DIRECTIONAL-CLAMP PPO

Anonymous authors
Paper under double-blind review

ABSTRACT

Proximal Policy Optimization (PPO) is one of the most widely used algorithms in deep reinforcement learning. Its objective encourages the importance ratio between the current and behavior policies to move in the "right direction", i.e., increasing the ratios for positive advantages and decreasing them for negative ones. To prevent overly aggressive updates in the right direction, PPO and its variants employ clipping mechanisms that restrict the magnitude of such updates. However, due to the stochastic nature of the training process, the importance ratios frequently move in the "wrong direction" during PPO optimization. We identify the *strictly wrong direction* updates as a key but largely overlooked factor that limits the performance of PPO. To address this, we propose the Directional-Clamp PPO algorithm (DC-PPO), which further penalizes the actions going to the "strict wrong direction" regions, where the advantage is positive (negative) and importance ratio falls below (above) $1-\beta$ $(1+\beta)$, for a tunable parameter $\beta \in (0,1)$. The penalty is by enforcing a steeper loss slope, i.e., a clamp, in those regions. We demonstrate across a range of environments that DC-PPO consistently outperforms PPO and its variants that modify the objective's behavior in the right direction. Moreover, we show, both theoretically and empirically, that DC-PPO better avoids strict wrong direction updates while keeping the importance ratio closer to 1. Finally, we prove global convergence for tabular DC-PPO with direct policy parameterization.

1 INTRODUCTION

Deep Reinforcement Learning (DRL) algorithms have achieved remarkable success in a variety of complex domains, transforming fields such as robotics (Kober et al., 2013), gaming (Mnih et al., 2015; Silver et al., 2016), and autonomous systems (Gu et al., 2017; Sallab et al., 2017). These breakthroughs demonstrate the ability of models to learn effective strategies through trial-and-error interactions in environments with uncertain dynamics and partial observability. Policy gradient methods are a very popular class of DRL algorithms that focus on directly optimizing a parameterized policy to maximize its expected cumulative reward. The key advantage of policy gradient methods lies in their directness and ability to handle high-dimensional and continuous action spaces. The convergence properties of the policy gradient methods are strongly influenced by the choice of step-size: small values imply a slow convergence rate, while large values may lead to oscillation or even divergence of the policy parameters (Pirotta et al., 2013). The well-known *Trust-Region Policy Optimization* algorithm (TRPO) (Schulman et al., 2015) addresses the step-size sensitivity issue by constraining the Kullback–Leibler (KL) divergence between successive policies with theoretically guaranteed monotonic policy improvement. However, TRPO is computationally expensive, requiring second-order approximations and conjugate gradient methods, which limits its scalability.

To improve efficiency and scalability, *Proximal Policy Optimization* (PPO) (Schulman et al., 2017) introduces a simple clipping mechanism that approximates TRPO's trust-region constraint. Throughout this paper, we refer to the PPO objective function as the PPO-clip objective function. The key idea is to constrain the importance ratio, i.e., the current policy divided by the trajectory sampling old policy (rollout policy), within a small interval $[1-\epsilon, 1+\epsilon]$ to avoid over-optimization. Specifically, the target is to increase (decrease) the ratio of the actions with positive (negative) advantage, which defines the "right direction" to optimize. The PPO objective function is clipped with zero gradient if the ratios are over-optimized too much along the "right direction", i.e., ratios greater than $1+\epsilon$ for positive advantage or smaller than $1-\epsilon$ for negative advantage. Due to its simplicity and efficiency, PPO is an iconic policy gradient algorithm that has been applied in many different fields and shown

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

great stability across domains, including games (Berner et al., 2019), healthcare (Yu et al., 2021), finance (Ye et al., 2020), energy systems (Li et al., 2023), UAV control (Koch et al., 2019), and more recently alignment of large language models (LLM) (Ouyang et al., 2022).

