Opportunistic Actor-Critic with Clipped Triple Q-learning

Srinjoy Roy[1*], Saptam Bakshi[1] and Tamal Maharaj[2]

[1]Institute for Advancing Intelligence (IAI), TCG Centres for Research and Education in Science and Technology (TCG CREST), Salt Lake, Kolkata, 700091, West Bengal, India.
[2]Department of Computer Science, Ramakrishna Mission Vivekananda Educational and Research Institute, Belur, Howrah, 711202, West Bengal, India.

*Corresponding author(s). E-mail(s): srinjoy.roy@tcgcrest.org;
Contributing authors: saptam.bakshi@tcgcrest.org;
tamal@gm.rkmvu.ac.in;

Abstract

Two of 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 achieved efficient continuous control. SAC managed the issues of sample complexity and hyper-parameter convergence brittleness. In complicated tasks, it surpassed all state-of-the-art algorithms, including TD3, although TD3 generated moderate outcomes in all contexts. However, because of the Gaussian nature of its policy, SAC suffers from inefficient exploration, resulting in questionable performance in simpler problems. We introduce Opportunistic Actor-Critic (OPAC), an ensemble model-free deep RL algorithm that performs well in easy and hard tasks alike. OPAC combines features of TD3 and SAC along with employing Clipped Triple Q-learning to fine-tune the Q-value estimates. We have systematically evaluated OPAC on MuJoCo environments and the results indicate the consistent superior performance of OPAC in general. It outperforms TD3 and SAC in terms of average reward accumulated over time on all environments.

Keywords: Model-free; Deep RL; Actor-Critic; SAC; TD3.

1 Introduction

Model-free deep reinforcement learning (RL) has proven to be enormously successful in a multitude of fields like, gaming [1–3], robotic control [4, 5], decision making [6], and many more. Despite achieving overwhelming performance in these tasks, the application of 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. The most popular and robust model-free deep RL methods are often based on actor-critic architectures, which first appeared in [7].

One important drawback of the traditional actor-critic methods is that they tend to over-estimate the Q-value of a state-action pair. Owing to this, Deep Deterministic Policy Gradient (DDPG) [8] based algorithms fail to deliver promising results on continuous control domains. Twin Delayed Deep Deterministic Policy Gradient (TD3) [9] was developed to address this issue. It introduced multiple novel features to address over-estimation as well as stabilisation: Clipped Double Q-learning, delayed policy updates, and target policy smoothing. Similar to DDPG, it is an off-policy algorithm that trains a deterministic policy and, hence, can reuse past samples using an experience replay memory. But the problem with TD3 is that it requires a lot of hyper-parameter tuning to converge to the optimal policy.

Soft Actor-Critic (SAC) [10] makes use of the maximum entropy framework. Like TD3, SAC also uses Clipped Double Q-learning but trains a stochastic policy. It was shown to outperform prior state-of-the-art methods (including TD3) in terms of performance and sample efficiency. But SAC proved to be brittle with respect to the tuning of the temperature parameter α. To combat this brittleness, an automated temperature-tuning based SAC was developed in [11]. This version of SAC eliminated the need for per-task hyper-parameter tuning and also demonstrated stability. The gap between DDPG style approaches and stochastic policy optimisation was bridged by SAC. However, due to the Gaussian nature of its policy, SAC suffers from poor exploration, resulting in borderline performance in relatively simpler continuous control environments [12].

Under-estimation, as opposed to over-estimation, has not been investigated too deeply. Taking the minimum of the Q-values, i.e., the critic outputs, results in pessimistic under-exploration as mentioned in the paper of Optimistic Actor-Critic (OAC) [12]. In order to deal with the curse of pessimism caused by the minimum operation, it employs an optimistic estimate of the Q-values by taking their maximum. Another approach to counter the under-estimation bias was given in Triplet-Average Deep Deterministic policy gradient (TADD) [13]. It uses three critics and takes a convex combination of the minimum of the first two Q-values and the third Q-value. Additionally, to reduce variance it uses the arithmetic mean of the third Q-value of the last K time steps. One

glaring shortcoming of this approach is its dependence on the convex combination coefficient β which is an environment-dependent hyper-parameter and balances the over-estimation and under-estimation bias.

The major contribution of our work is to have an algorithm that can perform well in easy and hard tasks with the same efficiency without any additional environment-specific hyper-parameters. In this context, we introduce Opportunistic Actor-Critic (OPAC), a model-free deep RL algorithm that puts the central features of TD3 and SAC under one roof to retain their respective benefits. It also utilises three critics contrary to TD3 and SAC, both of which employ two critics to estimate the Q-values. This design choice introduces the idea of voting for fine-tuning the value updates (more about it in Section 5). We present empirical results exhibiting OPAC's superiority compared to TD3 and SAC.

