Compatible Gradient Approximations for Actor-Critic Algorithms

Baturay Saglam and Dionysis Kalogerias
Department of Electrical and Computer Engineering
Yale University
New Haven, CT 06511
{baturay.saglam, dionysis.kalogerias}@yale.edu

Abstract

Deterministic policy gradient algorithms are foundational for actor-critic methods in controlling continuous systems, yet they often encounter inaccuracies due to their dependence on the derivative of the critic's value estimates with respect to input actions. This reliance requires precise action-value gradient computations, a task that proves challenging under function approximation. We introduce an actor-critic algorithm that bypasses the need for such precision by employing a zeroth-order approximation of the action-value gradient through two-point stochastic gradient estimation within the action space. This approach provably and effectively addresses compatibility issues inherent in deterministic policy gradient schemes. Empirical results further demonstrate that our algorithm not only matches but frequently exceeds the performance of current state-of-the-art methods.

1 Introduction

Reinforcement learning (RL) has established itself as a prominent and effective method for addressing dynamic decision problems [32]. A fundamental strategy in tackling RL problems is via *policy gradient* (PG) techniques [33], grounded in the hypothesis that actions are selected according to a parameterized distribution. PG methods iteratively update policies via stochastic gradient schemes [33], enabling solutions in complex problems featuring continuous state-action spaces under uncertainty. PG methods are often framed within an *actor-critic* framework [11], where an *actor* determines the control policy evaluated by a *critic*. Thus, the efficacy of the actor's improvements critically depends on the quality of the feedback provided by the critic. This interdependence raises nontrivial questions on the efficiency of training strategies for both the actor and the critic [5].

Conventionally, the critic is optimized through temporal difference (TD) learning [31] to accurately predict the expected returns led by the actor's decisions. If the control policy is deterministic, the value estimated by the critic is solely used to compute the *action-value gradient* during policy optimization [5]. This involves computing the gradient of the critic's action-value estimates with respect to the actions selected by the policy. More precisely, the *deterministic policy gradient* (DPG) [29] is determined by chaining the actor's Jacobian to the action-gradient of the critic. However, TD learning optimizes a critic that learns the *action-value*, rather than its gradient. Generally, when function approximation is used for the critic in DPG, the resulting PG may not align with the true gradient. In some cases, it may not even constitute an ascent direction [29]. Nevertheless, this failure mode can be circumvented provided that Q-function estimates are *compatible*, meaning a set of conditions necessary for (approximately) following the true gradient; see, e.g., [29, Theorem 3].

Integration of more complex scenarios requires the use of deep neural network approximations to effectively control and generalize across large state-action spaces [13]. This has led to the development of contemporary DPG-based algorithms grounded in deep neural networks [13, 7, 8].

ICML 2024 Workshop on Foundations of Reinforcement Learning and Control.

However, classical compatibility results call for *linear* critics [29, Theorem 3], contradicting the very foundations of DPG as well as the use of deep network critics, thereby paving the way for potential failure scenarios. Indeed, action differentiation of the Q-function (estimates), i.e., action-value gradients, leaves modern successors to DPG technically unsound [5].

In this paper, we introduce (off-policy) *Compatible Policy Gradient* (oCPG), a continuous control actor-critic algorithm that approximates the action-value gradient using only value estimates from the critic. The proposed method embodies a zeroth-order nature by removing the direct computation of the action-value gradient, while provably addressing the compatibility requirement in DPG. This is achieved through evaluating critics at low-dimensional random perturbations within the action space, resulting in (batch) two-point policy gradient approximations whose distance to the true deterministic policy gradient is controllable at will.

The performance of oCPG is empirically assessed on challenging OpenAI Gym continuous control tasks [4], in perfect (less uncertain) and imperfect (higher uncertainty) environmental conditions. Our numerical results demonstrate that oCPG exhibits robust performance and either matches or surpasses (often substantially) the state-of-the-art in terms of stability and the mean rewards achieved.

2 Related Work

Here, we discuss studies that specifically focus on PG and the compatibility issues in DPG. A more extensive review of related works is available in the supplementary material.

Policy gradient techniques are widely used in RL. Various algorithms for estimating policy gradients have been developed, some focusing solely on the policy [3, 16, 35], while others also incorporate a value function, known as actor-critic methods [17, 27, 28, 11, 20]. Particularly, algorithms based on the deterministic policy gradient [13, 29] rely on the action-gradient from the critic. In function approximation, the accuracy of the critic is crucial [2]; for example, applying compatibility conditions to the critic [29] helps achieve an unbiased estimate of the policy gradient.

In addition to the zeroth-order perspective, GProp [1] and MAGE [5] are the closest known methods to our study that directly focus on optimizing and ensuring accuracy in the action-value gradient computation. GProp modifies traditional TD learning [31] by using an additional neural network that predicts the action-value gradient. MAGE, drawing on recent theoretical [25] and practical [26] insights, trains an additional network to simulate environmental dynamics, employing it as a proxy to sample states for double gradient descent on the Q-function. In stark contrast, our method (CPG) simplifies the DPG approach by directly estimating the action-value through a zeroth-order method, employing two-point evaluations of the existing critic network. This eliminates the need to train additional networks, modify TD learning frameworks, or depend on model-based settings. CPG is adaptable to a broader range of problems, bypassing the requirements for double differentiability of the Q-function and detailed modeling of transition dynamics. Additionally, our approach seamlessly integrates with any TD learning method and policy learning framework, and requires only a *few lines of code* to adapt to modern DPG-based algorithms.

3 Technical Preliminaries

We consider a standardized RL problem in which an agent moves through an environment characterized by a continuous compact state space $\mathcal{S} \subset \mathbb{R}^q$ and takes actions in a finite dimensional action space $\mathcal{A} = \mathbb{R}^p$. Based on its action selection, the agent receives a reward from a (bounded, for simplicity) reward function R, where $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and observes the next state $s' \in \mathcal{S}$. This generic problem is often abstracted by a Markov decision process (MDP) as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where P denotes the environment dynamics, that is, the probabilities $p(s'|s, a)$ of moving to state s' from state s and if action a performed, satisfying the Markov property: $p(s_{t+1}|s_i, a_i; \forall i \leq t) = p(s_{t+1}|s_t, a_t)$. The value $\gamma \in (0, 1)$ is the discount factor that prioritizes future rewards.