Despite its success, PPO focuses on preventing over-optimization in the right direction region, implicitly assuming that updates can predominantly follow the right direction. However, we observe that a substantial portion of the importance sampling ratios drift in the wrong direction – lowering the probability of advantageous actions and raising that of disadvantageous ones – during the PPO training, as shown in Figure 3. This "wrong-directional" drift arises naturally from the stochastic training process: the trajectories are randomly sampled, the advantage estimates can have high variance, and the policy updates are performed on small mini-batches. Note that the advantage is estimated, but we treat the problem with given advantage estimates as a pure optimization problem and define right/wrong directions accordingly.

To address this issue, we introduce Directional-Clamp PPO (DC-PPO), which augments the standard PPO objective with a targeted penalty that more aggressively suppresses wrong-direction updates. The *directional* component applies this penalty selectively, activating it only when the ratio stays in the strictly wrong-direction region, i.e., ratio less than $(1-\beta)$ for positive advantage or ratio greater than $(1+\beta)$ for negative advantage, where $\beta \in (0,1)$ is a parameter to control this region.[1]

Focusing on the strictly wrong-direction region is crucial. Due to the inherent stochasticity of policy optimization, small deviations into the wrong direction are unavoidable and can be viewed as noise around the trust-region center. Penalizing all wrong-direction updates would therefore be overly restrictive and may hinder effective learning. Instead, we treat the problem as optimizing the objective with given (possibly noisy) advantage estimates, and define right and wrong directions accordingly. Under this view, large deviations into the wrong direction—captured by the strictly wrong-direction region—correspond to systematically harmful updates that move the policy away from improvement. DC-PPO specifically targets these harmful updates while tolerating small fluctuations.

As illustrated in Figure 4, the *clamp* component sharpens the loss landscape in these regions, effectively pulling the importance ratio back toward 1 and aligning policy updates more closely with the trust-region principle. Through this mechanism, DC-PPO substantially reduces harmful deviations, yielding both theoretical stability and empirical improvement over PPO and its strongest variants across diverse MuJoCo (Todorov et al., 2012) and Atari (Bellemare et al., 2013) benchmarks.

Our contributions are summarized as follows:

1. We propose DC-PPO, a simple, intuitive, and effective modification on the objective function of PPO without any overhead to the standard PPO pipeline, focusing on the wrong direction during policy optimization that has been largely overlooked.
2. Theoretically, we show that under some mild conditions, the DC-PPO update keeps the ratio closer to 1, thereby reducing harmful deviations and aligning updates within the trust-region.
3. DC-PPO consistently outperforms PPO and other reported competitive baselines to achieve state-of-the-art performance for a range of MuJoCo and Atari environments. This could motivate future research on the "wrong direction" region of the policy gradient algorithms.

2 BACKGROUND

We consider a discounted MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor $\gamma \in (0,1)$, and initial state distribution $\rho_0 \in \Delta_{\mathcal{S}}$. A stochastic Markov policy $\pi(\cdot|s) \in \Delta_{\mathcal{A}}$ induces trajectories $(s_0, a_0, r_0, s_1, \ldots)$ where $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)$. The goal is to maximize the expected discounted return $J(\pi) := \mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. The state-value and action-value functions under π are defined as

$$V^{\pi}(s) := \mathbb{E}_{\pi,P}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\Big], \quad Q^{\pi}(s,a) := \mathbb{E}_{\pi,P}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\Big],$$

[1] For simplicity, we use a single parameter β instead of separate parameters for positive and negative directions.

and the advantage function is defined by $A^\pi(s,a) := Q^\pi(s,a) - V^\pi(s)$. In practice, policies are drawn from a parameterized family $\{\pi_\theta : \theta \in \Theta\}$, where $\pi_\theta(a|s)$ denotes the probability of action a in state s under the set of parameters θ.

Policy Gradient. The objective $J(\pi_\theta)$ is differentiable w.r.t. the policy parameter θ, and its gradient is obtained by the policy gradient theorem (Sutton et al., 1999) as

$$(1-\gamma)\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\big[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)\big],$$

where $\rho_\pi(s) := (1-\gamma)\sum_{t\geq 0}\gamma^t \mathbb{P}(s_t = s)$ is the discounted state visiting distribution under any policy π. The expectation can be estimated via rollout trajectories, and the advantage function can be estimated by various estimators in the pursuit of less variance and bias. A well-adopted choice for the advantage estimator is Generalized Advantage Estimator (GAE), which estimates $A^\pi(s_t,a_t)$ as $\hat{A}(s_t,a_t) = \sum_{l=0}^{T-t-1}(\gamma\lambda)^l \delta_{t+l}$, where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ is the TD-error and V_ϕ is a critic function approximating the value function $V^\pi(s)$.