The paper is organised as follows. Section 2 gives a brief overview of Reinforcement Learning and the maximum entropy framework. Section 3, 4, and 5 collectively introduces the proposed algorithm. Section 3 lists the key features of our algorithm. Section 4 briefly presents the mathematical details of the algorithm. Section 5 motivates and illustrates one of the key aspects of the algorithm: Clipped Triple Q-learning. Section 6 wraps up the presentation and presents the final algorithm. Section 7 presents the detailed experimental results followed by an ablation study. The paper is concluded with some closing remarks with Section 8.

2 Preliminaries

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

2.1 Reinforcement Learning

Markov Decision Processes (MDPs) are generally defined by the 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, p represents the state transition probabilities and r represents the reward function. $\mathcal{S}$ and $\mathcal{A}$ are assumed to be continuous and the state transition probability $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents 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 π that specifies the action $\pi(s)$ to choose when in state s. Reinforcement learning (RL) 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 the above objective.

2.2 Maximum Entropy Reinforcement Learning

The maximum entropy objective [14] extends the normal RL objective by including an entropy term, so that the optimal policy also strives to maximise its entropy at each visited state:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{\infty} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))],$$

where α 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. Entropy is the 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,

$$\mathcal{H}(\mathcal{P}) = \mathbb{E}_{x\sim\mathcal{P}}[-\log \mathcal{P}(x)].$$

3 Features of OPAC

Before viewing the mathematics behind OPAC, we discuss its constituent features. First, we consider the features from SAC that are included in OPAC and the reasons behind the inclusion.

- **Maximum Entropy Framework**: The maximum entropy framework has been used in a number of areas [15–18]. The concept of maximising entropy as well as the expected total reward is central to the maximum entropy framework. In this framework, the policy is rewarded for broadening its boundaries while rejecting sub-optimal actions. The policy is also capable of capturing a wide range of near-optimal behaviour, and it assigns many actions equal probabilities when they appear to be equally desirable. When compared to approaches that optimise the standard RL objective, it has been discovered that the maximum entropy framework greatly improves learning speed.
- **Automatic Entropy Adjustment**: Since SAC is very sensitive to α [11], fixing its value apriori in the maximum entropy framework is not a good approach. In practise, determining the ideal temperature is very difficult as the entropy fluctuates unpredictably across tasks during training. Therefore, OPAC's temperature is automatically adjusted in the same manner as SAC.

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

- **Target Networks**: In deep RL, target networks proved to be key to stabilise the training [19]. Deep neural networks are great function approximators, but they 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. OPAC, like SAC and TD3, leverages target networks for both actors and critics.
- **Delayed Policy Updates**: By updating the actor at a lesser frequency than the critic, the risk of error is reduced before a policy update is performed. 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.
- **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.

We are now left discussing only one feature of OPAC that is unique in it.

- **Clipped Triple Q-learning**: Both TD3 and SAC make use of Clipped Double Q-learning which means that they consider minimum of the two Q-values while computing the shared Q-target. This is essentially a pessimistic approach. OPAC uses clipped triple Q-learning instead. But it considers the mean of the smallest two Q-values rather than simply taking the minimum of three. For example, if Q_1, Q_2, and Q_3 are the output of the three critics and (Q_1, Q_2) are the smallest two Q-values, then OPAC will consider the mean of (Q_1, Q_2) for computing the shared Q-target. This approach is neither pessimistic nor optimistic [12] where either the minimum or the maximum of two Q-values is considered. Rather, OPAC takes the opportunity to improve its estimation by adding a new critic and choosing the Q-value that is always more than or equal to the minimum of all the three Q-values. Our empirical results show that this decision indeed performs better than other approaches.

4 Opportunistic Actor-Critic

Our algorithm can be mathematically derived from Soft Policy Iteration. It was introduced and fully derived in [10, 11]. Since our derivation is inspired by that of SAC, both have many similarities. Therefore, we will skip the repetitive details. To translate the Soft Policy Iteration into the continuous setting, the soft Q-function and the policy both need to be approximated by neural networks. We alternate between optimising the soft Q-function and policy network by stochastic gradient descent. This is how we will build our algorithm of OPAC. Let $Q_\phi(s_t, a_t)$ and $\pi_\theta(a_t \mid s_t)$ denote the soft Q-function and the policy with parameters ϕ and θ respectively. ϕ can be trained to minimise the

soft Bellman-error as follows,

$$J_Q(\phi) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{R}}\left[\frac{1}{2}(Q_\phi(s_t,a_t) - (r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\phi_{\text{target}}}(s_{t+1})]))^2\right]. \tag{1}$$

