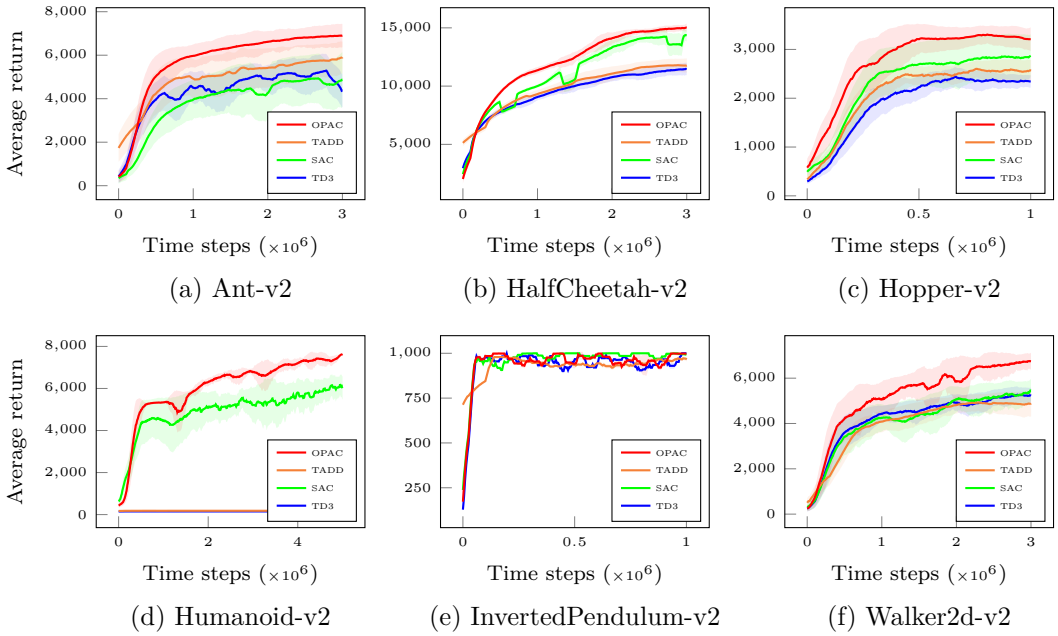



Figure 3: Learning curves of OPAC, SAC, TD3, and TADD on MuJoCo environments. The shaded regions correspond to one standard deviation. The plots suggest that OPAC has a definitive edge over the others in all environments.

## 8 Experiments and Results

We compare the performance of OPAC with TD3 (Fujimoto et al., 2018), SAC (Haarnoja et al., 2018c), and TADD (Wu et al., 2020). All the algorithms are executed in six MuJoCo continuous control tasks, namely, Ant-v2, HalfCheetah-v2, Hopper-v2, Humanoid-v2, InvertedPendulum-v2, and Walker2d-v2. Since those are the simplest environments, the algorithms run in Hopper-v2 and InvertedPendulum-v2 for one million time steps. Whereas in Ant-v2, HalfCheetah-v2, and Walker2d-v2, the algorithms run for three million time steps as they are moderately difficult. For Humanoid-v2, arguably the most challenging environment, we run the algorithms for five million time steps.

This section contains two subsections: comparative evaluation and ablation study. The first subsection, i.e., comparative evaluation, aims to compare the performance of OPAC with the vanilla versions of SAC, TD3, and TADD. The second subsection, ablation study, focuses on understanding the effect of varying the number of critics and changing the target-update rule.

### 8.1 Comparative Evaluation

Figure 3 shows the total average return of OPAC, SAC, TD3, and TADD during training in six MuJoCo environments. Solid curves correspond to the mean, and the shaded region corresponds to one standard deviation. The curves have been smoothed using simple moving average as needed. The plots indicate that OPAC consistently yields higher average rewards over time than the others. In other words, OPAC outperforms SAC, TD3, and TADD in average rewards accumulated over time in all the environments.

We train five instances of each algorithm with only different seed values and then plot the results by averaging over the five trials. The algorithms run purely on an exploratory policy for the first $10,000$ time steps. The policy is evaluated after every $5,000$ time step. Each evaluation step is performed over $50$ episodes. The evaluation reports the mean of the cumulative reward of the $50$ episodes without discount and any noise. All the plots in Section 8 and the Appendix have been generated in this fashion. A comparison of the maximum rewards obtained by OPAC, SAC, TD3, and TADD in each environment is shown in Table 1 and Table 2 (see Section A.6 of the Appendix).

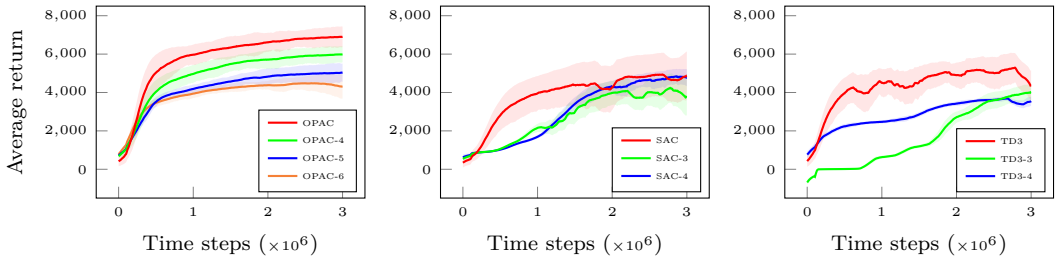
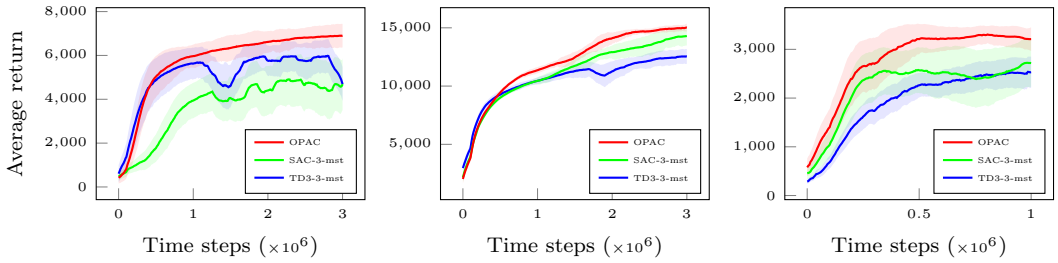



Figure 4: Performance comparison of OPAC, SAC, and TD3 with different number of critics in the Ant-v2 environment. Increasing the number of critics does not enhance the performance of the algorithms. The shaded regions correspond to one standard deviation.