The behavior of the agent is described by a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, parameterized by $\theta \in \Theta$, which can be either deterministic or represent a probability distribution. Note that we may occasionally omit the parameter subscript. The objective of the agent is to maximize the expected discounted sum of rewards, quantified by the (state-)value function: $V^{\pi_\theta}(s) \coloneqq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t \sim \pi_\theta(\cdot \mid s_t))|s_0 = s\right]$. By conditioning the state-value function on an action, we also define the action-value function

$Q^{\pi_\theta}(s,a) \coloneqq \mathbb{E}\left[R(s_0,a_0) + \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t \sim \pi_\theta(\cdot \mid s_t)) | s_0 = s, a_0 = a\right]$. Finally, let ρ^0 be the initial state distribution, and let ρ^π be the γ-discounted (improper) state distribution induced by the policy π, defined as $\rho^\pi(s') = \int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t p_0(s) p(s \to s', t, \theta)\, \mathrm{d}s$, where $p(s \to s', t, \theta)$ denotes the probability density (we assume such density exists for simplicity) at state s' after starting from state s and transitioning for t time steps under a policy parameterized by θ.

At the initial state s, by selecting the first action according to the policy and advancing the system, the Q-function becomes equivalent to the value function. Consequently, the objective is expressed as

$$J(\theta) \coloneqq \max_{\theta \in \Theta} \mathbb{E}_{s\sim\rho^0, a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\right].$$

When the search is confined to deterministic policy parameterizations (and we do so hereafter), the problem above reduces to a simpler form, i.e.,

$$J(\theta) = \max_{\pi \in \Pi} \mathbb{E}_{s\sim\rho^0}\left[Q^{\pi_\theta}(s, \pi_\theta(s))\right]. \tag{1}$$

Under mild problem regularity conditions, it has been demonstrated in [29] that the gradient of a deterministic policy may be expressed by

$$\begin{aligned}\nabla J(\theta) &= \frac{1}{1-\gamma} \int_{\mathcal{S}} \rho^{\pi_\theta}(s) \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)\big|_{a=\pi_\theta(s)}\, \mathrm{d}s \\ &= \frac{1}{1-\gamma} \mathbb{E}_{s\sim\rho^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)\big|_{a=\pi_\theta(s)}\right],\end{aligned} \tag{2}$$

where we use the normalizing term $1/(1-\gamma)$ since ρ^{π_θ} is improper. The latter equation is known as the *deterministic policy gradient theorem* [29], which has been instrumental in the development state-of-the-art policy gradient methods essentially relying on stochastic approximation, such as Deep Deterministic Policy Gradient (DDPG) [13] and Twin Delayed DDPG (TD3) [7].

4 Off-Policy Compatible Policy Gradient

4.1 Problem Statement

Using a function-approximated Q-function in the DPG algorithm, as expressed in (2), can disrupt the accuracy of the approximated PG unless specific conditions are met [29, Theorem 3]. A critical requirement in [29] is that the gradient of the Q-function must be linear, i.e., $\nabla_a Q_\omega(s,a)|_{a=\pi_\theta(s)} = \nabla_\theta \pi_\theta(s)^\top \omega$ for parameters ω. However, it has been demonstrated that scaling to large and complex state-action spaces inevitably requires the use of more expressive representations, such as deep neural networks [13]. This necessity compromises compatibility and results in deterministic policy gradients that may not accurately follow the true gradient [29].

This failure mode has recently been linked to the need to minimize the norm of the action-value gradient of the policy evaluation error, rather than its value, in order to reduce the approximation error of the computed PG [5, Proposition 3.1]. Aligning with this insight, employing double backpropagation through the TD error may address this issue in scenarios where model-based approaches are feasible [5]. However, model-free RL is often preferred due to its effectiveness in environments with complex or unknown dynamics, offering enhanced flexibility and robustness for direct policy learning through interactions with the environment [32].

While several studies have pursued the objective of overcoming the incompatibility of the critic [5, 1], our goal herein is to *consistently* approximate the deterministic policy gradient ∇J as

$$\boxed{\hat{\nabla} J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s\sim\rho^{\pi}}\left[\nabla_\theta \pi_\theta(s) \hat{\nabla}_a Q^{\pi_\theta}(s,a))\big|_{a=\pi_\theta(s)}\right],} \tag{3}$$

where $\hat{\nabla}_a Q^{\pi_\theta}(s,a)$ represents an appropriately designed approximation of $\nabla_a Q^{\pi_\theta}(s,a)$, resulting in a $\hat{\nabla} J$ that is *provably compatible*, *general* and *implementable in a model-free manner*. To the best of our knowledge, no study has yet proposed a gradient surrogate (i.e., $\hat{\nabla} J$) that possesses all aforementioned features *simultaneously*, in a model-free deep RL setting. As we empirically demonstrate later (Section 5), such an approach in fact results in substantial operational benefits over the state-of-the-art.

4.2 Provably Compatible Policy Gradient Approximations

To begin, we temporarily assume that Q^{π_θ} is available and, for a *smoothing parameter* $\mu > 0$, we consider the *μ-smoothed Q-function* defined as

$$Q_\mu^{\pi_\theta}(s,a) \coloneqq \mathbb{E}_{\boldsymbol{u}}\big[Q^{\pi_\theta}(s, a+\mu\boldsymbol{u})\big], \quad \boldsymbol{u} \sim \mathcal{N}(0, I_p). \tag{4}$$

Smoothing may be equivalently thought of as enforcing exploration, and the random perturbation $\mu\boldsymbol{u}$ may be thought of as the stochastic part of a standard *Gaussian policy* $\pi_\theta^\mu(\cdot \mid s) \coloneqq \pi_\theta(s) + \mu\boldsymbol{u}$, with $\boldsymbol{u} \sim \mathcal{N}(0, I_p)$. Here, it will be useful to think of such a Gaussian policy as a deterministic policy perturbed by leveraging low-dimensional noise in action space.

Implementing (Gaussian) smoothing on deterministic actions aligns with approaches used in seminal works such as [13, 7]. As mentioned above, this technique introduces stochasticity in action selection, which prevents premature convergence to suboptimal policies by promoting exploration of a broader range of state-action pairs [32, 7]. In any case, though, the ultimate goal is to discover a near-optimal *deterministic* control policy that near-solves (1). In our context, we achieve this by learning a deterministic policy through trajectories generated by a smoothed/perturbed policy, with all subsequent evaluations based on deterministic actions.