Putting in the value function parameters $V(s_t) = \mathbb{E}_{a_t\sim\pi(\cdot|s_t)}[Q(s_t,a_t) - \alpha\log\pi(a_t \mid s_t)]$ and differentiating Equation (1) w.r.t ϕ, we obtain the update rule for stochastic gradient descent,

$$\begin{aligned}\nabla_\phi J_Q(\phi) = \nabla_\phi Q_\phi(s_t,a_t)\big(Q_\phi(s_t,a_t) - (r(s_t,a_t) \\ + \gamma Q_{\phi_{\text{target}}}(s_{t+1},a_{t+1}) - \alpha\log\pi_{\theta_{\text{target}}}(a_{t+1} \mid s_{t+1}))\big),\end{aligned} \tag{2}$$

where ϕ_{target} and θ_{target} denote the parameters of the target Q-function and that of target policy respectively. θ can be learned by minimising the KL-divergence in the policy improvement equation (given in [11]) as follows,

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

Therefore we need to compute,

$$\nabla_\theta\mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}[\alpha\log\pi_\theta(a_t \mid s_t) - Q_\phi(s_t,a_t)].$$

As $Q_\phi(s_t,a_t)$ does not directly depend on θ, its gradient cannot be computed over θ. Since we parameterise the policy as a Gaussian distribution with parameters μ_θ and σ_θ, we have

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

As we use neural networks, it is convenient to use the reparameterization trick for obtaining the gradient. Setting $a_t = f_\theta(\epsilon_t,s_t) = \mu_\theta(s_t) + \epsilon_t\sigma_\theta(s_t)$ and combining Equations (3) and (4), we get

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

whose gradient w.r.t θ can be obtained by,

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

An off-policy algorithm for OPAC has been constructed but with a particular temperature i.e., the value of α was fixed. We have already discussed the need for automating α and that we are going to do it the same way as SAC. The standard maximum entropy learning problem for OPAC can be reformulated as follows: while aiming to maximise the expected return, the policy should

also satisfy a minimum entropy constraint:

$$\max_{\pi_0 \ldots \pi_T} \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t)\right], \quad \text{s.t. } \forall t, \mathcal{H}(\pi_t) \geq H_0, \tag{7}$$

where H_0 is a predefined minimum policy entropy threshold. The optimal value of α can be learned by minimising the objective function,

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t}[-\alpha \log \pi_t(a_t \mid s_t) - \alpha H_0]. \tag{8}$$

5 Clipped Triple Q-learning

Taking the minimum of the Q-values is the crux of Clipped Double Q-learning [9]. As mentioned earlier, OPAC maintains three action-value functions and considers the mean of the smallest two Q-values instead of the minimum of three. This is presented formally in Theorem 1. The same proof technique has been used for showing the convergence of many prior Q-learning based algorithms like Double Q-learning [20], Clipped Double Q-learning [9], and TADD [13].

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 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 $\alpha_t(s, a) = 0$, $\forall (s, a) \neq (s_t, a_t)$.

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

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

We now present the proof of Theorem 1 under deterministic policies. To this end, we first include a lemma from [21]. It originally appears as a proposition in [22] which was further generalised into the following lemma.

Lemma 2 Let $\|\cdot\|$ denote the maximum norm. Consider a stochastic process (ζ_t, Δ_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 σ-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 for $t = 0, 1, \ldots$. Assume that the following hold:

1. The set X is finite.
2. $\zeta_t(x_t) \in [0, 1]$, $\sum_t \zeta_t(x_t) = \infty$, $\sum_t (\zeta_t(x_t))^2 < \infty$ 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. $\text{Var}[F_t \mid P_t] \leq K(1 + \kappa\|\Delta_t\|)^2$, where K is some constant.

Then Δ_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 2 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 condition 1 and 4 of lemma 2 holds by the condition (ii) of the theorem. Lemma condition 2 holds by the theorem condition (iv) along with our selection of $\zeta_t = \alpha_t$. Following the same notation and argument as [9] 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 $\mathbb{E}\left[F_t^Q \mid P_t\right] \leq \gamma\|\Delta_t\|$ is known to be true due to Bellman operator being a *contraction mapping*. This implies condition 3 of lemma 2 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 Δ_t^{BA} and Δ_t^{BC} needs to converge to 0 with probability 1. The convergence of Δ_t^{BA} is fairly straightforward and has been laid out in [9]. 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, Δ_t^{BC} converges to 0. These imply we have fulfilled the condition 3 of lemma 2, 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. □

6 Algorithm

OPAC is described in Algorithm 1. It makes use of three soft Q-functions and considers mean of the smallest two Q-values for calculating the shared Q-target. Policy is updated by gradient ascent once every D iterations while the critics are updated by stochastic gradient descent in every iteration. OPAC

Algorithm 1 Opportunistic Actor-Critic (OPAC)

Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, ϕ_3 and an empty replay buffer $\mathcal{R}$.
Initialize target network parameters: $\theta' \leftarrow \theta$, $\phi'_i \leftarrow \phi_i$ $(i = 1, 2, 3)$.
Populate the replay buffer $\mathcal{R}$.
for $t = 1, \ldots, T$ **do**
  Sample a batch of transitions, $\mathcal{B} = \{(s, a, r, s')\}$ from $\mathcal{R}$.
  Compute next action: $a' \leftarrow \tilde{a} + \epsilon$, $\tilde{a} \sim \pi_{\theta'}(\cdot \mid s')$, $\epsilon \sim \mathcal{N}(0, \sigma)$.
  $\mathcal{X} \leftarrow$ mean of the smallest two Q-values.
  $y(r, s') \leftarrow r + \gamma\,(\mathcal{X} - \alpha \log \pi_{\theta'}(a' \mid s'))$.
  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 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 adjust α:

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

  end if
end for

consists of eight neural networks–two for the actor target and actor model, three for three critic targets and the other three for three critic models. The target networks are updated by polyak-averaging once every D iterations. Gaussian noise is added to the target actions a' but while implementing, the value is clipped to keep the target close to the original action.

