Addressing Function Approximation Error in Actor-Critic Methods

Scott Fujimoto [1] **Herke van Hoof** [2] **David Meger** [1]

Abstract

In value-based reinforcement learning methods such as deep Q-learning, function approximation errors are known to lead to overestimated value estimates and suboptimal policies. We show that this problem persists in an actor-critic setting and propose novel mechanisms to minimize its effects on both the actor and the critic. Our algorithm builds on Double Q-learning, by taking the minimum value between a pair of critics to limit overestimation. We draw the connection between target networks and overestimation bias, and suggest delaying policy updates to reduce per-update error and further improve performance. We evaluate our method on the suite of OpenAI gym tasks, outperforming the state of the art in every environment tested.

1. Introduction

In reinforcement learning problems with discrete action spaces, the issue of value overestimation as a result of function approximation errors is well-studied. However, similar issues with actor-critic methods in continuous control domains have been largely left untouched. In this paper, we show overestimation bias and the accumulation of error in temporal difference methods are present in an actor-critic setting. Our proposed method addresses these issues, and greatly outperforms the current state of the art.

Overestimation bias is a property of Q-learning in which the maximization of a noisy value estimate induces a consistent overestimation (Thrun & Schwartz, 1993). In a function approximation setting, this noise is unavoidable given the imprecision of the estimator. This inaccuracy is further exaggerated by the nature of temporal difference learning (Sutton, 1988), in which an estimate of the value function is updated using the estimate of a subsequent state. This means using an imprecise estimate within each update will lead to an accumulation of error. Due to overestimation bias, this accumulated error can cause arbitrarily bad states to be estimated as high value, resulting in suboptimal policy updates and divergent behavior.

This paper begins by establishing this overestimation property is also present for deterministic policy gradients (Silver et al., 2014), in the continuous control setting. Furthermore, we find the ubiquitous solution in the discrete action setting, Double DQN (Van Hasselt et al., 2016), to be ineffective in an actor-critic setting. During training, Double DQN estimates the value of the current policy with a separate target value function, allowing actions to be evaluated without maximization bias. Unfortunately, due to the slow-changing policy in an actor-critic setting, the current and target value estimates remain too similar to avoid maximization bias. This can be dealt with by adapting an older variant, Double Q-learning (Van Hasselt, 2010), to an actor-critic format by using a pair of independently trained critics. While this allows for a less biased value estimation, even an unbiased estimate with high variance can still lead to future overestimations in local regions of state space, which in turn can negatively affect the global policy. To address this concern, we propose a clipped Double Q-learning variant which leverages the notion that a value estimate suffering from overestimation bias can be used as an approximate upper-bound to the true value estimate. This favors underestimations, which do not tend to be propagated during learning, as actions with low value estimates are avoided by the policy.

Given the connection of noise to overestimation bias, this paper contains a number of components that address variance reduction. First, we show that target networks, a common approach in deep Q-learning methods, are critical for variance reduction by reducing the accumulation of errors. Second, to address the coupling of value and policy, we propose delaying policy updates until the value estimate has converged. Finally, we introduce a novel regularization strategy, where a SARSA-style update bootstraps similar action estimates to further reduce variance.

Our modifications are applied to the state of the art actor-critic method for continuous control, Deep Deterministic Policy Gradient algorithm (DDPG) (Lillicrap et al., 2015), to form the Twin Delayed Deep Deterministic policy gradient

[1]McGill University, Montreal, Canada [2]University of Amsterdam, Amsterdam, Netherlands. Correspondence to: Scott Fujimoto <scott.fujimoto@mail.mcgill.ca>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

algorithm (TD3), an actor-critic algorithm which considers the interplay between function approximation error in both policy and value updates. We evaluate our algorithm on seven continuous control domains from OpenAI gym (Brockman et al., 2016), where we outperform the state of the art by a wide margin.

Given the recent concerns in reproducibility (Henderson et al., 2017), we run our experiments across a large number of seeds with fair evaluation metrics, perform ablation studies across each contribution, and open source both our code and learning curves (`https://github.com/sfujim/TD3`).

2. Related Work

Function approximation error and its effect on bias and variance in reinforcement learning algorithms have been studied in prior works (Pendrith et al., 1997; Mannor et al., 2007). Our work focuses on two outcomes that occur as the result of estimation error, namely overestimation bias and a high variance build-up.

Several approaches exist to reduce the effects of overestimation bias due to function approximation and policy optimization in Q-learning. Double Q-learning uses two independent estimators to make unbiased value estimates (Van Hasselt, 2010; Van Hasselt et al., 2016). Other approaches have focused directly on reducing the variance (Anschel et al., 2017), minimizing over-fitting to early high variance estimates (Fox et al., 2016), or through corrective terms (Lee et al., 2013). Further, the variance of the value estimate has been considered directly for risk-aversion (Mannor & Tsitsiklis, 2011) and exploration (O'Donoghue et al., 2017), but without connection to overestimation bias.

The concern of variance due to the accumulation of error in temporal difference learning has been largely dealt with by either minimizing the size of errors at each time step or mixing off-policy and Monte-Carlo returns. Our work shows the importance of a standard technique, target networks, for the reduction of per-update error, and develops a regularization technique for the variance reduction by averaging over value estimates. Concurrently, Nachum et al. (2018) showed smoothed value functions could be used to train stochastic policies with reduced variance and improved performance. Methods with multi-step returns offer a trade-off between accumulated estimation bias and variance induced by the policy and the environment. These methods have been shown to be an effective approach, through importance sampling (Precup et al., 2001; Munos et al., 2016), distributed methods (Mnih et al., 2016; Espeholt et al., 2018), and approximate bounds (He et al., 2016). However, rather than provide a direct solution to the accumulation of error, these methods circumvent the problem by considering a longer horizon. Another approach is a reduction in the discount factor (Petrik & Scherrer, 2009), reducing the contribution of each error.

Our method builds on the Deterministic Policy Gradient algorithm (DPG) (Silver et al., 2014), an actor-critic method which uses a learned value estimate to train a deterministic policy. An extension of DPG to deep reinforcement learning, DDPG (Lillicrap et al., 2015), has shown to produce state of the art results with an efficient number of iterations. Orthogonal to our approach, recent improvements to DDPG include distributed methods (Popov et al., 2017), along with multi-step returns and prioritized experience replay (Schaul et al., 2016; Horgan et al., 2018), and distributional methods (Bellemare et al., 2017; Barth-Maron et al., 2018).

3. Background

Reinforcement learning considers the paradigm of an agent interacting with its environment with the aim of learning reward-maximizing behavior. At each discrete time step t, with a given state $s \in \mathcal{S}$, the agent selects actions $a \in \mathcal{A}$ with respect to its policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, receiving a reward r and the new state of the environment s'. The return is defined as the discounted sum of rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, where γ is a discount factor determining the priority of short-term rewards.