Our development will be relying on the following basic result from [19], which is central in the development and analysis of zeroth-order optimization methods, and establishes that the gradient of the smoothed functions such as that in (4) may be evaluated in a model-free fashion, through batches of two-point evaluations of the function itself.

Proposition 1 [19]**.** *Let $f : \mathbb{R}^p \to \mathbb{R}$ be a bounded function. For every $\mu > 0$, the smoothed function $f_\mu(x) \coloneqq \mathbb{E}_{\boldsymbol{u}}\big[f(x+\mu\boldsymbol{u})\big]$, $\boldsymbol{u} \sim \mathcal{N}(0, I_p)$ is well-defined, differentiable and its gradient admits the representation*

$$\nabla f_\mu(x) = \mathbb{E}_{\boldsymbol{u}\sim\mathcal{N}(0,I_n)}\left[\frac{f(x+\mu\boldsymbol{u}) - f(x)}{\mu}\boldsymbol{u}\right], \quad \forall x \in \mathbb{R}^n.$$

Further, if f is G-smooth (i.e., with G-Lipschitz Gradients), it holds that

$$\sup_x \|\nabla f_\mu(x) - \nabla f(x)\| \leq \mu G\sqrt{p}.$$

Proposition 1 easily motivates the Q-function *zeroth-order gradient approximation*

$$\nabla_a Q_\mu^{\pi_\theta}(s,a) = \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q^{\pi_\theta}(s, a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,a)}{\mu}\boldsymbol{u}\right],$$

resulting in a deterministic policy gradient approximation reading (cf. 3)

$$\hat{\nabla}^\mu J(\theta) \coloneqq \frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho^{\pi_\theta},\boldsymbol{u}\sim\mathcal{N}(0,I_p)}\left[\nabla_\theta\pi_\theta(s)\frac{Q^{\pi_\theta}(s, a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,a)}{\mu}\boldsymbol{u}\bigg|_{a=\pi_\theta(s)}\right].$$

While $\hat{\nabla}^\mu J$ is a genuine and consistent model-free approximation of the deterministic policy gradient ∇J (through Proposition 1), it requires access to (evaluations of) the Q-function itself, the latter being both unknown (in the vast majority of cases) and also hard, or at least non-trivial to sample.

Approximating the Q-function Following the actor-critic paradigm [11], we approximate the Q-function by a parameterized learning representation Q_ψ (the critic), for instance, a deep neural network (i.e., Q-network), similarly to the control policy π_θ (or actor). We denote Q-network parameters abstractly with the subscript ψ (different than the smoothing parameter μ). Then, we propose the parameterized deterministic policy gradient approximation

$$\boxed{\hat{\nabla}^{\mu,\psi} J(\theta) \coloneqq \frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho^{\pi_\theta},\boldsymbol{u}}\left[\nabla_\theta\pi_\theta(s)\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu}\boldsymbol{u}\bigg|_{a=\pi_\theta(s)}\right].} \tag{5}$$

Given (5), a natural question is how the representation Q_ψ should be trained, in order for $\hat{\nabla}^{\mu,\psi} J$ to achieve a small approximation error as compared with the true policy gradient ∇J. In other words, instead of approximating in value, we would like to choose ψ such that the representation

Q_ψ is compatible to Q^{π_θ} (in the standard sense of [29]). To this end, let us first define the *perturbed representation error*

$$\varepsilon^{\pi_\theta}_{\mu,\psi} \coloneqq \sqrt{\mathbb{E}_{s,\boldsymbol{u}}\big[\big|Q_\psi(s,\pi_\theta(s)+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,\pi_\theta(s)+\mu\boldsymbol{u})\big|^2\big]},$$

where θ, μ and ψ are given. We have the following result (see the supplementary material for a proof).

Theorem 1 (Compatible Policy Gradient). *Fix parameters θ, μ, ψ, and let $B > 0$ be such that $\sup_{(s,\theta)\in\mathcal{S}\times\Theta} \|\nabla_\theta \pi_\theta(s)\| \leq B$. Then the gradient approximation $\hat{\nabla}^{\mu,\psi} J$ satisfies, for every $\theta \in \Theta$,*

$$\|\hat{\nabla}^{\mu,\psi} J(\theta) - \nabla J(\theta)\| \leq \frac{B}{1-\gamma}\left[\frac{\varepsilon^{\pi_\theta}_{\mu,\psi}}{\mu}\sqrt{p} + \mathbb{E}_s\big[\big\|\nabla_a Q^{\pi_\theta}_\mu(s,a) - \nabla_a Q^{\pi_\theta}(s,a)\big\|_{a=\pi_\theta(s)}\big]\right].$$

In particular, if $Q^{\pi_\theta}(s,\cdot)$ is itself G-smooth (uniformly), it follows that, for every $\theta \in \Theta$,

$$\|\hat{\nabla}^{\mu,\psi} J(\theta) - \nabla J(\theta)\| \leq \frac{B\sqrt{p}}{1-\gamma}\left[\frac{\varepsilon^{\pi_\theta}_{\mu,\psi}}{\mu} + G\mu\right].$$

The usefulness of the result just stated is twofold: One the one hand, Theorem 1 cleanly quantifies the performance of $\hat{\nabla}^{\mu,\psi} J$ in terms of smoothing (μ) and function approximation (ψ) errors. On the other hand, Theorem 1 reveals *how* ψ can be chosen, so that the gradient approximation error achieved by adopting $\hat{\nabla}^{\mu,\psi} J$ is controlled *at will*. To see this, observe that if we take for clarity the case of a smooth Q-function, Theorem 1 implies that

$$\inf_\psi \|\hat{\nabla}^{\mu,\psi} J(\theta) - \nabla J(\theta)\| \leq \frac{B\sqrt{p}}{1-\gamma}\left[\frac{\inf_\psi \varepsilon^{\pi_\theta}_{\mu,\psi}}{\mu} + G\mu\right].$$

In other words, *for a fixed level of smoothing $\mu > 0$* (and at each actor instance indexed by θ), one can *optimally choose (i.e., train)* the representation Q_ψ such that the perturbed representation error $\varepsilon^{\pi_\theta}_{\mu,\psi}$ is made as small as possible, theoretically achieving an *arbitrarily small* gradient approximation error (by properly choosing an expressive enough class for the representation Q_ψ). Motivated from the discussion above, we hereafter refer to $\hat{\nabla}^{\mu,\psi} J$ as a *Compatible Policy Gradient* (CPG).