Trust-Region Policy Optimization. TRPO (Schulman et al., 2015) was proposed to reuse the rollout data and increase the stability of policy gradient methods. It tackles the policy optimization problem via iteratively solving the following constraint surrogate:

$$\max_\theta \hat{\mathbb{E}}_{s,a}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\hat{A}(s,a)\right], \quad \text{s.t. } \hat{\mathbb{E}}_{s,a}\left[D_{\text{KL}}\big(\pi_{\theta_{\text{old}}}(\cdot|s)\|\pi_\theta(\cdot|s)\big)\right] \leq \delta,$$

where D_{KL} is the KL divergence, $\hat{\mathbb{E}}_{s,a}[\cdot]$ denotes the empirical expectation over a batch of sampled state–action pairs, and π_{old} is a rollout policy that can potentially be different from π_θ. This constrained optimization surrogate theoretically guarantees monotonic policy improvement, provided a sufficiently large number of rollout samples and a small trust-region parameter δ are available. However, practically solving this optimization is computationally expensive, as it requires a second-order KL approximation with the Fisher information matrix and conjugate gradient descent for the quadratic program.

Proximal Policy Gradient and its Variants. To overcome this challenge, Schulman et al. (2017) proposed PPO as an alternative that relies only on first-order optimization. There are different variations of the PPO formulation, but we take the most adopted one in practice, the PPO(-clip), in which a simple clipped surrogate function replaces the constrained optimization of TRPO to maximize

$$\mathcal{J}_{\text{PPO}}(\theta) = \hat{\mathbb{E}}_{s,a}\Big[\min\big(w_\theta(s,a)\hat{A}(s,a), \text{clip}(w_\theta(s,a), 1-\epsilon, 1+\epsilon)\hat{A}(s,a)\big)\Big],$$

where $w_\theta(s,a) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the likelihood (or importance) ratio and $\epsilon > 0$ is the clipping hyperparameter.

Many PPO variants introduce regularization terms other than KL or adaptively update the trust-region. The PPO-based variants mostly focus on the clipping part of the surrogate loss, since it is believed to be the essence of the PPO's success. Here, we introduce two PPO variants that mathematically contradict each other due to different perspectives on the clip part of PPO.

Leaky PPO (Han et al., 2024) believes the clipping may slow down the training due to the vanishing gradient information outside the clipping band, and thus, proposes the following surrogate function:

$$\mathcal{J}_{\text{Leaky-PPO}}(\theta) = \hat{\mathbb{E}}_{s,a}\Big[\min\Big(w_\theta(s,a)\hat{A}(s,a), f_{\text{Leaky}}\big(w_\theta(s,a),\epsilon,\alpha\big)\hat{A}(s,a)\Big)\Big], \tag{1}$$

where $\alpha \in [0,1)$ is a hyperparameter controlling the slope of the clip that leaks the gradient information and

$$f_{\text{Leaky}}\big(w_\theta(s,a),\epsilon,\alpha\big) = \begin{cases} \alpha w_\theta(s,a) + (1-\alpha)(1-\epsilon) & \text{if } w_\theta(s,a) \leq 1-\epsilon, \\ \alpha w_\theta(s,a) + (1-\alpha)(1+\epsilon) & \text{if } w_\theta(s,a) \geq 1+\epsilon, \\ w_\theta(s,a) & \text{otherwise.} \end{cases}$$