7 Experiments and Results

Since TD3 and SAC are currently two of the best model-free deep RL algorithms, we limit our comparison of the performance of OPAC with only TD3 and SAC. Different variants of OPAC, SAC, and TD3 have been used throughout the section. Table 1 contains a detailed list of acronyms used for the different variants of OPAC, SAC, and TD3. Vanilla versions of these algorithms have been boldfaced in the table. Acronyms differ on the number of critics and on the strategy used to compute the shared Q-target from all the Q-values. All the algorithms have been tested in six MuJoCo continuous control tasks [23] interfaced through OpenAI Gym [24]. The algorithms were run in Hopper-v2 and InvertedPendulum-v2 for one million time steps whereas in Ant-v2, HalfCheetah-v2, and Walker2d-v2 for three million time steps. We run the algorithms for five million time steps in Humanoid-v2, arguably the most challenging environment.

This section is divided into two subsections: comparative evaluation and ablation study. Comparative evaluation aims to compare the performance of OPAC with the vanilla versions of SAC and TD3 while ablation study focuses

Table 1: Acronyms for different algorithms used in the text. Vanilla versions of these algorithms have been boldfaced in the table. The 'Strategy Used' column denotes the strategy used to compute the shared Q-target from all the Q-values.

Algorithm	No. of Critics	Strategy Used	Acronym
TD3	**2**	**Minimum of two critics**	**TD3**
	3	Minimum of three critics	TD3-3
	4	Minimum of four critics	TD3-4
SAC	**2**	**Minimum of two critics**	**SAC**
	3	Minimum of three critics	SAC-3
	4	Minimum of four critics	SAC-4
OPAC	2	Minimum of two critics	OPAC-2
	3	**Mean of smallest two critics**	**OPAC**
	3	Median of three critics	OPAC-3-Med
	4	Mean of smallest two critics	OPAC-4-Mean
	4	Median of four critics	OPAC-4-Med
	5	Mean of smallest two critics	OPAC-5-Mean
	6	Mean of smallest two critics	OPAC-6-Mean

on understanding the effect of varying the number of critics and changing the aggregation strategy.

Table 2: OPAC's performance compared to Vanilla TD3 and Vanilla SAC. The numbers denote the maximum return averaged over five trials. The $\pm$ corresponds to one standard deviation. Maximum reward in an environment has been boldfaced.

Environment \ Algorithm	SAC	TD3	OPAC
Ant-v2	5580.93 ± 307.76	5817.13 ± 138.28	$\mathbf{7097.45 \pm 90.80}$
HalfCheetah-v2	14656.11 ± 138.25	12029.47 ± 182.26	$\mathbf{15356.27 \pm 149.61}$
Hopper-v2	3354.10 ± 17.84	2858.70 ± 18.65	$\mathbf{3610.51 \pm 26.80}$
Humanoid-v2	6835.31 ± 17.66	193.09 ± 7.79	$\mathbf{7710.47 \pm 32.70}$
InvertedPendulum-v2	$\mathbf{1000.00 \pm 0.00}$	$\mathbf{1000.00 \pm 0.00}$	$\mathbf{1000.00 \pm 0.00}$
Walker2d-v2	5733.03 ± 49.22	5647.76 ± 52.52	$\mathbf{7008.71 \pm 50.41}$