In reinforcement learning, the objective is to find the optimal policy π_ϕ, with parameters ϕ, which maximizes the expected return $J(\phi) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_0]$. For continuous control, parametrized policies π_ϕ can be updated by taking the gradient of the expected return $\nabla_\phi J(\phi)$. In actor-critic methods, the policy, known as the actor, can be updated through the deterministic policy gradient algorithm (Silver et al., 2014):

$$\nabla_\phi J(\phi) = \mathbb{E}_{s \sim p_\pi}\left[\nabla_a Q^\pi(s,a)|_{a=\pi(s)} \nabla_\phi \pi_\phi(s)\right]. \quad (1)$$

$Q^\pi(s,a) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_t | s, a]$, the expected return when performing action a in state s and following π after, is known as the critic or the value function.

In Q-learning, the value function can be learned using temporal difference learning (Sutton, 1988; Watkins, 1989), an update rule based on the Bellman equation (Bellman, 1957). The Bellman equation is a fundamental relationship between the value of a state-action pair (s,a) and the value of the subsequent state-action pair (s',a'):

$$Q^\pi(s,a) = r + \gamma \mathbb{E}_{s',a'}\left[Q^\pi(s',a')\right], \quad a' \sim \pi(s'). \quad (2)$$

For a large state space, the value can be estimated with a differentiable function approximator $Q_\theta(s,a)$, with parameters θ. In deep Q-learning (Mnih et al., 2015), the network is updated by using temporal difference learning with a secondary frozen target network $Q_{\theta'}(s,a)$ to maintain a fixed

objective y over multiple updates:

$$y = r + \gamma Q_{\theta'}(s', a'), \quad a' \sim \pi_{\phi'}(s'), \tag{3}$$

where the actions are selected from a target actor network $\pi_{\phi'}$. The weights of a target network are either updated periodically to exactly match the weights of the current network, or by some proportion τ at each time step $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$. This update can be applied in an off-policy fashion, sampling random mini-batches of transitions from an experience replay buffer (Lin, 1992).

4. Overestimation Bias

In Q-learning with discrete actions, the value estimate is updated with a greedy target $y = r + \gamma \max_{a'} Q(s', a')$, however, if the target is susceptible to error ϵ, then the maximum over the value along with its error will generally be greater than the true maximum, $\mathbb{E}_\epsilon[\max_{a'}(Q(s', a') + \epsilon)] \geq \max_{a'} Q(s', a')$ (Thrun & Schwartz, 1993). As a result, even initially zero-mean error can cause value updates to result in a consistent overestimation bias, which is then propagated through the Bellman equation. This is problematic as errors induced by function approximation are unavoidable.

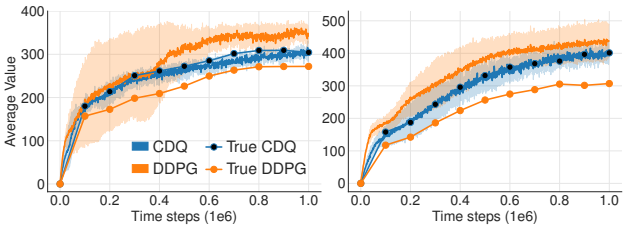
While in the discrete action setting overestimation bias is an obvious artifact from the analytical maximization, the presence and effects of overestimation bias is less clear in an actor-critic setting where the policy is updated via gradient descent. We begin by proving that the value estimate in deterministic policy gradients will be an overestimation under some basic assumptions in Section 4.1 and then propose a clipped variant of Double Q-learning in an actor-critic setting to reduce overestimation bias in Section 4.2.

4.1. Overestimation Bias in Actor-Critic

In actor-critic methods the policy is updated with respect to the value estimates of an approximate critic. In this section we assume the policy is updated using the deterministic policy gradient, and show that the update induces overestimation in the value estimate. Given current policy parameters ϕ, let ϕ_{approx} define the parameters from the actor update induced by the maximization of the approximate critic $Q_\theta(s, a)$ and ϕ_{true} the parameters from the hypothetical actor update with respect to the true underlying value function $Q^\pi(s, a)$ (which is not known during learning):

$$\phi_{\text{approx}} = \phi + \frac{\alpha}{Z_1}\mathbb{E}_{s\sim p_\pi}\left[\nabla_\phi \pi_\phi(s) \nabla_a Q_\theta(s, a)|_{a=\pi_\phi(s)}\right]$$
$$\phi_{\text{true}} = \phi + \frac{\alpha}{Z_2}\mathbb{E}_{s\sim p_\pi}\left[\nabla_\phi \pi_\phi(s) \nabla_a Q^\pi(s, a)|_{a=\pi_\phi(s)}\right], \tag{4}$$

where we assume Z_1 and Z_2 are chosen to normalize the gradient, i.e., such that $Z^{-1}||\mathbb{E}[\cdot]|| = 1$. Without normalized gradients, overestimation bias is still guaranteed to occur with slightly stricter conditions. We examine this case further in the supplementary material. We denote π_{approx} and π_{true} as the policy with parameters ϕ_{approx} and ϕ_{true} respectively.



(a) Hopper-v1 (b) Walker2d-v1

Figure 1. Measuring overestimation bias in the value estimates of DDPG and our proposed method, Clipped Double Q-learning (CDQ), on MuJoCo environments over 1 million time steps.

As the gradient direction is a local maximizer, there exists ϵ_1 sufficiently small such that if $\alpha \leq \epsilon_1$ then the *approximate* value of π_{approx} will be bounded below by the *approximate* value of π_{true}:

$$\mathbb{E}\left[Q_\theta(s, \pi_{\text{approx}}(s))\right] \geq \mathbb{E}\left[Q_\theta(s, \pi_{\text{true}}(s))\right]. \tag{5}$$

Conversely, there exists ϵ_2 sufficiently small such that if $\alpha \leq \epsilon_2$ then the *true* value of π_{approx} will be bounded above by the *true* value of π_{true}:

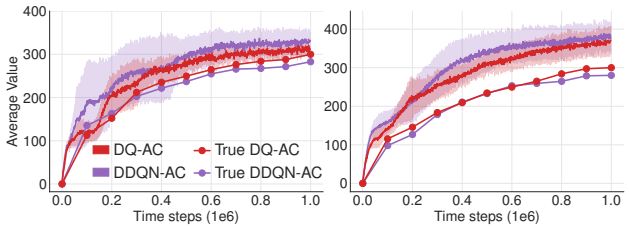
$$\mathbb{E}\left[Q^\pi(s, \pi_{\text{true}}(s))\right] \geq \mathbb{E}\left[Q^\pi(s, \pi_{\text{approx}}(s))\right]. \tag{6}$$

If in expectation the value estimate is at least as large as the *true* value with respect to ϕ_{true}, $\mathbb{E}\left[Q_\theta\left(s, \pi_{\text{true}}(s)\right)\right] \geq \mathbb{E}\left[Q^\pi\left(s, \pi_{\text{true}}(s)\right)\right]$, then Equations (5) and (6) imply that if $\alpha < \min(\epsilon_1, \epsilon_2)$, then the value estimate will be overestimated:

$$\mathbb{E}\left[Q_\theta(s, \pi_{\text{approx}}(s))\right] \geq \mathbb{E}\left[Q^\pi(s, \pi_{\text{approx}}(s))\right]. \tag{7}$$

Although this overestimation may be minimal with each update, the presence of error raises two concerns. Firstly, the overestimation may develop into a more significant bias over many updates if left unchecked. Secondly, an inaccurate value estimate may lead to poor policy updates. This is particularly problematic because a feedback loop is created, in which suboptimal actions might be highly rated by the suboptimal critic, reinforcing the suboptimal action in the next policy update.