### 8.2 Ablation Studies

Since OPAC constitutes features from SAC and TD3, we limit our ablation study to these algorithms only. Firstly, we perform an ablation on the number of critics. Let OPAC-$4$, OPAC-$5$, and OPAC-$6$ denote the variant of OPAC, which uses four, five, and six critics, respectively, with the target-update strategy as $\mathrm{mst}(\cdot)$. Similarly, let SAC-3, SAC-$4$, TD3-3, and TD3-$4$ denote the variants of SAC and TD3 with three and four critics, respectively. The target-update strategy for the variants of TD3 and SAC is $\min(\cdot)$. The plots in Figure 4 empirically verify the analytical results of Section 6.2, i.e., an increase in the number of critics increases the underestimation of $\mathrm{mst}(\cdot)$ and $\min(\cdot)$. Therefore, having more critics can negatively impact the performance of OPAC, SAC, and TD3. Figure 4 is in the Ant-v2 environment. Additional plots in the five remaining MuJoCo environments is given in Figure 6 (see Section A.7 of the Appendix).

Secondly, we perform an ablation on the target-update rule. Let SAC-3-mst and TD3-3-mst denote the variants of SAC and TD3, respectively, that employ $\mathrm{mst}(\cdot)$ on three critics as their target-update strategy. As illustrated in Figure 5, the usage of $\mathrm{mst}(\cdot)$ improves the performance of SAC and TD3, but it is not enough to outperform OPAC. Additional plots in the remaining three MuJoCo environments is provided in Figure 7 (see Section A.9 of the Appendix).




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

Figure 5: Comparing the performance of OPAC with SAC-3-mst and TD3-3-mst. The shaded regions correspond to one standard deviation.

## 9 Conclusions

This paper presents Opportunistic Actor-Critic (OPAC), an ensemble model-free deep RL algorithm that combines the core features of TD3 and SAC to retain their respective benefits. It also uses Clipped Triple Q-learning to mitigate the underestimation. Our analytical results in Section 6.2 prove that Clipped Triple Q-learning incurs less underestimation than Clipped Double Q-learning. Bias plots in Section 7 exhibit that OPAC is better at estimating the Q-values than SAC, TD3, and TADD. The empirical results in Section 8 demonstrate that OPAC consistently outperforms SAC, TD3, and TADD in easy and challenging MuJoCo continuous control tasks.

## REFERENCES

Eloi Alonso, Maxim Peter, David Goumard, and Joshua Romoff. Deep reinforcement learning for navigation in aaa video games. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2133–2139. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/294. URL https://doi.org/10.24963/ijcai.2021/294. Main Track.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, USA, 2nd edition, 2000. ISBN 1886529094.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2765–2771, 2019. doi: 10.1109/ITSC.2019.8917306.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1785–1796, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/a34bacf839b923770b2c360eefa26748-Abstract.html.

Jingliang Duan, Yang Guan, Yangang Ren, Shengbo Eben Li, and Bo Cheng. Addressing value estimation errors in reinforcement learning with a state-action return distribution function. *CoRR*, abs/2001.02811, 2020. URL http://arxiv.org/abs/2001.02811.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396, 2017. doi: 10.1109/ICRA.2017.7989385.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6244–6251, 2018a. doi: 10.1109/ICRA.2018.8460756.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 10–15 Jul 2018b. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018c. URL http://arxiv.org/abs/1812.05905.

Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23, pp. 2613–2621, 2010. URL https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

Qiang He and Xinwen Hou. Wd3: Taming the estimation bias in deep reinforcement learning. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 391–398, 2020. doi: 10.1109/ICTAI50040.2020.00068.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2016. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15.

Jiafei Lyu, Xiaoteng Ma, Jiangpeng Yan, and Xiu Li. Efficient continuous control with double actors and regularized critics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (7):7655–7663, Jun. 2022. doi: 10.1609/aaai.v36i7.20732. URL https://ojs.aaai.org/index.php/AAAI/article/view/20732.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Ling Pan, Qingpeng Cai, and Longbo Huang. Softmax deep double deterministic policy gradients. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/884d247c6f65a96a7da4d1105d584ddd-Abstract.html.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pp. 3052–3056, 2013. ISBN 9781577356332.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

Baturay Saglam, Enes Duran, Dogan C. Cicek, Furkan B. Mutlu, and Suleyman S. Kozat. Estimation error correction in deep reinforcement learning for deterministic actor-critic methods. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 137–144, 2021. doi: 10.1109/ICTAI52525.2021.00027.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach. Learn.*, 38(3):287–308, March 2000. ISSN 0885-6125. doi: 10.1023/A:1007678930559. URL https://doi.org/10.1023/A:1007678930559.

E. Todorov. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pp. 4286–4292, 2008.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1049–1056, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553508. URL https://doi.org/10.1145/1553374.1553508.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.

Dongming Wu, Xingping Dong, Jianbing Shen, and Steven C. H. Hoi. Reducing estimation bias via triplet-average deep deterministic policy gradient. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4933–4945, 2020. doi: 10.1109/TNNLS.2019.2959129.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In Dieter Fox and Carla P. Gomes (eds.), *Proceedings of the 23rd National Conference on Artificial Intelligence*, volume 3 of *AAAI'08*, pp. 1433–1438, 2008. ISBN 978-1-57735-368-3.