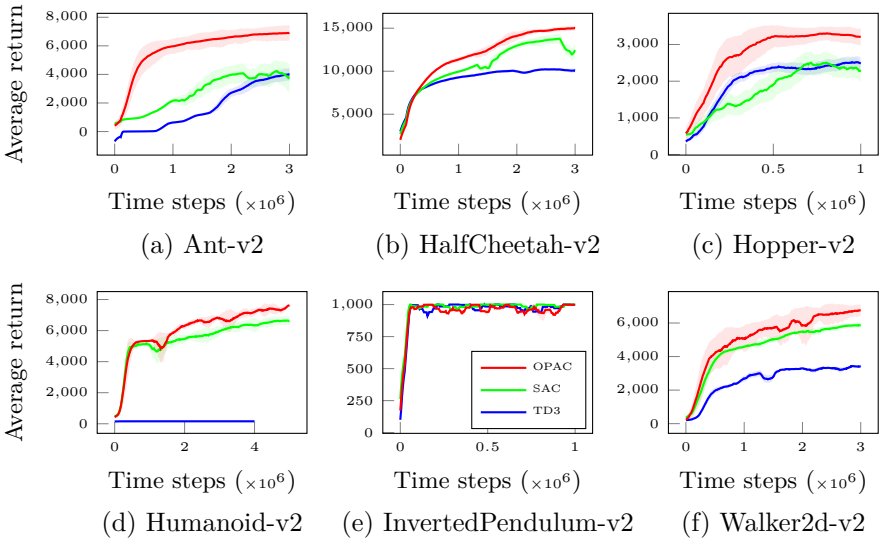


Fig. 1: Training curves on MuJoCo continuous control environments. The shaded region corresponds to one standard deviation. The results suggest that OPAC has a definitive edge over SAC and TD3.

7.1 Comparative Evaluation

The plots in Figure 1 show the total average return of evaluation rollouts 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. 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 have been run with a purely exploratory policy for the first 10,000 time steps. Policy evaluation is performed after every 5,000 time steps. Each evaluation step is performed over 20 episodes. The evaluation reports the mean of the cumulative reward of the 20 episodes without discount and any noise. In fact, all the plots in the paper and appendix adhere to these guidelines. Plots in Figure 1 indicate that OPAC yields higher average rewards over time than SAC and TD3. The maximum rewards obtained in each environment for OPAC, SAC, and TD3 is shown in Table 2.

7.2 Ablation Study

We perform ablation study on the number of critics first. In what follows, we try to address the following question: why use three critics and not more? Figure 2 helps us realise several important points. Firstly, OPAC doesn't outperform TD3 and SAC simply because it has more parameters to train owing to the usage of three critics. Had it been so, then SAC-3, SAC-4, TD3-3, and

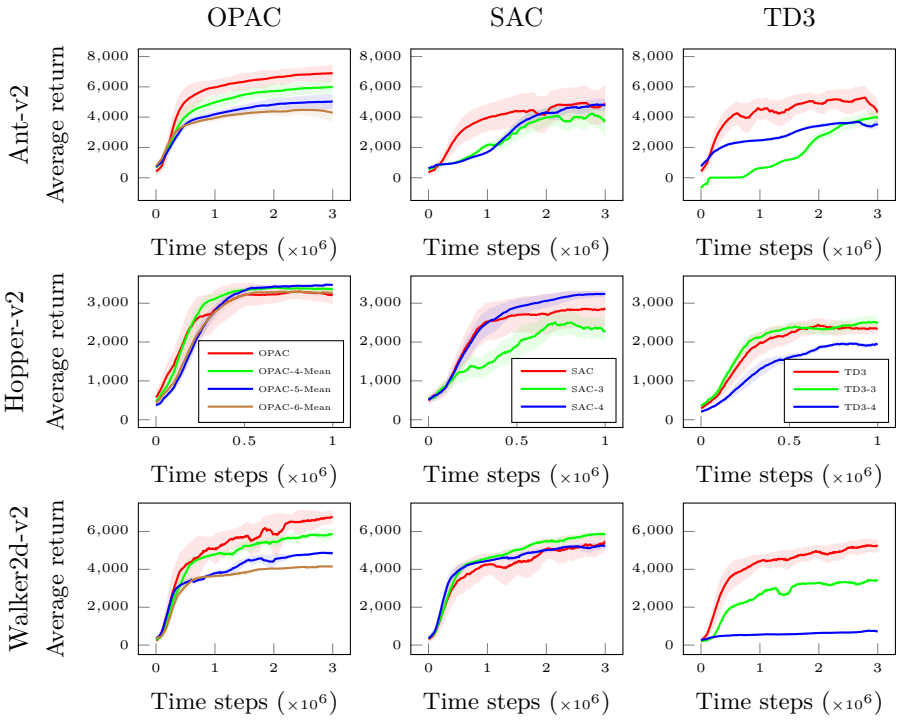


Fig. 2: Performance of different algorithms with more critics. The shaded region corresponds to one standard deviation. Results show that using three critics for OPAC consistently delivers a good performance.

TD3-4 would have performed consistently better than their vanilla versions. Second, having more than three critics in OPAC is not a good choice as we receive sub-optimal average rewards for OPAC-4-Mean, OPAC-5-Mean, and OPAC-6-Mean. Table 3 shows the maximum rewards obtained by the different variants of OPAC, SAC, and TD3 in the corresponding environments of Figure 2.