In passing, we would like to emphasize that optimizing $\varepsilon^{\pi_\theta}_{\mu,\psi}$ over ψ (for fixed θ and μ) in fact justifies the use of Gaussian exploration in off-policy methods such as DDPG [13, Algorithm 1, lines 8-13] and TD3 [7, Algorithm 1, lines 6-11], where the ultimate goal is deterministic policy optimization. However, both DDPG and TD3 use backpropagation in their corresponding gradient approximations to compute the action-value gradient. Theorem 1 implies that, in fact, this should not be the case, and that (5) should be used as a (compatible) gradient approximation of choice.

4.3 Application to Off-Policy Deep Reinforcement Learning

To devise an efficient learning framework centered on CPG, we incorporate various standard components from existing literature.

Clipped Double Q-learning [7] Based on our discussion above, to exploit $\hat{\nabla}^{\mu,\psi} J$ as a gradient surrogate (and thus harness the benefits of Theorem 1), we should learn a Q_{ψ^*} such that $\psi^* \in \arg\min_\psi \varepsilon^{\pi_\theta}_{\mu,\psi}$ (iteration-wise for fixed θ and μ). However, learning such a Q_{ψ^*} assumes the feasibility of obtaining unbiased estimates of the Q-function at any state-action (perturbation) pair, as well as sampling from the discounted state distribution ρ^π, are feasible tasks. Specifically for obtaining unbiased Q-value estimates in the context of deep RL, one can employ the practical clipped double Q-learning algorithm [7] to learn the Q-network (in conjunction with an experience replay buffer – see next paragraph), which has been shown to eliminate the estimation bias effectively.

Off-Policy Learning Our selection of an off-policy learning approach is driven by its potential for high sample efficiency, notably through the use of an experience replay buffer [14]. The agent stores samples as transition tuples $(s, a, R(s,a), s')$ in the buffer and periodically samples a minibatch of these transitions for policy and Q-network updates. Building upon this, CPG, although theoretically on-policy, is implemented in an off-policy framework. This approach, however, inherently involves a distributional shift, as the samples used may significantly diverge from the current agent's policy [32]. Despite this, our method draws on the strategy employed by modern off-policy algorithms [29, 13, 7, 8], which simplifies complexities by approximating off-policy samples as on-policy.

Algorithm 1 Off-Policy Compatible Policy Gradient (oCPG)

Initialize the policy network π_θ, and critic networks Q_{ψ_1}, Q_{ψ_2} with random parameters θ, ψ_1, ψ_2
Initialize target networks $\theta' \leftarrow \theta$, $\psi_1' \leftarrow \psi_1$, $\psi_2' \leftarrow \psi_2$
Initialize the experience replay buffer: $\mathcal{B} = \emptyset$
for $t = 1$ to T **do**
    Select action with Gaussian exploration noise $a = \pi_\theta(s) + \mu\boldsymbol{u}$, where $\boldsymbol{u} \sim \mathcal{N}(0, I_p)$
    Execute action a, receive reward $r = R(s, a)$, and observe new state s'
    Store the collected transition tuple in the replay buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup (s, a, r, s')$
    Sample a minibatch of N transitions from the replay buffer: $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}') \sim \mathcal{B}$
    $\tilde{\mathbf{a}} \leftarrow \pi_{\theta'}(\mathbf{s}') + \epsilon$, where $\{\epsilon_i \sim \text{clip}\left(\mathcal{N}(0, \sigma^2 I_p), -c, c\right)\}_{i=1}^N$ ▷ Clipped Double Q-learning
    $\mathbf{y} \leftarrow \mathbf{r} + \gamma \min_{i=1,2} Q_{\psi_i'}(\mathbf{s}', \tilde{\mathbf{a}})$ ▷ Clipped Double Q-learning
    Update critics: $\psi_i \leftarrow \text{argmin}_{\psi_i} \frac{1}{N} \sum_i \left(\mathbf{y}_i - Q_{\psi_i}(\mathbf{s}_i, \mathbf{a}_i)\right)^2$ ▷ Clipped Double Q-learning
    if t mod d **then** ▷ Delayed policy updates
        Update the policy by CPG:

$$\hat{\nabla}_\beta^{\mu,\psi} J(\theta) = \frac{1}{N(1-\gamma)} \sum_i \nabla_\theta \pi_\theta(\mathbf{s}_i) \frac{Q_{\psi_1}(\mathbf{s}_i, \mathbf{a}_i + \mu\boldsymbol{u}_i) - Q_{\psi_1}(\mathbf{s}_i, \mathbf{a}_i)}{\mu} \boldsymbol{u}_i \Big|_{\substack{\mathbf{a}_i = \pi_\theta(\mathbf{s}_i) \\ \boldsymbol{u}_i \sim \mathcal{N}(0, I_p)}}$$

        Update target networks:

$$\theta' \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta'$$
$$\psi_i' \leftarrow \tau \cdot \psi_i + (1 - \tau) \cdot \psi_i'$$

    end if
end for

Combining the components introduced, we present our proposed CPG method in an off-policy deep RL setting, implementing stochastic gradient ascent via the implementable gradient approximation

$$\hat{\nabla}^{\mu,\psi} J(\theta) \approx \hat{\nabla}_\beta^{\mu,\psi} J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho^\beta, \boldsymbol{u}} \left[\nabla_\theta \pi_\theta(s) \frac{Q_\psi(s, a + \mu\boldsymbol{u}) - Q_\psi(s, a)}{\mu} \boldsymbol{u} \Big|_{a = \pi_\theta(s)} \right],$$

where Q_ψ is obtained as described above, and β denotes the set of policies that collected the off-policy samples (referring to the replay buffer), which can also be on-policy. Acknowledging the limitations of this approximation, we focus on the immediate performance of our algorithm. A detailed treatment for the "off-policyness" of these samples (see, e.g., [18, 21]), is deferred for future research.

Pseudocode for off-policy CPG (oCPG) is provided in Algorithm 1. The inclusion of dual Q-networks, target parameters, and delayed policy updates draws from the structure of Clipped Double Q-learning. We also consistently optimize the policy with respect to the first Q-network to ensure stability.

5 Results

We evaluate the performance of oCPG by benchmarking it against state-of-the-art PG-based actor-critic algorithms: TD3 [7] and Soft Actor-Critic (SAC) [8]. Notably, as TD3 is a variant of DPG, comparing it with CPG serves as an effective test of our proposed PG method against the traditional DPG. Similarly, since SAC is an extension of the standard stochastic PG algorithm, our comparisons with SAC offer insights into CPG relative to stochastic PG.