# A APPENDIX

## A.1 PROOF OF CONVERGENCE OF CLIPPED TRIPLE Q-LEARNING

We now present the proof of Theorem 1 under deterministic policies. To this end, we first include a lemma from Singh et al. (2000). It originally appears as a proposition in Bertsekas (2000) which was further generalized into the following lemma.

**Lemma 3.** *Let $\|\cdot\|$ denote the maximum norm. Consider a stochastic process $(\zeta_t, \Delta_t, F_t)$, $t \geq 0$ where $\zeta_t, \Delta_t, F_t : X \to \mathbb{R}$ 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 $t = 0, 1, \ldots$. Let $P_t$ be a sequence of increasing $\sigma$-fields such that $\zeta_0$ and $\Delta_0$ are $P_0$ measurable and $\zeta_t, \Delta_t$, and $F_{t-1}$ are $P_t$ measurable for $t = 0, 1, \ldots$. Assume that the following hold:*

*(1) The set $X$ is finite.*

*(2) $\zeta_t(x_t) \in [0, 1]$, $\sum_t \zeta_t(x_t) = \infty$, and $\sum_t (\zeta_t(x_t))^2 < \infty$ with probability 1 and for all $x \neq x_t : \zeta_t = 0$.*

*(3) $\|\mathbb{E}[F_t \mid P_t]\| \leq \kappa\|\Delta_t\| + c_t$, $\kappa \in [0, 1)$ and $c_t$ converges to 0 with probability 1.*

*(4) $\mathrm{Var}[F_t \mid P_t] \leq K(1 + \kappa\|\Delta_t\|)^2$, where $K$ is some constant.*

*Then $\Delta_t$ converges to 0 with probability 1.*

For a finite MDP setting, we maintain three tabular estimates of the value functions: $Q^A$, $Q^B$, and $Q^C$. At each time step we update all of them.

*Proof of Theorem 1.* We apply Lemma 3 with $P_t = \left\{Q_0^A, Q_0^B, Q_0^C, s_0, a_0, \alpha_0, r_1, s_1, \ldots, s_t, a_t\right\}$, $X = S \times A$, $\zeta_t = \alpha_t$. Consider the target mapping $\left(Q_t^A, Q_t^B, Q_t^C\right) \mapsto g\left(Q_t^A, Q_t^B, Q_t^C\right)$ that computes the mean of the smallest two Q-values. Also without loss of generality, let's assume $\Delta_t = Q_t^A - Q^*$.

The conditions (1) and (4) of Lemma 3 hold by the condition (ii) of Theorem 1. Condition (2) of the lemma holds by condition (iv) of the theorem along with our selection of $\zeta_t = \alpha_t$. Following the same notation and argument as Fujimoto et al. (2018) we get

$$\Delta_{t+1}(s_t, a_t) = (1 - \alpha_t(s_t, a_t))\Delta_t(s_t, a_t) + \alpha_t(s_t, a_t)F_t(s_t, a_t),$$

where,

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

The first term $F_t^Q(s_t, a_t)$ is same as Clipped Double Q-Learning. The second term $c_t$ is slightly different and equal to $\gamma g\left(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*), Q_t^C(s_{t+1}, a^*)\right) - \gamma Q_t^A(s_{t+1}, a^*)$. The quantity $\mathbb{E}\left[F_t^Q \middle| P_t\right] \leq \gamma\|\Delta_t\|$ is known to be true due to Bellman operator being a contraction mapping. This implies condition (3) of Lemma 3 holds if we can show that $c_t$ converges to 0 with probability 1.

Let, $\Delta_t^{BA} \triangleq Q_t^B(s_t, a_t) - Q_t^A(s_t, a_t)$ and $\Delta_t^{BC} \triangleq Q_t^B(s_t, a_t) - Q_t^C(s_t, a_t)$. In our case, contrary to Clipped Double Q-Learning, for $c_t$ to converge to 0 both $\Delta_t^{BA}$ and $\Delta_t^{BC}$ needs to converge to 0 with probability 1. The convergence of $\Delta_t^{BA}$ is fairly straightforward and has been laid out in Fujimoto et al. (2018). By appealing to the generality of this result, we see that

$$\begin{aligned} \Delta_{t+1}^{BC}(s_t, a_t) &\triangleq Q_{t+1}^B(s_t, a_t) - Q_{t+1}^C(s_t, a_t) \\ &= (1 - \alpha_t(s_t, a_t))\Delta_t^{BC}(s_t, a_t). \end{aligned}$$

Clearly, $\Delta_t^{BC}$ converges to 0. These signify that we have fulfilled the condition (3) of Lemma 3, and thus $Q^A(s_t, a_t)$ converges to $Q_t^*(s_t, a_t)$. Repeating the same steps for $Q^B(s_t, a_t)$ and $Q^C(s_t, a_t)$ completes the proof. □

### A.2 PROOF OF THEOREM 2

*Proof.* We use the abbreviation "iid" for independent and identically distributed. According to our assumption, $\{Y_1, Y_2, \ldots, Y_M\} \overset{iid}{\sim} U[-\epsilon, \epsilon]$. We can write,

$$\left\{\frac{Y_1}{2\epsilon}, \frac{Y_2}{2\epsilon}, \ldots, \frac{Y_M}{2\epsilon}\right\} \overset{iid}{\sim} U[-\frac{1}{2}, \frac{1}{2}]$$
$$\left\{\frac{Y_1}{2\epsilon} + \frac{1}{2}, \frac{Y_2}{2\epsilon} + \frac{1}{2}, \ldots, \frac{Y_M}{2\epsilon} + \frac{1}{2}\right\} \overset{iid}{\sim} U[0, 1].$$

Let $X_i = \frac{Y_i}{2\epsilon} + \frac{1}{2}$ for $i = 1, \ldots, M$. Then $\{X_1, X_2, \ldots, X_M\} \overset{iid}{\sim} U[0, 1]$. Therefore, $Y_i = x \implies X_i = \frac{x}{2\epsilon} + \frac{1}{2}$ and we have the following probability density function,

$$f\left(Y_{(k)} = x\right) = f\left(X_{(k)} = \frac{x}{2\epsilon} + \frac{1}{2}\right).$$