Note that the minimisation 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 [12]. Let $\text{mst}(x_1, \ldots, x_n)$ and $\text{mlt}(x_1, \ldots, x_n)$ denote the mean of the *smallest* two Q-values and *largest* two Q-values respectively out of n Q-values. Experimental evaluations suggest both $\text{mst}(x_1, x_2, x_3)$ and $\text{mst}(x_1, \ldots, x_4)$ are lower bounded by $\mu - \sigma$ (as shown in Figure 3.(a)), where μ and σ denote the mean and the standard deviation of the numbers respectively. For $n > 4$, the function is not bounded below anymore and the probability that $\text{mst}(x_1, \ldots, x_n)$ exceeds $\mu - \sigma$ increases drastically. This probability roughly gets halved as n is incremented by one. In fact, for $n = 6$ it's lower bounded only by probability $\sim$0.2. We hypothesise

Table 3: The maximum mean return (averaged over 5 trials) of the different variants of TD3, SAC, and OPAC. The ± corresponds to one standard deviation. Maximum reward in an environment has been boldfaced.

Algorithm \ Environment	Ant-v2	Hopper-v2	Walker2d-v2
SAC	5580.93 ± 307.76	3354.10 ± 17.84	5733.03 ± 49.22
SAC-3	4751.11 ± 112.25	3317.38 ± 248.84	6035.48 ± 35.59
SAC-4	5091.44 ± 54.87	3316.90 ± 11.93	5423.74 ± 23.80
TD3	5817.13 ± 138.28	2858.70 ± 18.65	5647.76 ± 52.52
TD3-3	4309.67 ± 76.56	2908.60 ± 14.84	3585.04 ± 11.02
TD3-4	3894.86 ± 33.49	2091.47 ± 3.52	1103.03 ± 341.61
OPAC	**7097.45 ± 90.80**	**3610.51 ± 26.80**	**7008.71 ± 50.41**
OPAC-4-Mean	6166.51 ± 58.75	3466.41 ± 5.08	6081.18 ± 22.64
OPAC-4-Med	**7136.89 ± 386.10**	3420.31 ± 153.74	6188.68 ± 76.20
OPAC-5-Mean	5241.13 ± 47.31	**3525.72 ± 4.28**	4984.11 ± 14.74
OPAC-6-Mean	4713.99 ± 38.40	3369.56 ± 5.55	4273.17 ± 20.48

that the sub-optimality of Clipped Q-learning for more than three critics is probably due to this unbounded nature of the function.

To comprehend the above claim, we can consider it in terms of outliers. For more than four critics, since the estimates are not bounded, the candidate Q-values might be way off (specifically at the beginning when the Q-functions are fairly inaccurate). In other words, some of the Q-values might simply be an outlier and $\text{mst}(x_1, x_2, x_3)$ gives us a somewhat conservative (arguably pessimistic) estimate of the shared Q-value (Figure 3.(b)). Contrast it to $\text{mlt}(x_1, x_2, x_3)$, as depicted in the figure, which is susceptible to over-estimate the Q-values.

Roughly stated, as the number of critics increase, the function $\text{mst}(\cdot)$ starts acting more and more like $\min(\cdot)$. This also supports our previous observation of the lack of bound (note that $\min(x_1, \ldots, x_n)$ is not bounded below by $\mu - \sigma$ for $n > 2$). As described in [12], using the minimum of the estimators as an unbiased estimate of the maximum expected action value is simply too pessimistic and might cause pessimistic under-exploration. Due to this phenomenon, for relatively large n we fail to counter the effect of pessimism.

Let $\text{med}(x_1, \ldots, x_n)$ denote the median of n Q-values. Plots in Figure 4 suggest that $\text{mst}(x_1, x_2, x_3)$ is a better aggregation strategy than $\text{med}(x_1, x_2, x_3)$ and $\text{med}(x_1, \ldots, x_4)$ as it yields higher average rewards over time. Further results involving the comparison of $\text{mst}(x_1, x_2, x_3)$ with $\text{mst}(x_1, \ldots, x_4)$, $\text{mst}(x_1, \ldots, x_5)$, and $\text{mst}(x_1, \ldots, x_6)$ can be found in Figure 5 from Appendix A.1. These results suggest that taking the mean of the smallest two Q-values out of three Q-values works well for OPAC and is capable of delivering good performance consistently.