For an additional benchmark, we also considered testing against MAGE [5]. However, this comparison would be unfair due to significant differences in operational domains and structures. MAGE, being a model-based algorithm, fundamentally differs in sample efficiency and adaptability, and comparing our model-free oCPG with MAGE could lead to biased evaluation. For example, we in fact attempted to use MAGE as a benchmark but found it impractically slow due to its computationally demanding environment modeling, as compared to all oCPG, TD3, and SAC.

5.1 Experimental Setup

Evaluation The performance is assessed on a set of challenging continuous control tasks from the MuJoCo suite [34], interfaced by OpenAI Gym [4]. Each algorithm is run for 1 million time steps,

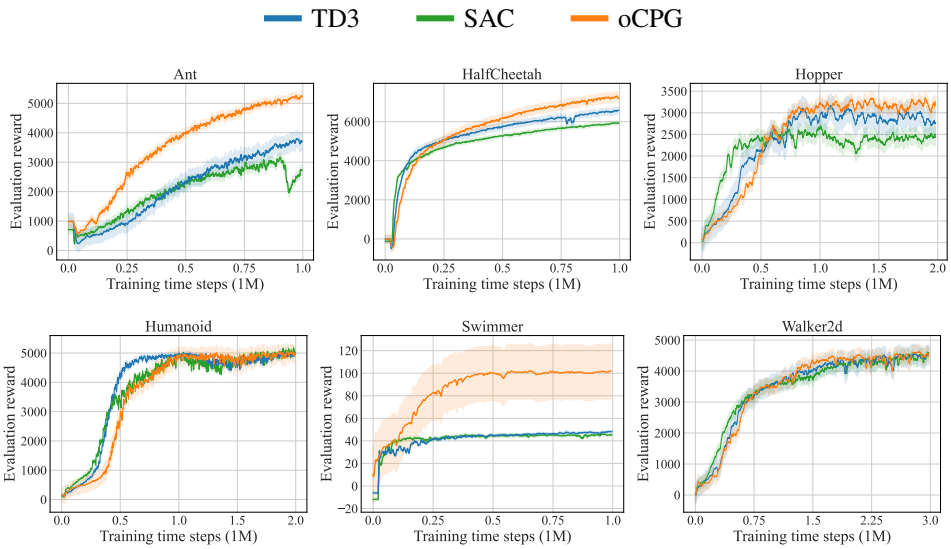


Figure 1: Learning curves for the benchmark MuJoCo environments, averaged over 10 random seeds. The shaded area represents the 95% confidence interval of the mean performance.

except for Hopper, Humanoid, and Walker2d, which are extended to more steps. This adjustment is made because we observed a continuing upward trend in the rewards at 1 million time steps, indicating that more time was needed for the algorithms to reach a steady state. The agent is evaluated in a separate evaluation environment without exploration noise, i.e., the system is controlled by deterministic actions. Each evaluation occurs every 1000 time step for 10 episodes, and the average reward is recorded. Results are reported over 10 random seeds, including the Gym simulator, network initialization, and any other stochastic processes involved.

Implementation and hyperparameters For SAC, we adhere to the hyperparameter setup as described in [8]. The parameters for TD3 are imported from the authors' GitHub repository[1], which provides the tuned setting. To mitigate dependency on initial policy parameters, we implement a purely exploratory policy during the first 25,000 time steps in all environments, a strategy shown to enhance sample complexity [7]. Additionally, in CPG, we fine-tune the policy learning rate to account for the PG being scaled by a smaller constant $1/(1-\gamma)$, typically around 0.01. Finally, we tune the standard deviation of action perturbations (or the Gaussian exploration noise) and find that $\mu = 0.125$ perform best in a variety of tasks, while TD3 and SAC use $\mu = 0.1$ by default. A comprehensive description of the experimental setup and implementation details is available in the supplementary material.

5.2 Discussion

The results are presented in Figure 1 and Table 5.2 includes a Welch t-test on the maximum rewards achieved to assess the statistical significance of the performance differences. Furthermore, we report the evaluation results under imperfect environmental conditions in the supplementary material, where limitations of our approach are identified and discussed.

In most environments, oCPG achieves faster convergence to higher final rewards. Although the performance curves are intertwined, making it difficult to discern a clear leader visually, the statistical tests reported in Table 5.2 confirm that oCPG is the best-performing algorithm in three out of six environments. In the remaining cases, it is on par with the baselines.

While oCPG generally achieves faster convergence to higher rewards, we note slower convergence in the Humanoid environment compared to TD3 and SAC. The state-action dimension in Humanoid is aggregated to 393, whereas the Q-networks in our experiments have only 256 hidden units per layer.

[1] `https://github.com/sfujim/TD3`

Environment	TD3	SAC	oCPG
Ant	4225.080 ± 835.236	3945.508 ± 673.329	5463.987 ± 202.699
HalfCheetah	6744.592 ± 671.739	6074.230 ± 405.968	7661.154 ± 650.760
Hopper	3620.488 ± 137.280	3585.355 ± 186.848	3634.739 ± 113.770
Humanoid	5309.081 ± 148.741	5567.597 ± 141.133	5328.928 ± 130.136
Swimmer	59.806 ± 9.477	58.396 ± 10.990	113.478 ± 71.670
Walker2d	4842.336 ± 723.696	4937.606 ± 760.053	4953.588 ± 298.321

Table 1: Average performance on the benchmark MuJoCo environments over 10 random seeds, where ± denotes a 95% confidence interval. The highest performance, statistically superior to the others, is highlighted. Additionally, the highest performances shared by multiple algorithms are indicated, based on Welch's t-test with a significance level of 0.05.

Neural networks can be sensitive to input noise, which is amplified when the input size exceeds the number of hidden units. This issue is likely intensified by our smoothing technique on line 13 in Algorithm 1, which introduces input noise to the neural network, potentially hindering performance in environments with large state-action spaces. However, this could be mitigated by using wider networks. Despite these challenges, oCPG still achieves higher maximum rewards, as confirmed in Table 5.2. Additionally, while the performance in the Swimmer task is statistically significant, the rewards display high variance. This variance stems from some seeds where oCPG converged to suboptimal levels, resulting in broad confidence intervals.