Here, $Y_{(k)}$ is the $k$-th order statistic of $\{Y_1, Y_2, \ldots, Y_M\} \overset{iid}{\sim} U[-\epsilon, \epsilon]$ and $X_{(k)}$ is the equivalent $k$-th order statistic of $\{X_1, X_2, \ldots, X_M\} \overset{iid}{\sim} U[0, 1]$. The probability distribution of $X_{(k)}$ is a Beta distribution with parameters $k$ and $M - k + 1$,

$$X_{(k)} \sim \mathcal{B}(k, M - k + 1).$$

The expected value of $X_{(k)}$ is,

$$\mathbb{E}\left[X_{(k)}\right] = \frac{k}{M+1}. \tag{14}$$

Since $Y_i = \left(X_i - \frac{1}{2}\right) 2\epsilon$ for $i = 1, \ldots, M$, the expected value of $Y_{(k)}$ can be easily calculated as,

$$\begin{aligned}\mathbb{E}\left[Y_{(k)}\right] &= \mathbb{E}\left[\left(X_{(k)} - \frac{1}{2}\right) 2\epsilon\right] \\ &= 2\epsilon\left(\mathbb{E}[X_{(k)}] - \frac{1}{2}\right).\end{aligned}$$

From 14 we have,

$$\begin{aligned}\mathbb{E}\left[Y_{(k)}\right] &= 2\epsilon\left(\frac{k}{M+1} - \frac{1}{2}\right) \\ &= \epsilon\left(\frac{2k - M - 1}{M+1}\right).\end{aligned} \tag{15}$$

We can obtain $\mathbb{E}[Y_{(1)}]$ and $\mathbb{E}[Y_{(2)}]$ from 15. Putting them in 10 we have,

$$\begin{aligned}\mathbb{E}\left[Z^M_{\text{mst}}\right] &= \frac{\gamma}{2}\left(\mathbb{E}[Y_{(1)}] + \mathbb{E}[Y_{(2)}]\right) \\ &= \frac{\gamma}{2}\left[\epsilon\left(\frac{2 - M - 1}{M+1}\right) + \epsilon\left(\frac{4 - M - 1}{M+1}\right)\right] \\ &= \frac{\gamma}{2}\left[\epsilon\left(\frac{1 - M}{M+1}\right) + \epsilon\left(\frac{3 - M}{M+1}\right)\right] \\ &= \frac{\gamma}{2}\left[\epsilon\left(\frac{1 - M + 3 - M}{M+1}\right)\right] \\ &= \frac{\gamma}{2}\left[\frac{(4 - 2M)\epsilon}{M+1}\right] \\ &= \gamma\left[\frac{(2 - M)\epsilon}{M+1}\right] \\ &\boxed{\therefore \mathbb{E}\left[Z^M_{\text{mst}}\right] = \frac{2 - M}{M+1}\epsilon\gamma}\end{aligned}$$

This completes the proof of Theorem 2. □

### A.3 CALCULATING THE EXPECTED ENTROPY OF OPAC

Recalling Equation 12 from Section 7

$$\mathbb{E}[Z_{\text{OPAC}}] = \mathbb{E}\left[Z^3_{\text{mst}}\right] - (\alpha\gamma)\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')].$$

$\pi_{\theta'}$ is the target Gaussian policy network with parameters $\theta'$. From the re-parameterization trick used in Section 5 and Algorithm 1 we know

$$a' = f_{\theta'}(\kappa, s') + \eta = \mu_{\theta'}(s') + \kappa\sigma_{\theta'}(s') + \eta, \text{ where } \kappa \sim \mathcal{N}(0,1) \text{ and } \eta \sim \mathcal{N}(0,\delta).$$

$\mu_{\theta'}$ and $\sigma_{\theta'}$ are obtained from $\pi_{\theta'}$ itself. Because $s'$ is already known and $\mu_{\theta'}$ and $\sigma_{\theta'}$ are functions of it, we omit unnecessary writing of $s'$ for simplicity. It is easily seen that

$$\begin{aligned}\kappa \sim \mathcal{N}(0,1) &\implies \kappa\sigma_{\theta'} \sim \mathcal{N}(0, \sigma^2_{\theta'}) \\ &\implies \kappa\sigma_{\theta'} + \eta \sim \mathcal{N}(0, \sigma^2_{\theta'} + \delta).\end{aligned}$$

Then,

$$a' = \mu_{\theta'} + \kappa\sigma_{\theta'} + \eta \implies a' \sim \mathcal{N}(\mu_{\theta'}, \sigma^2_{\theta'} + \delta).$$

Since $\pi_{\theta'}$ is a Gaussian policy network, then $\pi_{\theta'}(a' \mid s') \sim \mathcal{N}(\mu_{\theta'}, \sigma^2_{\theta'} + \delta)$ and it can be thought of as the probability distribution function of $\mathcal{N}(\mu_{\theta'}, \sigma^2_{\theta'} + \delta)$. Therefore,

$$\begin{aligned}\pi_{\theta'}(a' \mid s') &= \frac{1}{\sqrt{(\sigma^2_{\theta'} + \delta)}\sqrt{2\pi}} \cdot \exp^{\frac{-\left(a' - \mu_{\theta'}\right)^2}{2\left(\sigma^2_{\theta'} + \delta\right)}} \\ &= \frac{1}{\sqrt{(\sigma^2_{\theta'} + \delta)}\sqrt{2\pi}} \cdot \exp^{\frac{-\left(\mu_{\theta'} + \kappa\sigma_{\theta'} + \eta - \mu_{\theta'}\right)^2}{2\left(\sigma^2_{\theta'} + \delta\right)}} \\ &= \frac{1}{\sqrt{2\pi\left(\sigma^2_{\theta'} + \delta\right)}} \cdot \exp^{\frac{-\left(\kappa\sigma_{\theta'} + \eta\right)^2}{2\left(\sigma^2_{\theta'} + \delta\right)}}\end{aligned}$$