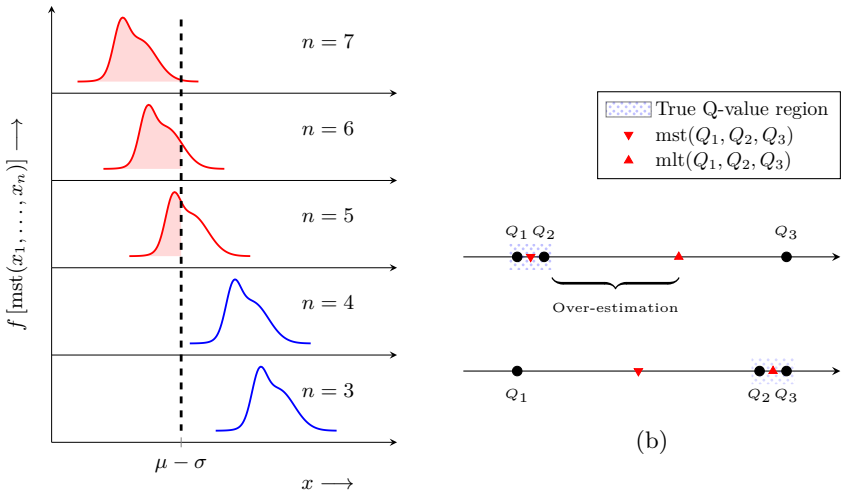
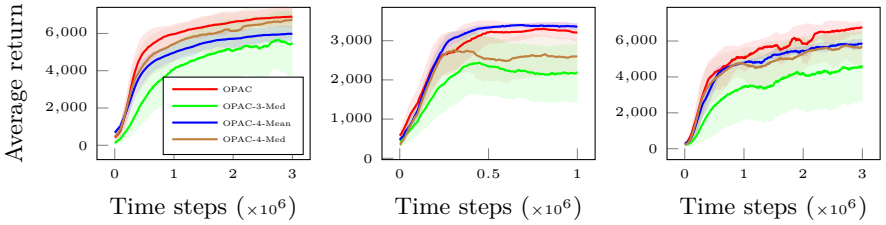


Fig. 3: (a) The distribution of $\text{mst}(x_1, \ldots, x_n)$ for randomly sampled numbers from an interval $[c_1, c_2]$. The red filled region, $f\,[\text{mst}(x_1, \ldots, x_n) < \mu - \sigma]$, indicates the probability that $\text{mst}(x_1, \ldots, x_n)$ violates the lower bound $\mu - \sigma$. Only $\text{mst}(x_1, x_2, x_3)$ and $\text{mst}(x_1, \ldots, x_4)$ are lower bounded by $\mu - \sigma$. (b) A comparison between $\text{mst}(x_1, x_2, x_3)$ and $\text{mlt}(x_1, x_2, x_3)$ in the presence of outliers. $\text{mlt}(x_1, x_2, x_3)$ tends to overestimate whereas $\text{mst}(x_1, x_2, x_3)$ doesn't.



(a) Ant-v2 (b) Hopper-v2 (c) Walker2d-v2

Fig. 4: OPAC consistently performs better than the other versions shown here.

8 Conclusions

This paper presents Opportunistic Actor-Critic (OPAC), an ensemble model-free deep RL algorithm that combines the core features of TD3 and SAC. In addition to this, OPAC uses three critics and considers the mean of the smallest two Q-values for fine tuning the value updates. We presented detailed empirical results which show that OPAC consistently yields higher rewards compared to SAC and TD3 in all the environments tested. OPAC achieves this feat without the need of environment-specific hyperparameter tuning. We

developed the theory of Clipped Triple Q-learning and also established its proof of convergence. Our experiments indicate that OPAC is robust and sample-efficient for easy environments like InvertedPendulum-v2 as well as challenging ones like Humanoid-v2.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. CoRR **abs/1312.5602** (2013) https://arxiv.org/abs/1312.5602

[2] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–503 (2016)

[3] Alonso, E., Peter, M., Goumard, D., Romoff, J.: Deep reinforcement learning for navigation in aaa video games. (2021). https://doi.org/10.24963/ijcai.2021/294

[4] Gu, S., Holly, E., Lillicrap, T., Levine, S.: 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). https://doi.org/10.1109/ICRA.2017.7989385

[5] Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., Levine, S.: Composable deep reinforcement learning for robotic manipulation. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6244–6251 (2018). https://doi.org/10.1109/ICRA.2018.8460756

[6] Chen, J., Yuan, B., Tomizuka, M.: Model-free deep reinforcement learning for urban autonomous driving. (2019). https://doi.org/10.1109/ITSC.2019.8917306

[7] Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. (2000)

[8] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Bengio, Y., LeCun, Y. (eds.) ICLR (2016). http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15

[9] Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1587–1596 (2018). https://

proceedings.mlr.press/v80/fujimoto18a.html

[10] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1861–1870 (2018). https://proceedings.mlr.press/v80/haarnoja18b.html

[11] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft actor-critic algorithms and applications. CoRR **abs/1812.05905** (2018) https://arxiv.org/abs/1812.05905

[12] Ciosek, K., Vuong, Q., Loftin, R., Hofmann, K.: Better exploration with optimistic actor critic. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (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). https://proceedings.neurips.cc/paper/2019/hash/a34bacf839b923770b2c360eefa26748-Abstract.html

[13] Wu, D., Dong, X., Shen, J., Hoi, S.C.H.: Reducing estimation bias via triplet-average deep deterministic policy gradient. IEEE Transactions on Neural Networks and Learning Systems **31**(11), 4933–4945 (2020). https://doi.org/10.1109/TNNLS.2019.2959129

[14] Ziebart, B.D.: Modeling purposeful adaptive behavior with the principle of maximum causal entropy. PhD thesis, USA (2010)

[15] Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Fox, D., Gomes, C.P. (eds.) Proceedings of the 23rd National Conference on Artificial Intelligence. AAAI'08, vol. 3, pp. 1433–1438 (2008)