In our comparisons with TD3 and SAC, discrepancies in reported reward levels compared to those in the original studies may arise from variations in environmental stochasticity and the version of the MuJoCo simulator used. Nonetheless, consistent performance improvements over multiple random seeds provide a robust basis for assessing the efficacy of our proposed method. This approach follows established deep RL experimentation standards [9], focusing on comparative analysis rather than absolute performance metrics.

Overall, the findings from our discussion indicate that CPG provides a robust and state-of-the-art alternative to traditional PG methods, especially in scenarios where the accuracy of action-value gradient computations is less reliable, effectively resolving the incompatibility issue.

6 Conclusions and Future Work

In this work, we have presented a policy gradient technique for actor-critic algorithms, Compatible Policy Gradient (CPG), which introduces a zeroth-order approximation method to estimate the action-value gradient in deterministic policy gradient (DPG) algorithms. Our approach circumvents the need for explicit action-value gradient computation by employing stochastic two-point gradient estimation by leveraging low-dimensional perturbations in action space. This effectively resolves compatibility issues often encountered in conventional DPG methods with function approximation. Empirical evaluations on various continuous control tasks from the OpenAI Gym suite demonstrate that CPG exhibits competitive performance as compared with the state-of-the-art methods.

The implications of our findings are twofold. First, they underscore the potential of zeroth-order (gradient-free) methods in overcoming the limitations of dependency on accurate derivative computations in policy optimization. Second, the efficacy of CPG in complex continuous control environments suggests that similar approaches could be beneficial in other domains of reinforcement learning where the dynamics are highly uncertain or the system models are imperfect.

Future Work Developing an off-policy correction scheme for CPG could further improve its effectiveness. We believe that a scheme based on a ratio of the probabilities of the current agent and behavioral policies (collecting transitions) could provide an effective correction. Additionally, exploring the potential of the CPG approach in model-based RL scenarios could provide deeper insights into its versatility and robustness.

References

[1] David Balduzzi and Muhammad Ghifary. Compatible value gradients for reinforcement learning of continuous deep policies, 2015.

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributional policy gradients. In *International Conference on Learning Representations*, 2018.

[3] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, November 2001.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[5] Pierluca D' Oro and Wojciech Jaśkowski. How to learn a useful critic? model-based action-gradient-estimator policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 313–324. Curran Associates, Inc., 2020.

[6] John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[7] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[9] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[12] Yuheng Lei, Jianyu Chen, Shengbo Eben Li, and Sifa Zheng. Zeroth-order actor-critic, 2022.

[13] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

[14] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321, May 1992.

[15] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[16] Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[17] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[18] Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[19] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, Apr 2017.

[20] D.V. Prokhorov and D.C. Wunsch. Adaptive critic designs. *IEEE Transactions on Neural Networks*, 8(5):997–1007, 1997.

[21] Baturay Saglam, Doğan Can Çiçek, Furkan Burak Mutlu, and Suleyman Kozat. Mitigating off-policy bias in actor-critic methods with one-step q-learning: A novel correction approach. *Transactions on Machine Learning Research*, 2024.

[22] Baturay Saglam and Suleyman S. Kozat. Deep intrinsically motivated exploration in continuous control. *Machine Learning*, 112(12):4959–4993, Dec 2023.

[23] Baturay Saglam, Furkan B. Mutlu, and Suleyman S. Kozat. An optimistic approach to the temporal difference error in off-policy actor-critic algorithms. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 875–883, 2022.

[24] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[25] Saeed Saremi. On approximating ∇f with neural networks. *ArXiv*, abs/1910.12744, 2019.

[26] Saeed Saremi and Aapo Hyvärinen. Neural empirical bayes. *Journal of Machine Learning Research*, 20(181):1–23, 2019.

[27] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[29] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR.

[30] Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, Mar 2000.

[31] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.

[32] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[33] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[34] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[35] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, May 1992.

A Extended Related Work: Review of Zeroth-order Optimization Methods

The incompatibility of the critic in DPG could be resolved by employing zeroth-order optimization (ZOO), which replaces the action-value gradient computation with an approach that does not require any (first-order) gradient information. These techniques treat the RL problem as a black-box optimization by ignoring the underlying MDP structures and directly searching for the optimal policy without gradient computations. Recent ZOO methods have proven to be competitive on common RL benchmarks [15, 24, 12], benefiting especially from not being constrained to differentiable policies. Despite these advantages, ZOO techniques suffer from high sample complexity and significant variance in gradient updates, due to their disregard for the MDP structure [12]. These issues are further amplified with the increase in perturbation noise scale, potentially escalating dramatically as parameter counts can reach into the millions [6, 12]. Moreover, while parameter space perturbations can induce diverse behaviors viewed as exploration, they are less beneficial in DPG methods, which are typically framed in an off-policy context [29, 13, 7]. Exploration and policy learning are distinct processes in off-policy methods, making parameter-based zeroth-order techniques less effective for enhancing exploration, primarily confined to on-policy and stochastic policies [30]. Contrasting with parameter space-based ZOO techniques, our approach operates in the action space and involves low-dimensional perturbations. Perturbing the action also incorporates Gaussian exploration [7], which has been empirically shown to outperform parameter space exploration in off-policy settings [23, 22]. Lastly, our objective is not merely to devise a pure zeroth-order PG method but to synthesize the strengths of both ZOO and first-order approaches, employing a differentiable policy whose Jacobian is chained to the zeroth-order approximation of the action-value gradient.

B Proof of Theorem 1

We may write, for every $\theta \in \Theta$,

$$
\begin{aligned}
&(1-\gamma)\|\hat{\nabla} J(\theta) - \nabla J(\theta)\| \\
&= \left\| \mathbb{E}_{s,\boldsymbol{u}}\left[\nabla_\theta \pi_\theta(s) \frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu} \boldsymbol{u} \right]\Big|_{a=\pi_\theta(s)} - \mathbb{E}_s\left[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a) \right]_{a=\pi_\theta(s)} \right\| \\
&= \left\| \mathbb{E}_s\left[\nabla_\theta \pi_\theta(s) \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu} \boldsymbol{u} - \nabla_a Q^{\pi_\theta}(s,a) \right]\Big|_{a=\pi_\theta(s)} \right] \right\| \\
&\leq \mathbb{E}_s\left[\left\| \nabla_\theta \pi_\theta(s) \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu} \boldsymbol{u} - \nabla_a Q^{\pi_\theta}(s,a) \right]\Big|_{a=\pi_\theta(s)} \right\| \right] \\
&\leq B \mathbb{E}_s\left[\left\| \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu} \boldsymbol{u} - \nabla_a Q^{\pi_\theta}(s,a) \right] \right\|\Big|_{a=\pi_\theta(s)} \right],
\end{aligned}
$$

where we have used Jensen (norms are convex), (matrix) Cauchy-Schwarz, and the assumption that $\sup_{s,\theta} \|\nabla_\theta \pi_\theta(s)\| \leq B$. In passing, we note that this expression remains valid as $\mu \to 0$ in which case we obtain the *deterministic policy gradient error* reading

$$
(1-\gamma)\|\hat{\nabla} J(\theta) - \nabla J(\theta)\| \leq B \mathbb{E}_s\left[\left\| \nabla_a Q_\psi(s,a) - \nabla_a Q^{\pi_\theta}(s,a) \right\|_{a=\pi_\theta(s)} \right].
$$