Taking logarithm on both sides,

$$\begin{aligned}\log \pi_{\theta'}(a' \mid s') &= -\frac{1}{2}\log\left(2\pi(\sigma^2_{\theta'} + \delta)\right) + \log\left(\exp^{\frac{-\left(\kappa\sigma_{\theta'} + \eta\right)^2}{2\left(\sigma^2_{\theta'} + \delta\right)}}\right) \\ &= -\frac{1}{2}\log\left(2\pi(\sigma^2_{\theta'} + \delta)\right) - \frac{\left(\kappa\sigma_{\theta'} + \eta\right)^2}{2(\sigma^2_{\theta'} + \delta)}. \qquad (16)\end{aligned}$$

Since $\kappa\sigma_{\theta'} + \eta \sim \mathcal{N}(0, \sigma^2_{\theta'} + \delta)$, then $\frac{\kappa\sigma_{\theta'} + \eta}{\sqrt{(\sigma^2_{\theta'} + \delta)}} \sim \mathcal{N}(0,1)$. Quite clearly $\frac{\left(\kappa\sigma_{\theta'} + \eta\right)^2}{\left(\sigma^2_{\theta'} + \delta\right)} \sim \chi^2_1$, which implies that $\mathbb{E}\left[\frac{\left(\kappa\sigma_{\theta'} + \eta\right)^2}{\left(\sigma^2_{\theta'} + \delta\right)}\right] = 1$. Taking expectation on both sides of Equation 16,

$$\begin{aligned}\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')] &= -\frac{1}{2}\mathbb{E}\left[\log\left(2\pi(\sigma^2_{\theta'} + \delta)\right)\right] - \frac{1}{2}\mathbb{E}\left[\frac{\left(\kappa\sigma_{\theta'} + \eta\right)^2}{(\sigma^2_{\theta'} + \delta)}\right] \\ &= -\frac{1}{2}\log\left(2\pi(\sigma^2_{\theta'} + \delta)\right) - \frac{1}{2}.\end{aligned}$$

Finally we have,

$$\boxed{\mathbb{E}_{\text{OPAC}}[\log \pi_{\theta'}(a' \mid s')] = -\frac{1}{2}\log\left(2\pi(\sigma^2_{\theta'} + \delta)\right) - \frac{1}{2}}$$

The expected value of the entropy has been calculated using $a'$ according to Algorithm 1. We sampled $a'$ from $\pi_{\theta'}(\cdot \mid s')$ and added a Gaussian noise. However, in practice, we compute the entropy using only $a'$ and without the noise.

### A.4 CALCULATING THE EXPECTED ENTROPY OF SAC

The target-update equation of SAC is

$$y^{\text{SAC}}(r, s') \leftarrow r + \gamma \left( \min_{i=1,2} Q_{\phi_i'}(s', a') - \alpha \log \pi_{\theta'}(a' \mid s') \right).$$

$\pi_{\theta'}$ is the target Gaussian policy network with parameters $\theta'$. From the re-parameterization trick used in Haarnoja et al. (2018c) we know

$$a' = f_{\theta'}(\kappa, s') = \mu_{\theta'}(s') + \kappa \sigma_{\theta'}(s'), \text{ where } \kappa \sim \mathcal{N}(0, 1).$$

$\mu_{\theta'}$ and $\sigma_{\theta'}$ are obtained from $\pi_{\theta'}$ itself. Because $s'$ is already known and $\mu_{\theta'}$ and $\sigma_{\theta'}$ are functions of it, we omit unnecessary writing of $s'$ for simplicity. It is easily seen that

$$\kappa \sim \mathcal{N}(0, 1) \implies \kappa \sigma_{\theta'} \sim \mathcal{N}(0, \sigma_{\theta'}^2).$$

Also,

$$a' = \mu_{\theta'} + \kappa \sigma_{\theta'} \implies \kappa \sigma_{\theta'} = a' - \mu_{\theta'}.$$

Since $\pi_{\theta'}$ is a Gaussian policy network, then $\pi_{\theta'}(a' \mid s') \sim \mathcal{N}(\mu_{\theta'}, \sigma_{\theta'}^2)$ and it can be thought of as the probability distribution function of $\mathcal{N}(\mu_{\theta'}, \sigma_{\theta'}^2)$. Therefore,

$$\begin{aligned}
\pi_{\theta'}(a' \mid s') &= \frac{1}{\sigma_{\theta'}\sqrt{2\pi}} \cdot \exp^{\frac{-(a' - \mu_{\theta'})^2}{2\sigma_{\theta'}^2}} \\
&= \frac{1}{\sigma_{\theta'}\sqrt{2\pi}} \cdot \exp^{\frac{-(\mu_{\theta'} + \kappa\sigma_{\theta'} - \mu_{\theta'})^2}{2\sigma_{\theta'}^2}} \\
&= \frac{1}{\sigma_{\theta'}\sqrt{2\pi}} \cdot \exp^{\frac{-(\kappa\sigma_{\theta'})^2}{2\sigma_{\theta'}^2}} \\
&= \frac{1}{\sigma_{\theta'}\sqrt{2\pi}} \cdot \exp^{\frac{-\kappa^2}{2}}.
\end{aligned}$$

Taking logarithm on both sides,

$$\begin{aligned}
\log \pi_{\theta'}(a' \mid s') &= \log \left( \frac{1}{\sigma_{\theta'}\sqrt{2\pi}} \right) + \log \left( \exp^{\frac{-\kappa^2}{2}} \right) \\
&= -\log \left( \sigma_{\theta'}\sqrt{2\pi} \right) - \frac{\kappa^2}{2}.
\end{aligned}$$