[16] Todorov, E.: General duality between optimal control and estimation. In: 2008 47th IEEE Conference on Decision and Control, pp. 4286–4292 (2008)

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

[18] Rawlik, K., Toussaint, M., Vijayakumar, S.: 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)

[19] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

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

[21] Singh, S., Jaakkola, T., Littman, M.L., Szepesvári, C.: Convergence results for single-step on-policy reinforcement-learning algorithms. Mach. Learn. **38**(3), 287–308 (2000). https://doi.org/10.1023/A:1007678930559

[22] Bertsekas, D.P.: Dynamic Programming and Optimal Control, 2nd edn. Athena Scientific, Belmont, MA, USA (2000)

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

[24] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016)

A Appendix

A.1 Additional results

Figure 5 shows a performance comparison between the variants of OPAC which differ only in the number of critics. This is to demonstrate that OPAC is superior to OPAC-4-Mean, OPAC-5-Mean, and OPAC-6-Mean. To put it another way, having more than three critics has no bearing on OPAC's performance. Having more than three critics not only complicates the algorithm's architecture, but it also affects the quality of the average rewards accumulated. The explanation for this is discussed in greater depth in Section 7.2 of the paper. The plots in Figure 6 contain the training curves of OPAC-2, TD3, and SAC. It can be seen from the plots that OPAC-2 outperforms TD3 and SAC. We prefer OPAC to OPAC-2 because the former returns higher average rewards over time.

Let OPAC-3 and OPAC-4 denote the variant of OPAC with three and four critics respectively and let their aggregation strategy be taking the minimum

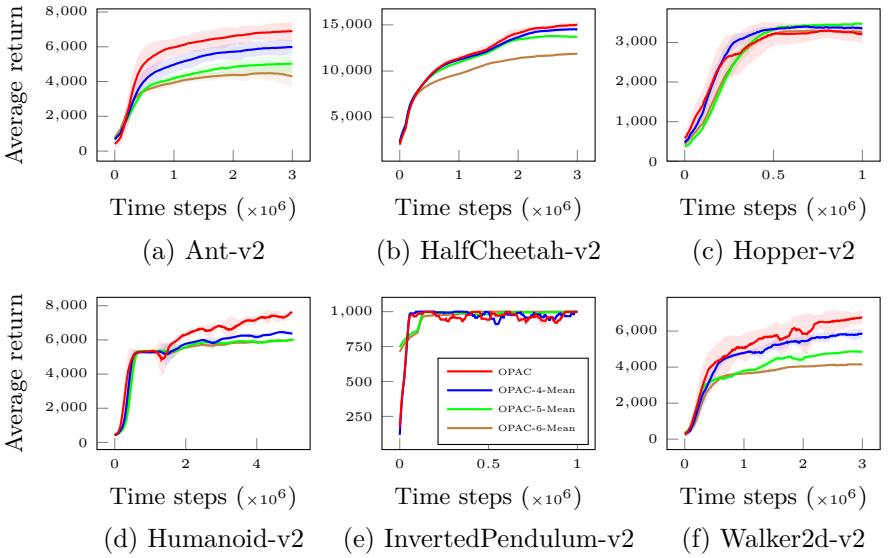


Fig. 5: Performance comparsion of OPAC varying only in the number of critics.

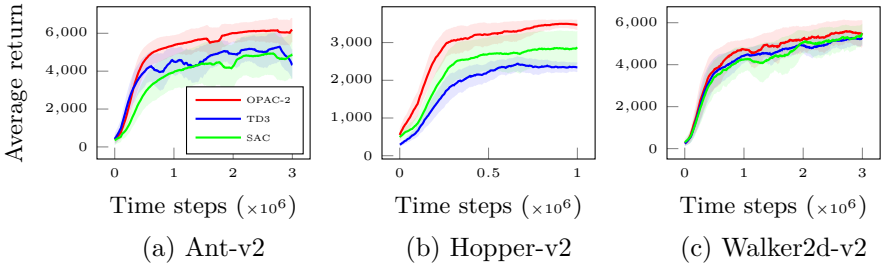


Fig. 6: Performance comparison of two-critic variants of OPAC, TD3, and SAC.

of all the critics. Comparing the performances of OPAC, with OPAC-2, OPAC-3, and OPAC-4 seemed redundant to us because the function $\text{mst}(x_1, \ldots, x_n)$ starts behaving like $\min(\cdot)$ with the increase in n.

A.2 List of hyperparameters

Table 4 lists the common OPAC, SAC, and TD3 parameters used in the comparative evaluation.

A.3 System Specifications

Table 5 lists the necessary specifications of the GPUs used to execute the codes.

Table 4: Hyperparameter choices for OPAC, TD3, and SAC.

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

Table 5: GPU specifications.

Attribute	**Value**
GPU Name	GeForce RTX 2080 Ti Rev. A
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