Does this theoretical overestimation occur in practice for state-of-the-art methods? We answer this question by plotting the value estimate of DDPG (Lillicrap et al., 2015) over time while it learns on the OpenAI gym environments Hopper-v1 and Walker2d-v1 (Brockman et al., 2016). In Figure 1, we graph the average value estimate over 10000 states and compare it to an estimate of the true value. The



(a) Hopper-v1 (b) Walker2d-v1

Figure 2. Measuring overestimation bias in the value estimates of actor critic variants of Double DQN (DDQN-AC) and Double Q-learning (DQ-AC) on MuJoCo environments over 1 million time steps.

true value is estimated using the average discounted return over 1000 episodes following the current policy, starting from states sampled from the replay buffer. A very clear overestimation bias occurs from the learning procedure, which contrasts with the novel method that we describe in the following section, Clipped Double Q-learning, which greatly reduces overestimation by the critic.

4.2. Clipped Double Q-Learning for Actor-Critic

While several approaches to reducing overestimation bias have been proposed, we find them ineffective in an actor-critic setting. This section introduces a novel clipped variant of Double Q-learning (Van Hasselt, 2010), which can replace the critic in any actor-critic method.

In Double Q-learning, the greedy update is disentangled from the value function by maintaining two separate value estimates, each of which is used to update the other. If the value estimates are independent, they can be used to make unbiased estimates of the actions selected using the opposite value estimate. In Double DQN (Van Hasselt et al., 2016), the authors propose using the target network as one of the value estimates, and obtain a policy by greedy maximization of the current value network rather than the target network. In an actor-critic setting, an analogous update uses the current policy rather than the target policy in the learning target:

$$y = r + \gamma Q_{\theta'}(s', \pi_\phi(s')). \tag{8}$$

In practice however, we found that with the slow-changing policy in actor-critic, the current and target networks were too similar to make an independent estimation, and offered little improvement. Instead, the original Double Q-learning formulation can be used, with a pair of actors $(\pi_{\phi_1}, \pi_{\phi_2})$ and critics $(Q_{\theta_1}, Q_{\theta_2})$, where π_{ϕ_1} is optimized with respect to Q_{θ_1} and π_{ϕ_2} with respect to Q_{θ_2}:

$$\begin{aligned} y_1 &= r + \gamma Q_{\theta'_2}(s', \pi_{\phi_1}(s')) \\ y_2 &= r + \gamma Q_{\theta'_1}(s', \pi_{\phi_2}(s')). \end{aligned} \tag{9}$$

We measure the overestimation bias in Figure 2, which demonstrates that the actor-critic Double DQN suffers from a similar overestimation as DDPG (as shown in Figure 1). While Double Q-learning is more effective, it does not entirely eliminate the overestimation. We further show this reduction is not sufficient experimentally in Section 6.1.

As π_{ϕ_1} optimizes with respect to Q_{θ_1}, using an independent estimate in the target update of Q_{θ_1} would avoid the bias introduced by the policy update. However the critics are not entirely independent, due to the use of the opposite critic in the learning targets, as well as the same replay buffer. As a result, for some states s we will have $Q_{\theta_2}(s, \pi_{\phi_1}(s)) > Q_{\theta_1}(s, \pi_{\phi_1}(s))$. This is problematic because $Q_{\theta_1}(s, \pi_{\phi_1}(s))$ will generally overestimate the true value, and in certain areas of the state space the overestimation will be further exaggerated. To address this problem, we propose to simply upper-bound the less biased value estimate Q_{θ_2} by the biased estimate Q_{θ_1}. This results in taking the minimum between the two estimates, to give the target update of our Clipped Double Q-learning algorithm:

$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi_1}(s')). \tag{10}$$

With Clipped Double Q-learning, the value target cannot introduce any additional overestimation over using the standard Q-learning target. While this update rule may induce an underestimation bias, this is far preferable to overestimation bias, as unlike overestimated actions, the value of underestimated actions will not be explicitly propagated through the policy update.

In implementation, computational costs can be reduced by using a single actor optimized with respect to Q_{θ_1}. We then use the same target $y_2 = y_1$ for Q_{θ_2}. If $Q_{\theta_2} > Q_{\theta_1}$ then the update is identical to the standard update and induces no additional bias. If $Q_{\theta_2} < Q_{\theta_1}$, this suggests overestimation has occurred and the value is reduced similar to Double Q-learning. A proof of convergence in the finite MDP setting follows from this intuition. We provide formal details and justification in the supplementary material.

A secondary benefit is that by treating the function approximation error as a random variable we can see that the minimum operator should provide higher value to states with lower variance estimation error, as the expected minimum of a set of random variables decreases as the variance of the random variables increases. This effect means that the minimization in Equation (10) will lead to a preference for states with low-variance value estimates, leading to safer policy updates with stable learning targets.

5. Addressing Variance

While Section 4 deals with the contribution of variance to overestimation bias, we also argue that variance itself should be directly addressed. Besides the impact on overestimation

bias, high variance estimates provide a noisy gradient for the policy update. This is known to reduce learning speed (Sutton & Barto, 1998) as well as hurt performance in practice. In this section we emphasize the importance of minimizing error at each update, build the connection between target networks and estimation error and propose modifications to the learning procedure of actor-critic for variance reduction.

5.1. Accumulating Error

Due to the temporal difference update, where an estimate of the value function is built from an estimate of a subsequent state, there is a build up of error. While it is reasonable to expect small error for an individual update, these estimation errors can accumulate, resulting in the potential for large overestimation bias and suboptimal policy updates. This is exacerbated in a function approximation setting where the Bellman equation is never exactly satisfied, and each update leaves some amount of residual TD-error $\delta(s, a)$:

$$Q_\theta(s, a) = r + \gamma \mathbb{E}[Q_\theta(s', a')] - \delta(s, a). \tag{11}$$

It can then be shown that rather than learning an estimate of the expected return, the value estimate approximates the expected return minus the expected discounted sum of future TD-errors:

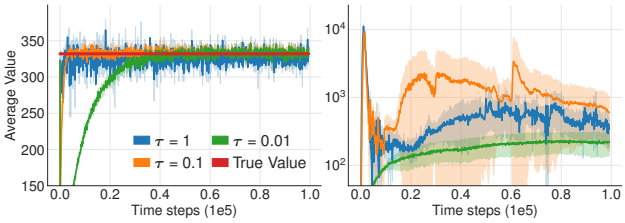
$$\begin{aligned} Q_\theta(s_t, a_t) &= r_t + \gamma \mathbb{E}[Q_\theta(s_{t+1}, a_{t+1})] - \delta_t \\ &= r_t + \gamma \mathbb{E}\left[r_{t+1} + \gamma \mathbb{E}\left[Q_\theta(s_{t+2}, a_{t+2}) - \delta_{t+1}\right]\right] - \delta_t \\ &= \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}\left[\sum_{i=t}^{T} \gamma^{i-t}(r_i - \delta_i)\right]. \end{aligned} \tag{12}$$

If the value estimate is a function of future reward and estimation error, it follows that the variance of the estimate will be proportional to the variance of future reward and estimation error. Given a large discount factor γ, the variance can grow rapidly with each update if the error from each update is not tamed. Furthermore each gradient update only reduces error with respect to a small mini-batch which gives no guarantees about the size of errors in value estimates outside the mini-batch.