On the other hand, Wang et al. (2020) aim at further increasing PPO's stability via enforcing stricter trust-region constraint using the following surrogate function:

$$\mathcal{J}_{\text{PPO-RB}}(\theta) = \hat{\mathbb{E}}_{s,a}\Big[\min\Big(w_\theta(s,a)\hat{A}(s,a), f_{\text{RB}}\big(w_\theta(s,a),\epsilon,\alpha\big)\hat{A}(s,a)\Big)\Big], \tag{2}$$

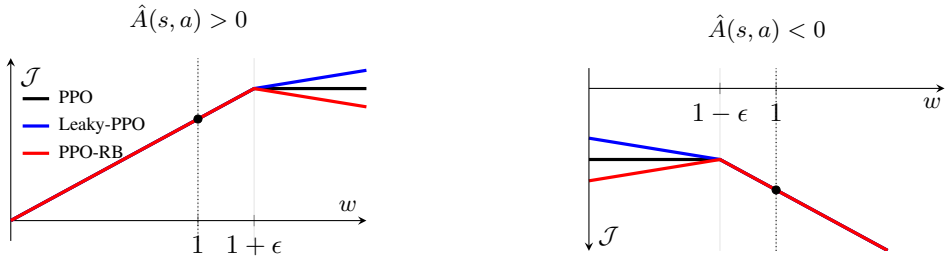


Figure 1: The surrogate objective of PPO, Leaky PPO, and PPO-RB as a function of the likelihood ratio $w_\theta(s,a)$ for positive advantage (left) and negative advantage (right). The black dot denotes $w_\theta(s,a) = 1$.

with $\alpha > 0$ and

$$f_{\text{RB}}\big(w_\theta(s,a), \epsilon, \alpha\big) = \begin{cases} -\alpha w_\theta(s,a) + (1+\alpha)(1-\epsilon), & \text{if } w_\theta(s,a) \le 1-\epsilon, \\ -\alpha w_\theta(s,a) + (1+\alpha)(1+\epsilon), & \text{if } w_\theta(s,a) \ge 1+\epsilon, \\ w_\theta(s,a), & \text{otherwise.} \end{cases}$$

Figure 1 shows the difference between the surrogate objectives of PPO and these two algorithms. From Figure 1, we notice that both Leaky-PPO and PPO-RB attribute the success and potential improvements of PPO to the clip function, which even leads to their contradicting algorithmic designs. We do not judge which of these variants is better as it depends on the situation, such as the environment and hyperparameters. While these approaches specifically modify the objective's behavior by adjusting the clipping mechanism in the "right direction" regions, we highlight that there also exists a "wrong direction" that can significantly improve PPO, but has been largely overlooked.

3 Main Results

In this section, we quantify and address a directional failure mode of PPO. In Section 3.1, we formally define "right" and "wrong" direction updates, illustrate them on the PPO surrogate (Figure 2), and empirically show substantial mass in the wrong direction (Figure 3). Motivated by these observations, in Section 3.2, we introduce the notion of a "strict wrong direction" and propose Directional-Clamp PPO (DC-PPO), which augments the PPO surrogate with a directional penalty that only activates in strict wrong direction regions (Figure 4). We further prove that when initialized in a strict wrong direction, DC-PPO moves ratios closer to 1 than PPO (Lemma 1), and corroborate this with experiments (Tables 1 and 2, and Figure 5).

3.1 Importance Ratios in the Wrong Direction

The optimization of the surrogate in Equation 1 begins with $\theta = \theta_{\text{old}}$, where the importance ratio satisfies $w_\theta(s,a) = \pi_\theta(a|s)/\pi_{\theta_{\text{old}}}(a|s) = 1$. During optimization, we say that a sample (s,a) lies in the *wrong direction region* if $(w_\theta(s,a) - 1)\hat{A}(s,a) < 0$, meaning that the parameter update induced by θ changes the policy probability in a direction opposite to the advantage signal, i.e., it decreases the likelihood of advantageous actions or increases that of disadvantageous ones. The *right direction region* can be defined conversely as $(w_\theta(s,a) - 1)\hat{A}(s,a) > 0$. Figure 2 shows the right and wrong directions and their relations to the PPO objective.