Taking expectation on both sides,

$$\mathbb{E}_{\text{SAC}}[\log \pi_{\theta'}(a' \mid s')] = \mathbb{E}\left[ -\log \left( \sigma_{\theta'}\sqrt{2\pi} \right) \right] - \mathbb{E}\left[ \frac{\kappa^2}{2} \right].$$

$\because \kappa \sim \mathcal{N}(0, 1) \implies \kappa^2 \sim \chi_1^2 \implies \mathbb{E}[\kappa^2] = 1$. Therefore,

$$\boxed{\mathbb{E}_{\text{SAC}}[\log \pi_{\theta'}(a' \mid s')] = -\log \left( \sigma_{\theta'}\sqrt{2\pi} \right) - \frac{1}{2}} \tag{17}$$

### A.5 CALCULATING THE EXPECTED ERROR OF SAC, TD3, AND TADD

Let $Q_{\phi_i'}$ denote the i-th target critic network with parameters $\phi_i'$. Similarly, let $\pi_{\theta'}$ denote the target Gaussian policy network with parameters $\theta'$. The target-update equations of SAC, TD3, and TADD are as follows

$$\begin{aligned}
y^{\text{SAC}}(r, s') &\leftarrow r + \gamma \left( \min_{i=1,2} Q_{\phi_i'}(s', a') - \alpha \log \pi_{\theta'}(a' \mid s') \right) \\
y^{\text{TD3}}(r, s') &\leftarrow r + \gamma \min_{i=1,2} Q_{\phi_i'}(s', a') \\
y^{\text{TADD}}(r, s') &\leftarrow r + \gamma \left( \beta \min_{i=1,2} Q_{\phi_i'}(s', a') + (1 - \beta) \frac{1}{K} \sum_{k=1}^{K} [Q_{\phi_3'}(s', a')]^k \right).
\end{aligned}$$

Now, let $Z_{\text{SAC}}$, $Z_{\text{TD3}}$, and $Z_{\text{TADD}}$ respectively denote the error induced in the target-update equations of SAC, TD3, and TADD due to function approximation. Then let $\mathbb{E}[Z_{\text{SAC}}]$, $\mathbb{E}[Z_{\text{TD3}}]$, and $\mathbb{E}[Z_{\text{TADD}}]$ denote the expected error in the Q-value estimation of SAC, TD3, and TADD respectively. Using the same type of calculations as in Section 6.2 and Section 7 we have

$$\mathbb{E}[Z_{\text{SAC}}] = \mathbb{E}[Z^2_{\min}] - (\alpha\gamma)\mathbb{E}_{\text{SAC}}[\log \pi_{\theta'}(a' \mid s')]$$
$$\mathbb{E}[Z_{\text{TD3}}] = \mathbb{E}[Z^2_{\min}]$$
$$\mathbb{E}[Z_{\text{TADD}}] = \beta\mathbb{E}[Z^2_{\min}] + (1-\beta)\lambda$$

In $\mathbb{E}[Z_{\text{TADD}}]$, $\lambda$ ($> 0$) denotes the overestimation bias of $Q_{\phi'_3}$ and $\beta \in (0, 1)$ is the weight of the double-critics (Wu et al., 2020). Substituting the values of $\mathbb{E}[Z^2_{\min}]$ (from Section 6.2) and $\mathbb{E}_{\text{SAC}}[\log \pi_{\theta'}(a' \mid s')$ (from Equation 17) in the above equations we get

$$\boxed{\mathbb{E}[Z_{\text{SAC}}] = -\frac{1}{3}\epsilon\gamma + \alpha\gamma \cdot \log\left(\sigma_{\theta'}\sqrt{2\pi}\right) + \frac{\alpha\gamma}{2}}$$

$$\boxed{\mathbb{E}[Z_{\text{TD3}}] = -\frac{1}{3}\epsilon\gamma}$$

$$\boxed{\mathbb{E}[Z_{\text{TADD}}] = -\frac{1}{3}\epsilon\gamma\beta + (1-\beta)\lambda}$$

### A.6 TABLE CORRESPONDING TO FIGURE 3

Table 1: Maximum reward obtained by SAC, TD3, and TADD in the six MuJoCo continuous control environments. The $\pm$ corresponds to one standard deviation. To maintain margins on both sides of the paper, the maximum rewards obtained by OPAC is shown separately in Table 2.

| **Environment** | **SAC** | **TD3** | **TADD** |
|---|---|---|---|
| Ant-v2 | $5580.93 \pm 307.76$ | $5817.13 \pm 138.28$ | $6129.01 \pm 218.52$ |
| HalfCheetah-v2 | $14656.11 \pm 138.25$ | $12029.47 \pm 182.26$ | $12416.0 \pm 138.06$ |
| Hopper-v2 | $3354.10 \pm 17.84$ | $2858.70 \pm 18.65$ | $2954.54 \pm 57.33$ |
| Humanoid-v2 | $6835.31 \pm 17.66$ | $193.09 \pm 7.79$ | $167.09 \pm 7.14$ |
| InvertedPendulum-v2 | $\mathbf{1000.00 \pm 0.00}$ | $\mathbf{1000.00 \pm 0.00}$ | $\mathbf{1000.00 \pm 0.00}$ |
| Walker2d-v2 | $5733.03 \pm 49.22$ | $5647.76 \pm 52.52$ | $5391.95 \pm 50.91$ |

Table 2: Maximum reward obtained by OPAC in the six MuJoCo continuous control environments. The $\pm$ corresponds to one standard deviation.

| **Environment** | **OPAC** |
|---|---|
| Ant-v2 | $\mathbf{7097.45 \pm 90.80}$ |
| HalfCheetah-v2 | $\mathbf{15356.27 \pm 149.61}$ |
| Hopper-v2 | $\mathbf{3610.51 \pm 26.80}$ |
| Humanoid-v2 | $\mathbf{7710.47 \pm 32.70}$ |
| InvertedPendulum-v2 | $\mathbf{1000.00 \pm 0.00}$ |
| Walker2d-v2 | $\mathbf{7008.71 \pm 50.41}$ |

The numbers in the columns of Table 1 and Table 2 denote the maximum return averaged over five trials for an algorithm. Ideally, the entries for OPAC should have been present in Table 1 itself.