5.2. Target Networks and Delayed Policy Updates

In this section we examine the relationship between target networks and function approximation error, and show the use of a stable target reduces the growth of error. This insight allows us to consider the interplay between high variance estimates and policy performance, when designing reinforcement learning algorithms.

Target networks are a well-known tool to achieve stability in deep reinforcement learning. As deep function approximators require multiple gradient updates to converge, target networks provide a stable objective in the learning procedure, and allow a greater coverage of the training data. Without a fixed target, each update may leave residual error which will begin to accumulate. While the accumulation of error can be detrimental in itself, when paired with a policy maximizing over the value estimate, it can result in wildly divergent values.



(a) Fixed Policy (b) Learned Policy

Figure 3. Average estimated value of a randomly selected state on Hopper-v1 without target networks, ($\tau = 1$), and with slow-updating target networks, ($\tau = 0.1, 0.01$), with a fixed and a learned policy.

To provide some intuition, we examine the learning behavior with and without target networks on both the critic and actor in Figure 3, where we graph the value, in a similar manner to Figure 1, in the Hopper-v1 environment. In (a) we compare the behavior with a fixed policy and in (b) we examine the value estimates with a policy that continues to learn, trained with the current value estimate. The target networks use a slow-moving update rate, parametrized by τ.

While updating the value estimate without target networks ($\tau = 1$) increases the volatility, all update rates result in similar convergent behaviors when considering a fixed policy. However, when the policy is trained with the current value estimate, the use of fast-updating target networks results in highly divergent behavior.

When do actor-critic methods fail to learn? These results suggest that the divergence that occurs without target networks is the result of policy updates with a high variance value estimate. Figure 3, as well as Section 4, suggest failure can occur due to the interplay between the actor and critic updates. Value estimates diverge through overestimation when the policy is poor, and the policy will become poor if the value estimate itself is inaccurate.

If target networks can be used to reduce the error over multiple updates, and policy updates on high-error states cause divergent behavior, then the policy network should be updated at a lower frequency than the value network, to first minimize error before introducing a policy update. We propose delaying policy updates until the value error is as small as possible. The modification is to only update the policy and target networks after a fixed number of updates d to the critic. To ensure the TD-error remains small, we update the

target networks slowly $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$.

By sufficiently delaying the policy updates we limit the likelihood of repeating updates with respect to an unchanged critic. The less frequent policy updates that do occur will use a value estimate with lower variance, and in principle, should result in higher quality policy updates. This creates a two-timescale algorithm, as often required for convergence in the linear setting (Konda & Tsitsiklis, 2003). The effectiveness of this strategy is captured by our empirical results presented in Section 6.1, which show an improvement in performance while using fewer policy updates.

5.3. Target Policy Smoothing Regularization

A concern with deterministic policies is they can overfit to narrow peaks in the value estimate. When updating the critic, a learning target using a deterministic policy is highly susceptible to inaccuracies induced by function approximation error, increasing the variance of the target. This induced variance can be reduced through regularization. We introduce a regularization strategy for deep value learning, target policy smoothing, which mimics the learning update from SARSA (Sutton & Barto, 1998). Our approach enforces the notion that similar actions should have similar value. While the function approximation does this implicitly, the relationship between similar actions can be forced explicitly by modifying the training procedure. We propose that fitting the value of a small area around the target action

$$y = r + \mathbb{E}_\epsilon \left[Q_{\theta'}(s', \pi_{\phi'}(s') + \epsilon) \right], \tag{13}$$

would have the benefit of smoothing the value estimate by bootstrapping off of similar state-action value estimates. In practice, we can approximate this expectation over actions by adding a small amount of random noise to the target policy and averaging over mini-batches. This makes our modified target update:

$$\begin{aligned} y &= r + \gamma Q_{\theta'}(s', \pi_{\phi'}(s') + \epsilon), \\ \epsilon &\sim \text{clip}(\mathcal{N}(0, \sigma), -c, c), \end{aligned} \tag{14}$$

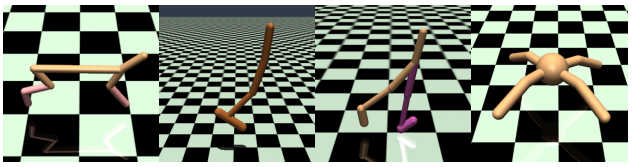
where the added noise is clipped to keep the target close to the original action. The outcome is an algorithm reminiscent of Expected SARSA (Van Seijen et al., 2009), where the value estimate is instead learned off-policy and the noise added to the target policy is chosen independently of the exploration policy. The value estimate learned is with respect to a noisy policy defined by the parameter σ.

Intuitively, it is known that policies derived from SARSA value estimates tend to be safer, as they provide higher value to actions resistant to perturbations. Thus, this style of update can additionally lead to improvement in stochastic domains with failure cases. A similar idea was introduced concurrently by Nachum et al. (2018), smoothing over Q_θ, rather than $Q_{\theta'}$.