Although PPO's objective function in Equation 1 is designed to discourage updates in the wrong directions, our experimental results illustrated in Figure 3 reveal that this safeguard is not always effective in practice. In our experiments, we observed that approximately 34.95% of the ratio samples were in the wrong direction on average when the advantage was negative. Moreover, when the advantage was positive, 39.88% of the ratio samples were in the wrong direction on average, with the proportion reaching as high as 49% in a single environment. These ratios suggest that PPO frequently produces updates that move the policy against the advantage signal, which can undermine learning stability. Specifically, for each collected state-action pair (s,a), we compute the corresponding

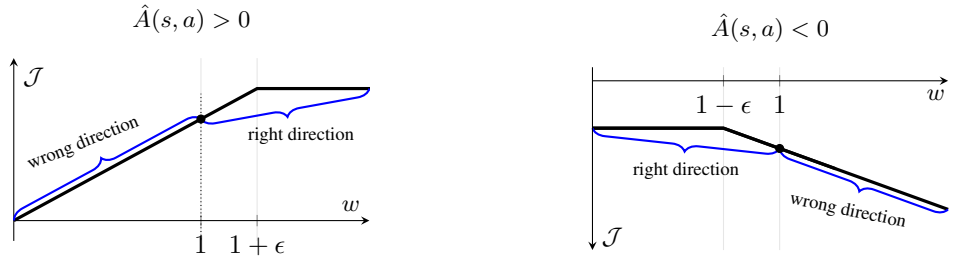
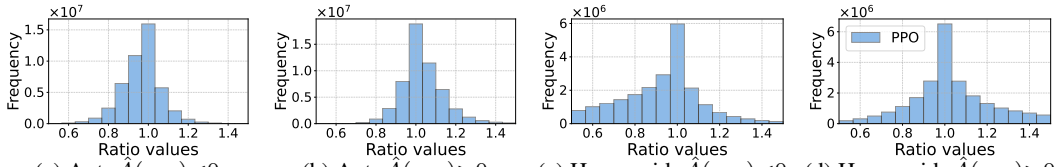


Figure 2: The right and wrong directions on the PPO surrogate function. The black dot at the importance ratio 1 symbolizes the starting point of the optimization.



(a) Ant, $\hat{A}(s,a)<0$ (b) Ant, $\hat{A}(s,a)>0$ (c) Humanoid, $\hat{A}(s,a)<0$ (d) Humanoid, $\hat{A}(s,a)>0$

Figure 3: Histograms of importance ratios during PPO optimization for two MuJoCo environments.

importance ratio $w_\theta(s,a)$ and advantage estimate $\hat{A}(s,a)$. The reported percentages correspond to the empirical proportion of such samples across all updates. The histogram in Figure 3 visualizes this distribution.

3.2 Directional-Clamp PPO Algorithm

Motivated by the observations reported in Section 3.1, we now propose our algorithm, Directional-Clamp PPO (DC-PPO), which is mainly based on a modification to PPO clipping. Before describing DC-PPO, it is important to note that small deviations from $w_\theta(s,a) = 1$ are natural during stochastic optimization and may not harm performance. Thus, to separate benign deviations from genuinely harmful ones, we introduce the notion of a *strict wrong direction*. More precisely, we consider an update in a strictly wrong direction if the updated parameter θ_{new} assigns a probability "significantly" lower (higher) than the old policy to an action with positive (negative) advantage. We quantify the degree of strictness using a hyperparameter $0 \leq \beta \leq 1$ and formally define strict wrong direction as

$$\begin{cases} w_\theta(s,a) > 1+\beta & \text{if } \hat{A}(s,a) < 0, \\ w_\theta(s,a) < 1-\beta & \text{if } \hat{A}(s,a) > 0. \end{cases}$$

DC-PPO steers the policy parameter θ toward the correct direction during optimization by introducing a hyperparameter that controls the strength of an additional penalty term in the surrogate objective. Importantly, this penalty is applied only when updates fall into the "strict wrong direction" region, thereby focusing the correction on harmful updates, while tolerating benign fluctuations near $w_\theta(s,a) = 1$. In doing so, DC-PPO also mitigates the overshooting problem, where probability ratios drift too far from 1 despite clipping.