We further have, for every $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$
\begin{aligned}
&\left\| \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu} \boldsymbol{u} \right] - \nabla_a Q^{\pi_\theta}(s,a) \right\| \\
&= \left\| \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q_\psi(s,a)}{\mu} \boldsymbol{u} \right] - \nabla_a Q_\mu^{\pi_\theta}(s,a) + \nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a) \right\| \\
&= \left\| \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s, a+\mu\boldsymbol{u})}{\mu} \boldsymbol{u} \right] + \nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a) \right\| \\
&\leq \left\| \mathbb{E}_{\boldsymbol{u}}\left[\frac{Q_\psi(s, a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s, a+\mu\boldsymbol{u})}{\mu} \boldsymbol{u} \right] \right\| + \left\| \nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a) \right\| \\
&\leq \frac{1}{\mu} \sqrt{\mathbb{E}_{\boldsymbol{u}}[|Q_\psi(s, a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s, a+\mu\boldsymbol{u})|^2]} \sqrt{\mathbb{E}_{\boldsymbol{u}}[\|\boldsymbol{u}\|^2]} + \left\| \nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a) \right\|
\end{aligned}
$$

$$= \frac{\sqrt{p}}{\mu}\sqrt{\mathbb{E}_{\boldsymbol{u}}[|Q_\psi(s,a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,a+\mu\boldsymbol{u})|^2]} + \big\|\nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a)\big\|,$$

where we have used Cauchy-Schwarz in the penultimate line. Taking expectations over s, we obtain

$$\begin{aligned}
&\frac{(1-\gamma)}{B}\|\hat{\nabla}J(\theta) - \nabla J(\theta)\| \\
&\leq \mathbb{E}_s\left[\frac{\sqrt{p}}{\mu}\sqrt{\mathbb{E}_{\boldsymbol{u}}[|Q_\psi(s,a+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,a+\mu\boldsymbol{u})|^2]} + \big\|\nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a)\big\|\Big|_{a=\pi_\theta(s)}\right] \\
&= \frac{\sqrt{p}}{\mu}\mathbb{E}_s\left[\sqrt{\mathbb{E}_{\boldsymbol{u}}[|Q_\psi(s,\pi_\theta(s)+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,\pi_\theta(s)+\mu\boldsymbol{u})|^2]}\right] \\
&\qquad\qquad + \mathbb{E}_s\left[\big\|\nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a)\big\|\big|_{a=\pi_\theta(s)}\right] \\
&\leq \frac{\sqrt{p}}{\mu}\sqrt{\mathbb{E}_{s,\boldsymbol{u}}[|Q_\psi(s,\pi_\theta(s)+\mu\boldsymbol{u}) - Q^{\pi_\theta}(s,\pi_\theta(s)+\mu\boldsymbol{u})|^2]} \\
&\qquad\qquad + \mathbb{E}_s\left[\big\|\nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a)\big\|\big|_{a=\pi_\theta(s)}\right] \\
&= \frac{\sqrt{p}}{\mu}\varepsilon_{\mu,\psi}^{\pi_\theta} + \mathbb{E}_s\left[\big\|\nabla_a Q_\mu^{\pi_\theta}(s,a) - \nabla_a Q^{\pi_\theta}(s,a)\big\|\big|_{a=\pi_\theta(s)}\right],
\end{aligned}$$

where we have used Jensen in the third line (the root function is concave). The rest of the theorem follows trivially from Proposition 1, and we are done.

C Experimental Details

In Section C.1, we provide details of the hyperparameters and network architectures. Additionally, the experimental setup, such as simulation and performance assessment, is examined in Section C.2.

C.1 Hyperparameters and Neural Networks

Architecture All algorithms use two Q-networks following the Clipped Double Q-learning algorithm, and one actor network. Each network consists of two hidden layers with 256 units each, using ReLU activations. The Q-networks process state-action pairs (s, a) and output a scalar value. The actor network inputs state s and outputs an action a using a `tanh` activation, scaled to the action range of the environment.

Hyperparameters The Adam optimizer [10] is used for training the networks, with a learning rate set to 3×10^{-4}. When sampling from the replay buffer, minibatches of 256 samples are employed. We update the target networks using polyak averaging, with a rate of $\tau = 0.005$. This process follows the update rule: $\theta' \leftarrow 0.995 \times \theta' + 0.005 \times \theta$.

Hyperparameter optimization For SAC, hyperparameter optimization was performed, particularly adjusting the reward scale for the Swimmer task since the original article did not specify this value. We experimented with reward scales of $\{5, 10, 20\}$ and found that a factor of 5 yielded the best performance in this environment. Apart from this, all algorithms strictly followed the parameter settings and methodologies from their original publications or the latest versions of their code on GitHub. Specifically, SAC adhered to the hyperparameters from its original paper, except for an increase in exploration steps to 25,000. For TD3, we used the version from the author's GitHub repository[1], which introduces slight changes in parameters compared to the paper. Notably, this version increases the start steps to 25,000 and the batch size to 256 across all environments, leading to improved results.

For CPG, we tested values of $\mu = \{0.25, 0.5, 0.75, 0.9, 1.0, 1.25, 1.5\}$ on the Hopper and Walker2d tasks and determined that $\mu = 1.25$ was most effective. Due to the selected small μ value in the denominator of the PG, it was necessary to adjust the learning rate for the policy in CPG. For the Hopper and Walker2d environments, we found that an actor learning rate of 5×10^{-5} was optimal. The complete hyperparameter settings are listed in Table 2.