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
    $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
  end if
end for



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

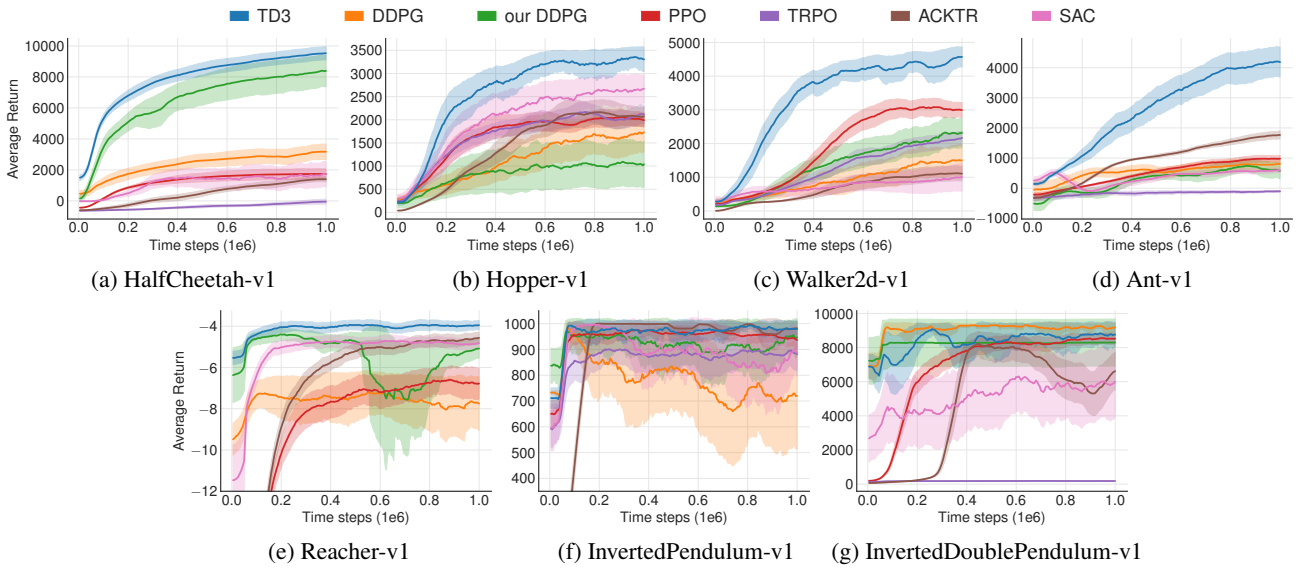
Figure 4. Example MuJoCo environments (a) HalfCheetah-v1, (b) Hopper-v1, (c) Walker2d-v1, (d) Ant-v1.

6. Experiments

We present the Twin Delayed Deep Deterministic policy gradient algorithm (TD3), which builds on the Deep Deterministic Policy Gradient algorithm (DDPG) (Lillicrap et al., 2015) by applying the modifications described in Sections 4.2, 5.2 and 5.3 to increase the stability and performance with consideration of function approximation error. TD3 maintains a pair of critics along with a single actor. For each time step, we update the pair of critics towards the minimum target value of actions selected by the target policy:

$$\begin{aligned} y &= r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi'}(s') + \epsilon), \\ \epsilon &\sim \text{clip}(\mathcal{N}(0, \sigma), -c, c). \end{aligned} \tag{15}$$

Every d iterations, the policy is updated with respect to Q_{θ_1} following the deterministic policy gradient algorithm (Silver et al., 2014). TD3 is summarized in Algorithm 1.



(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1

(e) Reacher-v1 (f) InvertedPendulum-v1 (g) InvertedDoublePendulum-v1

Figure 5. Learning curves for the OpenAI gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Curves are smoothed uniformly for visual clarity.

Table 1. Max Average Return over 10 trials of 1 million time steps. Maximum value for each task is bolded. $\pm$ corresponds to a single standard deviation over trials.

Environment	TD3	DDPG	Our DDPG	PPO	TRPO	ACKTR	SAC
HalfCheetah	**9636.95 $\pm$ 859.065**	3305.60	8577.29	1795.43	-15.57	1450.46	2347.19
Hopper	**3564.07 $\pm$ 114.74**	2020.46	1860.02	2164.70	2471.30	2428.39	2996.66
Walker2d	**4682.82 $\pm$ 539.64**	1843.85	3098.11	3317.69	2321.47	1216.70	1283.67
Ant	**4372.44 $\pm$ 1000.33**	1005.30	888.77	1083.20	-75.85	1821.94	655.35
Reacher	**-3.60 $\pm$ 0.56**	-6.51	**-4.01**	-6.18	-111.43	-4.26	-4.44
InvPendulum	**1000.00 $\pm$ 0.00**	**1000.00**	**1000.00**	**1000.00**	985.40	**1000.00**	**1000.00**
InvDoublePendulum	**9337.47 $\pm$ 14.96**	**9355.52**	8369.95	8977.94	205.85	9081.92	8487.15

6.1. Evaluation

To evaluate our algorithm, we measure its performance on the suite of MuJoCo continuous control tasks (Todorov et al., 2012), interfaced through OpenAI Gym (Brockman et al., 2016) (Figure 4). To allow for reproducible comparison, we use the original set of tasks from Brockman et al. (2016) with no modifications to the environment or reward.

For our implementation of DDPG (Lillicrap et al., 2015), we use a two layer feedforward neural network of 400 and 300 hidden nodes respectively, with rectified linear units (ReLU) between each layer for both the actor and critic, and a final tanh unit following the output of the actor. Unlike the original DDPG, the critic receives both the state and action as input to the first layer. Both network parameters are updated using Adam (Kingma & Ba, 2014) with a learning rate of 10^{-3}. After each time step, the networks are trained with a mini-batch of a 100 transitions, sampled uniformly from a replay buffer containing the entire history of the agent.

The target policy smoothing is implemented by adding $\epsilon \sim \mathcal{N}(0, 0.2)$ to the actions chosen by the target actor network, clipped to $(-0.5, 0.5)$, delayed policy updates consists of only updating the actor and target critic network every d iterations, with $d = 2$. While a larger d would result in a larger benefit with respect to accumulating errors, for fair comparison, the critics are only trained once per time step, and training the actor for too few iterations would cripple learning. Both target networks are updated with $\tau = 0.005$.

To remove the dependency on the initial parameters of the policy we use a purely exploratory policy for the first 10000 time steps of stable length environments (HalfCheetah-v1 and Ant-v1) and the first 1000 time steps for the remaining environments. Afterwards, we use an off-policy exploration strategy, adding Gaussian noise $\mathcal{N}(0, 0.1)$ to each action. Unlike the original implementation of DDPG, we used uncorrelated noise for exploration as we found noise drawn from the Ornstein-Uhlenbeck (Uhlenbeck & Ornstein, 1930) process offered no performance benefits.

Each task is run for 1 million time steps with evaluations every 5000 time steps, where each evaluation reports the

average reward over 10 episodes with no exploration noise. Our results are reported over 10 random seeds of the Gym simulator and the network initialization.

We compare our algorithm against DDPG (Lillicrap et al., 2015) as well as the state of art policy gradient algorithms: PPO (Schulman et al., 2017), ACKTR (Wu et al., 2017) and TRPO (Schulman et al., 2015), as implemented by OpenAI's baselines repository (Dhariwal et al., 2017), and SAC (Haarnoja et al., 2018), as implemented by the author's GitHub[1]. Additionally, we compare our method with our re-tuned version of DDPG, which includes all architecture and hyper-parameter modifications to DDPG without any of our proposed adjustments. A full comparison between our re-tuned version and the baselines DDPG is provided in the supplementary material.

Our results are presented in Table 1 and learning curves in Figure 5. TD3 matches or outperforms all other algorithms in both final performance and learning speed across all tasks.