DC-PPO Surrogate To implement the idea described above, we introduce a hyperparameter $\alpha > 1$ that determines the slope of the penalty applied in the strictly wrong direction regions. This results in the surrogate function $\mathcal{J}_{\text{DC-PPO}}(\theta)$, which we define by

$$\hat{\mathbb{E}}_{s,a}\Big[\min\Big(w_\theta(s,a)\hat{A}(s,a), \text{clip}\left(w_\theta(s,a), 1-\epsilon, 1+\epsilon\right)\hat{A}(s,a), f_{\text{DC}}\big(w_\theta(s,a), \epsilon, \alpha, \beta\big)\hat{A}(s,a)\Big)\Big]. \tag{3}$$

where

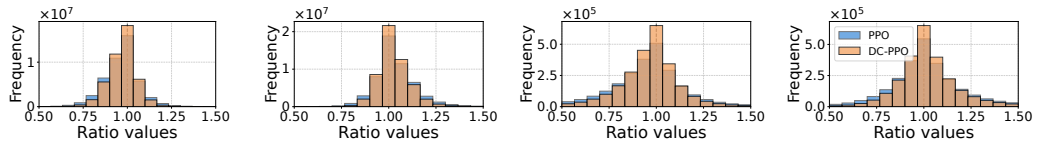
$$f_{\text{DC}}\big(w_\theta(s,a), \epsilon, \alpha, \beta\big) = \begin{cases} \alpha w_\theta(s,a) - (\alpha-1)(1-\beta) & \text{if } \hat{A}(s,a) > 0, \\ \alpha w_\theta(s,a) - (\alpha-1)(1+\beta) & \text{if } \hat{A}(s,a) < 0. \end{cases}$$



Figure 4: The surrogate objectives of PPO and DC-PPO, where the red curve is $\mathcal{J}_{\text{DC-PPO}}$ and the black curve is $\mathcal{J}_{\text{PPO}}$.

Algorithm 1 A pseudo-code for our Directional-Clamp PPO (DC-PPO) Algorithm.

Initialize $\alpha > 1$, $\beta \in [0, 1]$, and ϵ.
for iteration $= 1, 2, \ldots$ **do**
  for actor $= 1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T time-steps.
    Compute generalized advantage estimators $\hat{A}(s_1, a_1), \ldots, \hat{A}(s_T, a_T)$.
  end for
  for epoch $= 1, 2, \ldots, K$ **do**
    Optimize policy π_θ based on $\mathcal{J}_{\text{DC-PPO}}$ in equation 3 using the coefficients α and β.
  end for
  $\theta_{\text{old}} \leftarrow \theta$
end for



(a) Ant-v4, $\hat{A}(s,a)<0$ (b) Ant-v4, $\hat{A}(s,a)>0$ (c) Hopper-v4, $\hat{A}(s,a)<0$ (d) Hopper-v4, $\hat{A}(s,a)>0$

Figure 5: Histograms of importance ratios $w_\theta(s,a)$ during training of DC-PPO and PPO. Each histogram aggregates tens of millions of samples collected across optimization steps.

Figure 4 shows the difference between the surrogate functions of PPO and DC-PPO for both cases, where the advantage estimate is positive and negative. Given the surrogate function in Equation 3, we can write a pseudo-code for DC-PPO as shown in Algorithm 1.

With the optimal values of hyperparameters α and β (see Appendix A), DC-PPO substantially reduces the proportion of ratio samples in the strict wrong direction region, for both positive and negative advantages, across all the MuJoCo environments in our experiments (see Table 1 and Figure 5). The results in Table 1 indicate that the average proportion of ratio samples in the strict wrong direction regions across seven MuJoCo environments for DC-PPO is roughly two-thirds of that for PPO. Figure 5 also shows that DC-PPO substantially reduces the number of ratio samples falling into the strict wrong direction region compared to PPO. The fact that most ratios remain concentrated around $w_\theta(s,a) = 1$ indicates that updates stay closer to the intended trust-region under DC-PPO than PPO.