Table 2: Hyperparameters used in the experiments.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate (all networks)	3×10^{-4}
Learning rate (CPG actor only)	5×10^{-5}
Minibatch size	256
Discount factor γ	0.99
Target update rate	0.005
Initial exploration steps	25,000
CPG exploration noise μ_{CPG}	0.125
TD3 exploration policy μ_{TD3}	0.1
TD3 target policy noise σ	0.2
TD3 policy noise clipping	$(-0.5, 0.5)$
SAC log-standard deviation clipping	$(-20, 2)$
SAC log constant	10^{-6}
SAC reward scale (except Humanoid)	5
SAC reward scale (Humanoid)	20

C.2 Experimental Setup

Simulation Environments All agents are evaluated using continuous control benchmarks from the MuJoCo[2] physics engine, accessed through OpenAI Gym[3] with `v3` environments. We maintained the original state-action spaces and reward functions in these environments to ensure reproducibility and fair comparison with existing empirical results. Each environment features a multi-dimensional action space within the range of [-1, 1], except for Humanoid, which has a range of [-0.4, 0.4].

Terminal Transitions In computing the target Q-value, we apply a discount factor of $\gamma = 0.99$ for non-terminal transitions, and zero for terminal ones. A transition is considered as terminal only if it ends due to a termination condition, such as a failure or exceeding the time limit or fall of the agent.

Evaluation Evaluations are performed every 1000 time steps, with each evaluation representing the average reward across 10 episodes. These evaluations use the deterministic policy from oCPG and TD3 without exploration noise, and the deterministic mean action from SAC. To minimize variation due to different seeds, a new environment with a fixed seed (the training seed plus a constant) is used for each evaluation. This approach ensures consistent initial start states for each evaluation.

Visualization of the learning curves Learning curves, illustrating performance, are plotted as averages from 10 trials. A shaded region around each curve represents the 95% confidence interval across these trials. For enhanced visual clarity, the curves are smoothed using a sliding window averaging over a number evaluations determined by 0.05 of the length of the curves.

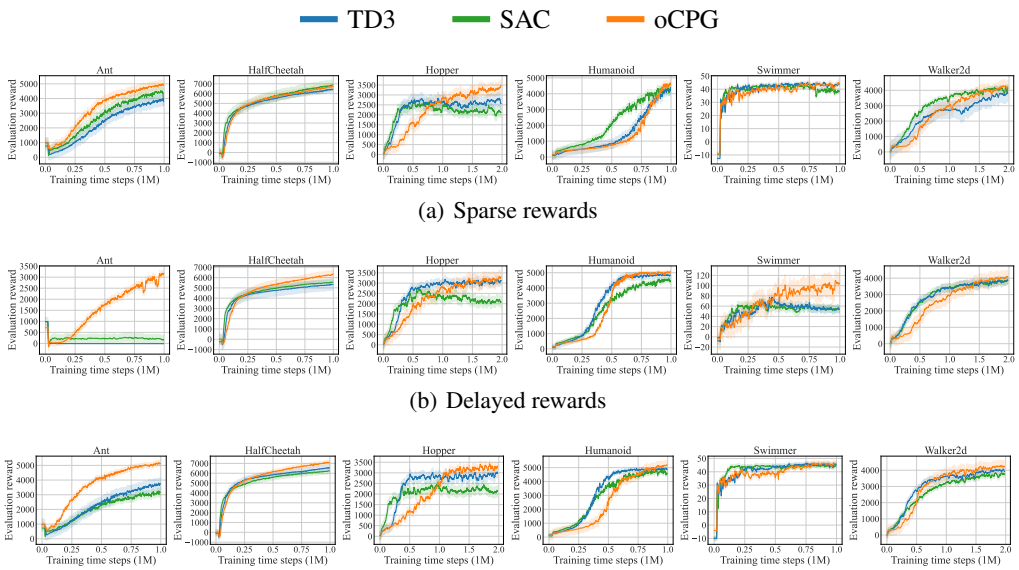
C.3 Computational Resources

Our experiments were conducted on a high-performance computing system featuring an AMD Ryzen Threadripper PRO 3995WX 64-Core processor. This system is equipped with 128 logical CPUs operating at a base frequency of 3.31 GHz, with a maximum frequency of 4.31 GHz, supported by 512 GB of RAM. The neural network training leveraged single NVIDIA RTX A6000 GPU with 48 GB of VRAM. This setup provided the necessary computational power to efficiently handle the intensive tasks associated with neural network training and experimentation.

[2] `https://mujoco.org/`

[3] `https://www.gymlibrary.ml/`

D Evaluation in Imperfect Environment Conditions



(a) Sparse rewards

(b) Delayed rewards

(c) Noisy rewards

Figure 2: Learning curves for the benchmark MuJoCo environments, averaged over 10 random seeds, under imperfect environment conditions: (a) the agent observes the true reward with a probability of 0.5, otherwise zero; (b) rewards are delayed by 10 time steps; and (c) rewards are perturbed by a zero-mean Gaussian noise with a standard deviation equal to 0.1 of the reward range in the replay buffer. The shaded area represents the 95% confidence interval of the mean performance.

To evaluate the robustness of oCPG, we simulate imperfect environmental conditions by introducing uncertainty. This includes considering scenarios with noisy, sparse, and delayed rewards. The delayed reward setting delays the instantaneous rewards received by the agent by 10 time steps. For noisy rewards, we monitor the range of rewards collected by the agent, specifically the minimum and maximum values. The reward-perturbation noise is then sampled from a normal distribution $\mathcal{N}(0, 0.1 \cdot (r_{\text{max}} - r_{\text{min}}))$, where r_{max} and r_{min} are the maximum and minimum of the rewards in the replay buffer. In the case of sparse rewards, with a random probability of 0.5, the agent either observes the actual instantaneous reward or zero.

The results for imperfect environmental conditions are presented in Figure 2. While oCPG generally achieves the highest rewards or matches the baselines in the worst-case scenarios, we observe a slower convergence rate in certain environments, such as Hopper and Walker2d. We attribute this to the nature of the reward-perturbing scenarios included in our tests. These scenarios significantly alter the rewards observed by the agent, disrupting the learning process of the Q-network where the reward is a direct input. Such changes impact the value estimates of the Q-network, which CPG directly uses. Although the baselines rely on the gradient of the value estimate, modifications to the rewards do not appear to significantly affect their final performance. Despite these challenges, oCPG maintains robust performance across most environments.