6.2. Ablation Studies

We perform ablation studies to understand the contribution of each individual component: Clipped Double Q-learning (Section 4.2), delayed policy updates (Section 5.2) and target policy smoothing (Section 5.3). We present our results in Table 2 in which we compare the performance of removing each component from TD3 along with our modifications to the architecture and hyper-parameters. Additional learning curves can be found in the supplementary material.

The significance of each component varies task to task. While the addition of only a single component causes insignificant improvement in most cases, the addition of combinations performs at a much higher level. The full algorithm outperforms every other combination in most tasks. Although the actor is trained for only half the number of iterations, the inclusion of delayed policy update generally improves performance, while reducing training time.

We additionally compare the effectiveness of the actor-critic variants of Double Q-learning (Van Hasselt, 2010) and Double DQN (Van Hasselt et al., 2016), denoted DQ-AC and DDQN-AC respectively, in Table 2. For fairness in comparison, these methods also benefited from delayed policy updates, target policy smoothing and use our architecture and hyper-parameters. Both methods were shown to reduce overestimation bias less than Clipped Double Q-learning in Section 4. This is reflected empirically, as both methods result in insignificant improvements over TD3 - CDQ, with an exception in the Ant-v1 environment, which appears to benefit greatly from any overestimation reduction. As the inclusion of Clipped Double Q-learning into our full method outperforms both prior methods, this suggests that subduing the overestimations from the unbiased estimator is an effective measure to improve performance.

[1]See the supplementary material for hyper-parameters and a discussion on the discrepancy in the reported results of SAC.

Table 2. Average return over the last 10 evaluations over 10 trials of 1 million time steps, comparing ablation over delayed policy updates (DP), target policy smoothing (TPS), Clipped Double Q-learning (CDQ) and our architecture, hyper-parameters and exploration (AHE). Maximum value for each task is bolded.

Method	HCheetah	Hopper	Walker2d	Ant
TD3	9532.99	**3304.75**	**4565.24**	**4185.06**
DDPG	3162.50	1731.94	1520.90	816.35
AHE	8401.02	1061.77	2362.13	564.07
AHE + DP	7588.64	1465.11	2459.53	896.13
AHE + TPS	9023.40	907.56	2961.36	872.17
AHE + CDQ	6470.20	1134.14	3979.21	3818.71
TD3 - DP	9590.65	2407.42	**4695.50**	3754.26
TD3 - TPS	8987.69	2392.59	4033.67	**4155.24**
TD3 - CDQ	9792.80	1837.32	2579.39	849.75
DQ-AC	9433.87	1773.71	3100.45	2445.97
DDQN-AC	**10306.90**	2155.75	3116.81	1092.18

7. Conclusion

Overestimation has been identified as a key problem in value-based methods. In this paper, we establish overestimation bias is also problematic in actor-critic methods. We find the common solutions for reducing overestimation bias in deep Q-learning with discrete actions are ineffective in an actor-critic setting, and develop a novel variant of Double Q-learning which limits possible overestimation. Our results demonstrate that mitigating overestimation can greatly improve the performance of modern algorithms.

Due to the connection between noise and overestimation, we examine the accumulation of errors from temporal difference learning. Our work investigates the importance of a standard technique in deep reinforcement learning, target networks, and examines their role in limiting errors from imprecise function approximation and stochastic optimization. Finally, we introduce a SARSA-style regularization technique which modifies the temporal difference target to bootstrap off similar state-action pairs.

Taken together, these improvements define our proposed approach, the Twin Delayed Deep Deterministic policy gradient algorithm (TD3), which greatly improves both the learning speed and performance of DDPG in a number of challenging tasks in the continuous control setting. Our algorithm exceeds the performance of numerous state of the art algorithms. As our modifications are simple to implement, they can be easily added to any other actor-critic algorithm.

References

Anschel, O., Baram, N., and Shimkin, N. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 176–185, 2017.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. Distributional policy gradients. *International Conference on Learning Representations*, 2018.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458, 2017.

Bellman, R. *Dynamic Programming*. Princeton University Press, 1957.

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

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Dhariwal, P., Hesse, C., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Openai baselines. https://github.com/openai/baselines, 2017.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 202–211. AUAI Press, 2016.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

He, F. S., Liu, Y., Schwing, A. G., and Peng, J. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. *arXiv preprint arXiv:1611.01606*, 2016.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep Reinforcement Learning that Matters. *arXiv preprint arXiv:1709.06560*, 2017.

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

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konda, V. R. and Tsitsiklis, J. N. On actor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4):1143–1166, 2003.

Lee, D., Defourny, B., and Powell, W. B. Bias-corrected q-learning to control max-operator bias in q-learning. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*, pp. 93–99. IEEE, 2013.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Mannor, S. and Tsitsiklis, J. N. Mean-variance optimization in markov decision processes. In *International Conference on Machine Learning*, pp. 177–184, 2011.

Mannor, S., Simester, D., Sun, P., and Tsitsiklis, J. N. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Nachum, O., Norouzi, M., Tucker, G., and Schuurmans, D. Smoothed action value functions for learning gaussian policies. *arXiv preprint arXiv:1803.02348*, 2018.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty bellman equation and exploration. *arXiv preprint arXiv:1709.05380*, 2017.

Pendrith, M. D., Ryan, M. R., et al. *Estimator variance in reinforcement learning: Theoretical problems and practical solutions*. University of New South Wales, School of Computer Science and Engineering, 1997.

Petrik, M. and Scherrer, B. Biasing approximate dynamic programming with a lower discount factor. In *Advances in Neural Information Processing Systems*, pp. 1265–1272, 2009.

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

Precup, D., Sutton, R. S., and Dasgupta, S. Off-policy temporal-difference learning with function approximation. In *International Conference on Machine Learning*, pp. 417–424, 2001.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *International Conference on Learning Representations*, Puerto Rico, 2016.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395, 2014.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Thrun, S. and Schwartz, A. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

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

Uhlenbeck, G. E. and Ornstein, L. S. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Van Hasselt, H. Double q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, pp. 2094–2100, 2016.

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected sarsa. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pp. 177–184. IEEE, 2009.

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems*, pp. 5285–5294, 2017.

Supplementary Material

A. Proof of Convergence of Clipped Double Q-Learning

In a version of Clipped Double Q-learning for a finite MDP setting, we maintain two tabular value estimates Q^A, Q^B. At each time step we select actions $a^* = \text{argmax}_a Q^A(s, a)$ and then perform an update by setting target y:

$$\begin{aligned} a^* &= \operatorname*{argmax}_a Q^A(s', a) \\ y &= r + \gamma \min(Q^A(s', a^*), Q^B(s', a^*)), \end{aligned} \tag{16}$$

and update the value estimates with respect to the target and learning rate $\alpha_t(s, a)$:

$$\begin{aligned} Q^A(s, a) &= Q^A(s, a) + \alpha_t(s, a)(y - Q^A(s, a)) \\ Q^B(s, a) &= Q^B(s, a) + \alpha_t(s, a)(y - Q^B(s, a)). \end{aligned} \tag{17}$$

In a finite MDP setting, Double Q-learning is often used to deal with noise induced by random rewards or state transitions, and so either Q^A or Q^B is updated randomly. However, in a function approximation setting, the interest may be more towards the approximation error and thus we can update both Q^A and Q^B at each iteration. The proof extends naturally to updating either randomly.