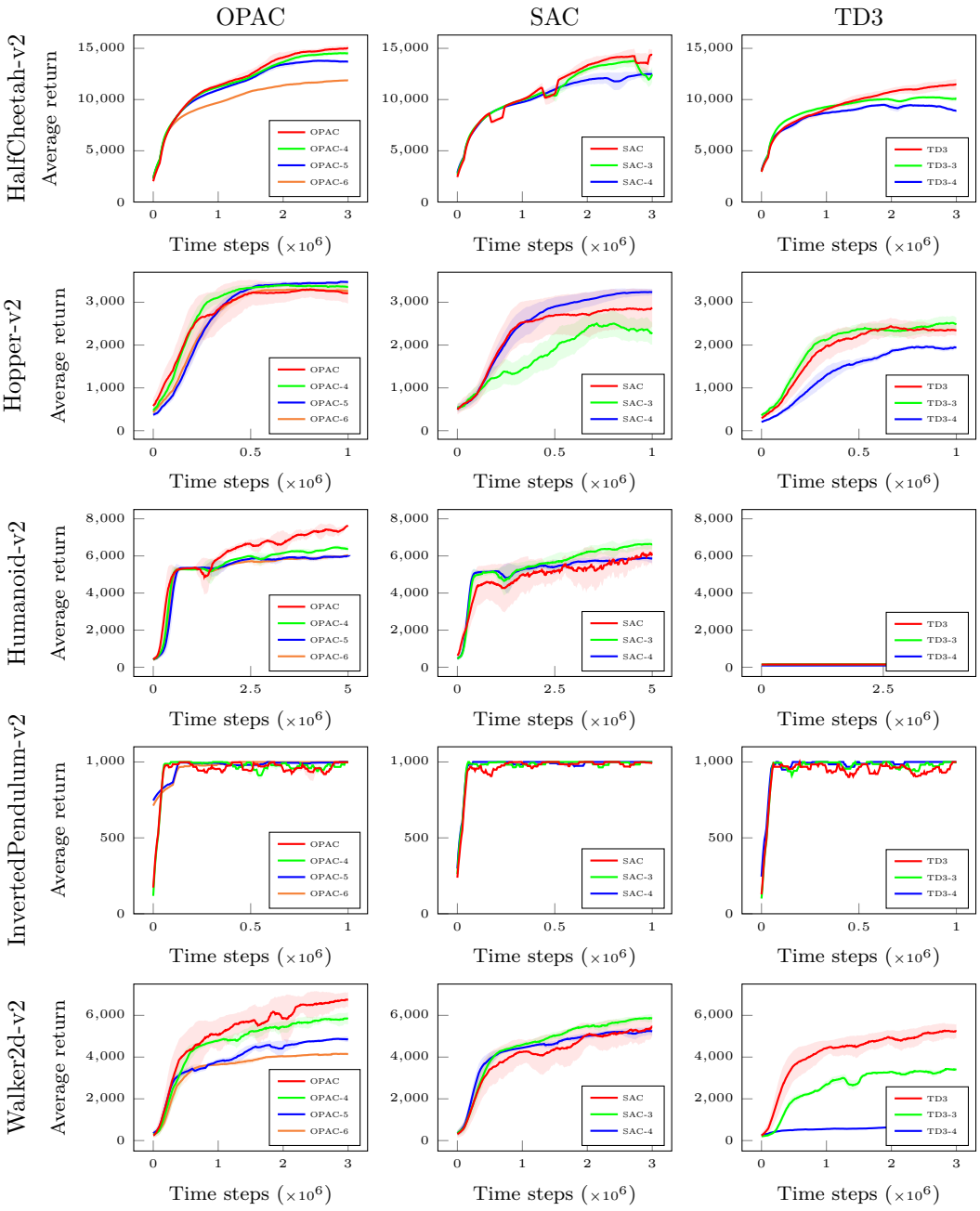



Figure 6: Performance comparison of OPAC, SAC, and TD3 with different number of critics in the remaining five MuJoCo environments. The shaded regions correspond to one standard deviation.

However, a split was necessary to maintain the margins on both sides of the paper. Therefore, the OPAC entries were placed in Table 2. On observing the tables and the corresponding columns, it is seen that OPAC attains the highest reward in all the environments. That's why each entry in Table 2 is boldfaced.

### A.7 Additional plots for Ablation on Critics

As already mentioned in Section 8.2, increasing the number of critics degrades the performance of the algorithms. Figure 6 conforms to this fact. It is evident from the plots that the vanilla versions of

OPAC, SAC, and TD3 deliver consistently in the environments. However, in some cases, the variants (like SAC-3, OPAC-4) marginally perform better than the respective vanilla versions. Nevertheless, this improvement is nothing significant.

Therefore, increasing the number of critics brings about more underestimation, which is responsible for the sub-optimal performance of the variants of OPAC, SAC, and TD3.