The proof borrows heavily from the proof of convergence of SARSA (Singh et al., 2000) as well as Double Q-learning (Van Hasselt, 2010). The proof of lemma 1 can be found in Singh et al. (2000), building on a proposition from Bertsekas (1995).

Lemma 1. 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:

$$\Delta_{t+1}(x_t) = (1 - \zeta_t(x_t))\Delta_t(x_t) + \zeta_t(x_t)F_t(x_t), \tag{18}$$

where $x_t \in X$ and $t = 0, 1, 2, ...$ 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, $t = 1, 2, ...$ Assume that the following hold:

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

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

Theorem 1. Given the following conditions:

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

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

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

Proof of Theorem 1. We apply Lemma 1 with $P_t = \{Q_0^A, Q_0^B, s_0, a_0, \alpha_0, r_1, s_1, ..., s_t, a_t\}, X = S \times A, \Delta_t = Q_t^A - Q^*, \zeta_t = \alpha_t$.

First note that condition 1 and 4 of the lemma holds by the conditions 2 and 7 of the theorem respectively. Lemma condition 2 holds by the theorem condition 6 along with our selection of $\zeta_t = \alpha_t$.

Defining $a^* = \text{argmax}_a Q^A(s_{t+1}, a)$ we have

$$\begin{aligned}\Delta_{t+1}(s_t, a_t) &= (1 - \alpha_t(s_t, a_t))(Q_t^A(s_t, a_t) - Q^*(s_t, a_t)) \\ &\quad + \alpha_t(s_t, a_t)(r_t + \gamma \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) - Q^*(s_t, a_t)) \\ &= (1 - \alpha_t(s_t, a_t))\Delta_t(s_t, a_t) + \alpha_t(s_t, a_t)F_t(s_t, a_t)),\end{aligned} \tag{19}$$

where we have defined $F_t(s_t, a_t)$ as:

$$\begin{aligned}F_t(s_t, a_t) &= r_t + \gamma \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) - Q_t^*(s_t, a_t) \\ &= r_t + \gamma \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) - Q_t^*(s_t, a_t) + \gamma Q_t^A(s_{t+1}, a^*) - \gamma Q_t^A(s_{t+1}, a^*) \\ &= F_t^Q(s_t, a_t) + c_t,\end{aligned} \tag{20}$$

where $F_t^Q = r_t + \gamma Q_t^A(s_{t+1}, a^*) - Q_t^*(s_t, a_t)$ denotes the value of F_t under standard Q-learning and $c_t = \gamma \min(Q_t^A(s_{t+1}, a^*), Q_t^B(s_{t+1}, a^*)) - \gamma Q_t^A(s_{t+1}, a^*)$. As $\mathbb{E}\left[F_t^Q | P_t\right] \leq \gamma ||\Delta_t||$ is a well-known result, then condition 3 of lemma 1 holds if it can be shown that c_t converges to 0 with probability 1.

Let $y = r_t + \gamma \min(Q_t^B(s_{t+1}, a^*), Q_t^A(s_{t+1}, a^*))$ and $\Delta_t^{BA}(s_t, a_t) = Q_t^B(s_t, a_t) - Q_t^A(s_t, a_t)$, where c_t converges to 0 if Δ^{BA} converges to 0. The update of Δ_t^{BA} at time t is the sum of updates of Q^A and Q^B:

$$\begin{aligned}\Delta_{t+1}^{BA}(s_t, a_t) &= \Delta_t^{BA}(s_t, a_t) + \alpha_t(s_t, a_t)\left(y - Q_t^B(s_t, a_t) - (y - Q_t^A(s_t, a_t))\right) \\ &= \Delta_t^{BA}(s_t, a_t) + \alpha_t(s_t, a_t)\left(Q_t^A(s_t, a_t) - Q_t^B(s_t, a_t)\right) \\ &= (1 - \alpha_t(s_t, a_t))\Delta_t^{BA}(s_t, a_t).\end{aligned} \tag{21}$$

Clearly Δ_t^{BA} will converge to 0, which then shows we have satisfied condition 3 of lemma 1, implying that $Q^A(s_t, a_t)$ converges to $Q_t^*(s_t, a_t)$. Similarly, we get convergence of $Q^B(s_t, a_t)$ to the optimal vale function by choosing $\Delta_t = Q_t^B - Q^*$ and repeating the same arguments, thus proving theorem 1.

B. Overestimation Bias in Deterministic Policy Gradients

If the gradients from the deterministic policy gradient update are unnormalized, this overestimation is still guaranteed to occur under a slightly stronger condition on the expectation of the value estimate. Assume the approximate value function is equal to the true value function, in expectation over the steady-state distribution, with respect to policy parameters between the original policy and in the direction of the true policy update:

$$\begin{aligned}&\mathbb{E}_{s\sim\pi}\left[Q_\theta(s, \pi_{\text{new}}(s))\right] = \mathbb{E}_{s\sim\pi}\left[Q^\pi(s, \pi_{\text{new}}(s))\right] \\ &\forall \phi_{\text{new}} \in [\phi, \phi + \beta(\phi_{\text{true}} - \phi)] \text{ such that } \beta > 0.\end{aligned} \tag{22}$$

Noting that ϕ_{true} maximizes the rate of change of the true value $\Delta_{\text{true}}^\pi = Q^\pi(s, \pi_{\text{true}}(s)) - Q^\pi(s, \pi_\phi(s))$, $\Delta_{\text{true}}^\pi \geq \Delta_{\text{approx}}^\pi$. By the given condition 22 the maximal rate of change of the approximate value must be at least as great $\Delta_{\text{approx}}^\theta \geq \Delta_{\text{true}}^\pi$. Given $Q_\theta(s, \pi_\phi) = Q^\pi(s, \pi_\phi)$ this implies $Q_\theta(s, \pi_{\text{approx}}(s)) \geq Q^\pi(s, \pi_{\text{true}}(s)) \geq Q^\pi(s, \pi_{\text{approx}}(s))$, showing an overestimation of the value function.

Table 3. A complete comparison of hyper-parameter choices between our DDPG and the OpenAI baselines implementation (Dhariwal et al., 2017).