### A.8 TABLE CORRESPONDING TO FIGURE 4 AND FIGURE 6

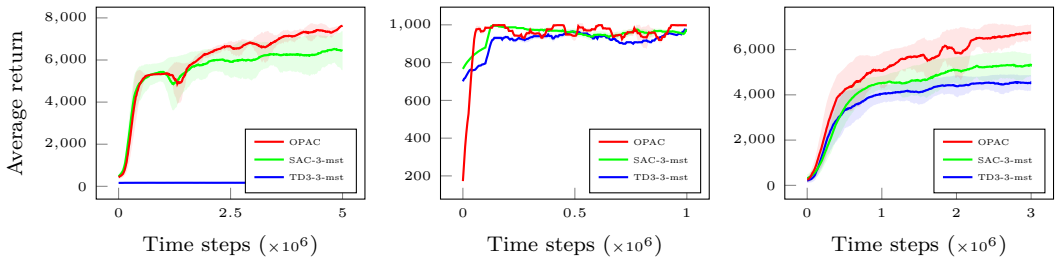
Table 3: The maximum mean return (averaged over five trials) of the different SAC, TD3, and OPAC versions in three MuJoCo environments. The $\pm$ corresponds to one standard deviation. The maximum reward in an environment has been boldfaced.

| Algorithm | Ant-v2 | Hopper-v2 | Walker2d-v2 |
|---|---|---|---|
| SAC | $5580.93 \pm 307.76$ | $3354.10 \pm 17.84$ | $5733.03 \pm 49.22$ |
| SAC-3 | $4751.11 \pm 112.25$ | $3317.38 \pm 248.84$ | $6035.48 \pm 35.59$ |
| SAC-4 | $5091.44 \pm 54.87$ | $3316.90 \pm 11.93$ | $5423.74 \pm 23.80$ |
| TD3 | $5817.13 \pm 138.28$ | $2858.70 \pm 18.65$ | $5647.76 \pm 52.52$ |
| TD3-3 | $4309.67 \pm 76.56$ | $2908.60 \pm 14.84$ | $3585.04 \pm 11.02$ |
| TD3-4 | $3894.86 \pm 33.49$ | $2091.47 \pm 3.52$ | $1103.03 \pm 341.61$ |
| OPAC | $\mathbf{7097.45 \pm 90.80}$ | $\mathbf{3610.51 \pm 26.80}$ | $\mathbf{7008.71 \pm 50.41}$ |
| OPAC-4 | $6166.51 \pm 58.75$ | $3466.41 \pm 5.08$ | $6081.18 \pm 22.64$ |
| OPAC-5 | $5241.13 \pm 47.31$ | $\mathbf{3525.72 \pm 4.28}$ | $4984.11 \pm 14.74$ |
| OPAC-6 | $4713.99 \pm 38.40$ | $3369.56 \pm 5.55$ | $4273.17 \pm 20.48$ |

Table 3 contains a comparison between the maximum rewards obtained by the variants (based on the number of critics) of SAC, TD3, and OPAC. The boldfaced value in each column of the table indicates the maximum reward obtained by an algorithm in Ant-v2, Hopper-v2, and Walker2d-v2.

The row-wise entries inside the table indicate that OPAC achieves the highest reward in all the environments. To put it differently, OPAC performs consistently well in all environments.

### A.9 ADDITIONAL PLOTS FOR ABLATION ON TARGET-UPDATE RULE




(a) Humanoid-v2 (b) InvertedPendulum-v2 (c) Walker2d-v2

Figure 7: Comparing the performance of OPAC with SAC-3-mst and TD3-3-mst in the remaining three MuJoCo environments. The shaded regions correspond to one standard deviation.

Figure 7 shows the plots of OPAC, SAC-3-mst, and TD3-3-mst on the remaining three MuJoCo environments. Quite clearly, the usage of $\mathrm{mst}(\cdot)$ boosts the performance of SAC and TD3, but this boost is insufficient to outperform OPAC. In additon to the theoretical analysis given in Section 6.2, the plots in Figure 5 and Figure 7 also demonstrate that $\mathrm{mst}(\cdot)$ is a better target-update strategy than $\min(\cdot)$.

### A.10 LIST OF HYPER-PARAMETERS

Table 4 lists the value of the OPAC, SAC, TD3, and TADD hyper-parameters used to generate the results of Section 8, i.e., comparative evaluation and ablation study.

Table 4: Hyper-parameter choices for OPAC, SAC, TD3, and TADD.

| **Hyperparameter** | **OPAC** | **SAC** | **TD3** | **TADD** |
|---|---|---|---|---|
| Optimizer | Adam | Adam | Adam | Adam |
| Learning Rate | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| Beta ($\beta$) | N/A | N/A | N/A | 0.95 |
| Target Update Rate ($\tau$) | $5 \times 10^{-3}$ | $5 \times 10^{-3}$ | $5 \times 10^{-3}$ | $5 \times 10^{-3}$ |
| Target Update Interval ($D$) | 2 | 1 | 2 | 2 |
| Policy Update Interval ($D$) | 2 | 1 | 2 | 2 |
| Replay Buffer Size | $10^6$ | $10^6$ | $10^6$ | $10^6$ |
| No. of hidden layers | 2 | 2 | 2 | 2 |
| Hidden units per layer | 512 | 256 | $(400, 300)$ | $(400, 300)$ |
| Batch Size | 256 | 256 | 100 | 100 |
| Nonlinearity | ReLU | ReLU | ReLU | ReLU |
| Discount Factor ($\gamma$) | 0.99 | 0.99 | 0.99 | 0.99 |
| Reward Scaling | 1 | 1 | 1 | 1 |
| Noise Clipping | $(-0.5, 0.5)$ | N/A | $(-0.5, 0.5)$ | $(-0.5, 0.5)$ |
| Policy Noise | $\mathcal{N}(0, 0.2)$ | N/A | $\mathcal{N}(0, 0.2)$ | $\mathcal{N}(0, 0.2)$ |
| Exploration Noise | $\mathcal{N}(0, 0.1)$ | N/A | $\mathcal{N}(0, 0.1)$ | $\mathcal{N}(0, 0.1)$ |
| Gradient Steps | 1 | 1 | 1 | 1 |
| Critic Regularisation | None | None | None | None |
| Actor Regularisation | None | None | None | None |

### A.11 SYSTEM SPECIFICATIONS

Table 5 lists the GPU's specifications that have been used to execute the codes.

Table 5: GPU specifications.

| **Attribute** | **Value** |
|---|---|
| GPU Name | GeForce RTX 2080 Ti |
| Count | 4 |
| Width | 64-bit |
| Clock Speed | 33 MHz |
| RAM | 32 GB |
| Operating System | Ubuntu 20.04.1 LTS |
| Kernel | Linux 5.4.0-45-generic |