Hyper-parameter	**Ours**	**DDPG**
Critic Learning Rate	10^{-3}	10^{-3}
Critic Regularization	None	$10^{-2} \cdot \|\|\theta\|\|^2$
Actor Learning Rate	10^{-3}	10^{-4}
Actor Regularization	None	None
Optimizer	Adam	Adam
Target Update Rate (τ)	$5 \cdot 10^{-3}$	10^{-3}
Batch Size	100	64
Iterations per time step	1	1
Discount Factor	0.99	0.99
Reward Scaling	1.0	1.0
Normalized Observations	False	True
Gradient Clipping	False	False
Exploration Policy	$\mathcal{N}(0, 0.1)$	OU, $\theta = 0.15, \mu = 0, \sigma = 0.2$

C. DDPG Network and Hyper-parameter Comparison

DDPG Critic Architecture

```
(state dim, 400)
ReLU
(action dim + 400, 300)
ReLU
(300, 1)
```

DDPG Actor Architecture

```
(state dim, 400)
ReLU
(400, 300)
ReLU
(300, 1)
tanh
```

Our Critic Architecture

```
(state dim + action dim, 400)
ReLU
(action dim + 400, 300)
RelU
(300, 1)
```

Our Actor Architecture

```
(state dim, 400)
ReLU
(400, 300)
RelU
(300, 1)
tanh
```

D. Additional Implementation Details

For clarity in presentation, certain implementation details were omitted, which we describe here. For the most complete possible description of the algorithm, code can be found on our GitHub (https://github.com/sfujim/TD3).

Our implementation of both DDPG and TD3 follows a standard practice in deep Q-learning, in which the update differs for terminal transitions. For transitions where the episode terminates by reaching some failure state, and not due to the episode running until the max horizon, the value of $Q(s, \cdot)$ is set to 0 in the target y:

$$y = \begin{cases} r & \text{if terminal } s' \text{ and } t < \text{max horizon} \\ r + \gamma Q_{\theta'}(s', \pi_{\phi'}(s')) & \text{else} \end{cases}$$

For target policy smoothing (Section 5.3), the added noise is clipped to the range of possible actions, to avoid error introduced by using values of impossible actions:

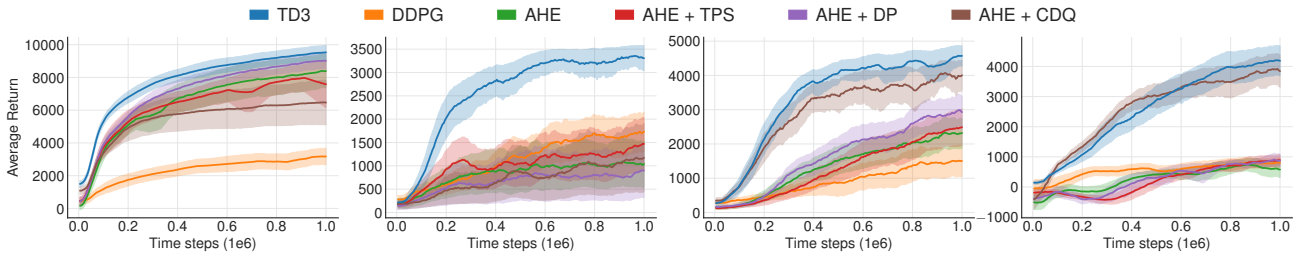
$$y = r + \gamma Q_{\theta'}(s', \text{clip}(\pi_{\phi'}(s') + \epsilon, \text{min action}, \text{max action})),$$
$$\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c).$$

E. Soft Actor-Critic Implementation Details

For our implementation of Soft Actor-Critic (Haarnoja et al., 2018) we use the code provided by the author (https://github.com/haarnoja/sac), using the hyper-parameters described by the paper. We use a Gaussian mixture policy with 4 Gaussian distributions, except for the Reacher-v1 task, where we use a single Gaussian distribution due to numerical instability issues in the provided implementation. We use the environment-dependent reward scaling as described by the authors, multiplying the rewards by 3 for Walker2d-v1 and Ant-v1, and 1 for all remaining environments.

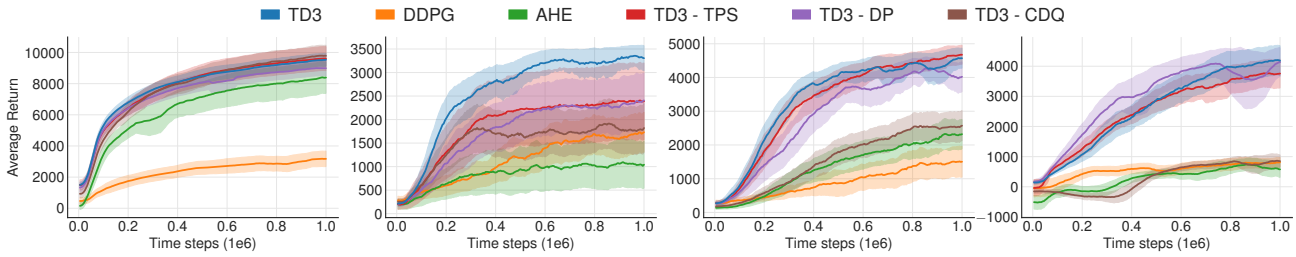
For fair comparison with our method, we train for only 1 iteration per time step, rather than the 4 iterations used by the results reported by the authors. This along with fewer total time steps should explain for the discrepancy in results on some of the environments. Additionally, we note this comparison is against a prior version of Soft Actor-Critic, while the most recent variant includes our Clipped Double Q-learning in the value update and produces competitive results to TD3 on most tasks.

F. Additional Learning Curves



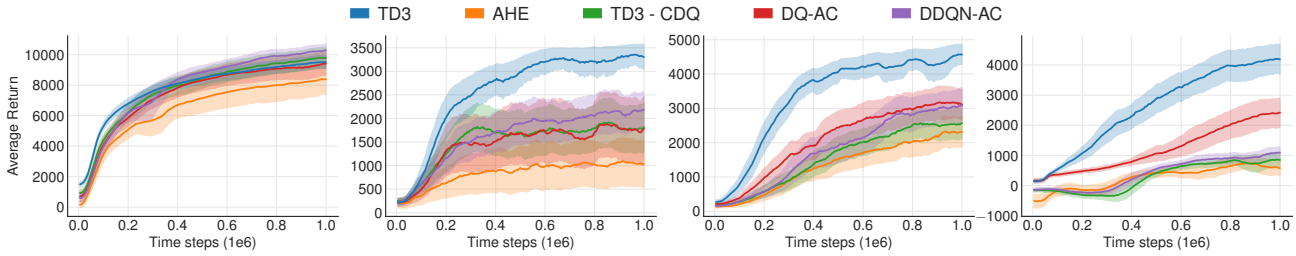
(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1

Figure 6. Ablation over the varying modifications to our DDPG (AHE), comparing the subtraction of delayed policy updates (TD3 - DP), target policy smoothing (TD3 - TPS) and Clipped Double Q-learning (TD3 - CDQ).



(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1

Figure 7. Ablation over the varying modifications to our DDPG (AHE), comparing the addition of delayed policy updates (AHE + DP), target policy smoothing (AHE + TPS) and Clipped Double Q-learning (AHE + CDQ).



(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1

Figure 8. Comparison of TD3 and the Double Q-learning (DQ-AC) and Double DQN (DDQN-AC) actor-critic variants, which also leverage delayed policy updates and target policy smoothing.