Table 1: Proportion of strict wrong direction regions evaluated across MuJoCo environments.

Environment	$\hat{A}(s,a) < 0$		$\hat{A}(s,a) > 0$	
	DC-PPO	PPO	DC-PPO	PPO
HalfCheetah-v4	**0.0647**	0.1014	**0.0745**	0.1258
Hopper-v4	**0.0568**	0.0883	**0.0600**	0.1062
Swimmer-v4	**0.0153**	0.0388	**0.0110**	0.0231
Walker2d-v4	**0.0293**	0.0461	**0.0284**	0.0458
Ant-v4	**0.0428**	0.0692	**0.0445**	0.0753
Humanoid-v4	**0.0342**	0.0406	**0.0371**	0.0471
Reacher-v4	**0.0428**	0.0671	**0.0278**	0.0518
Overall	**0.0405**	0.0617	**0.0425**	0.0689

We can also formally justify the above numerical results using the following theoretical finding. Given a horizon $T \in \mathbb{N}$ and a parameterized policy π_θ continuous w.r.t. θ, we define the set of all

iterations $t \in [T]$ in the mini-batch for which the ratio $w_{\theta_0}(s_t, a_t)$ is in the strict wrong direction region as $\Omega_T = \{t \in [T] : |w_{\theta_0}(s_t, a_t) - 1| > \beta \wedge (w_{\theta_0}(s_t, a_t) - 1)\hat{A}(s_t, a_t) < 0\}$. For simplicity, we use the following two notations: $\theta_1^{\text{PPO}} = \theta_0 + \gamma \nabla \mathcal{J}_{\text{PPO}}(\theta_0)$ and $\theta_1^{\text{DC-PPO}} = \theta_0 + \gamma \nabla \mathcal{J}_{\text{DC-PPO}}(\theta_0)$, for any given policy parameter θ_0.

Lemma 1. *For any $t \in \Omega_T$, if the initial policy parameter θ_0 produces an importance ratio $w_{\theta_0}(s_t, a_t)$ satisfying $\sum_{t' \in \Omega_T} \langle \nabla w_{\theta_0}(s_t, a_t), \nabla w_{\theta_0}(s_{t'}, a_{t'}) \rangle \hat{A}(s_t, a_t)\hat{A}(s_{t'}, a_{t'}) > 0$, then there exists a $\bar{\gamma} > 0$, such that for any $\gamma \in (0, \bar{\gamma})$, we have*

$$|w_{\theta_1^{\text{DC-PPO}}}(s_t, a_t) - 1|^2 < |w_{\theta_1^{\text{PPO}}}(s_t, a_t) - 1|^2.$$

Lemma 1 shows that the clamp in DC-PPO updates the ratio in a strict wrong direction region closer to 1 than PPO. This is corroborated empirically in Table 2, which shows that DC-PPO achieves a lower MSE of ratios from 1 compared to PPO for all seven MuJoCo environments (except a single case when $\hat{A}(s, a) > 0$). This suggests that DC-PPO strongly constrains updates within the trust-region, leading to more stable policy changes, whereas PPO exhibits larger variability in its ratio distributions.

Table 2: Reduction in MSE of the critic under DC-PPO vs PPO.

Environment	$\hat{A}(s,a) < 0$		$\hat{A}(s,a) > 0$	
	DC-PPO	PPO	DC-PPO	PPO
HalfCheetah-v4	**0.0134**	0.0212	**0.0155**	0.0250
Hopper-v4	**0.0369**	0.0716	**0.0757**	0.1510
Swimmer-v4	**0.0072**	0.0133	**0.0062**	0.0099
Ant-v4	**0.0070**	0.0110	**0.0080**	0.0133
Reacher-v4	**0.0365**	0.0686	**0.0308**	0.0534
Humanoid-v4	**0.0583**	0.0720	1.5816	**1.3608**
Walker2d-v4	**0.0058**	0.0080	**0.0059**	0.0083

4 DC-PPO ATTAINS GLOBAL OPTIMALITY

In this section, we establish the global convergence of DC-PPO. Following an earlier work by Huang et al. (2024), we study the global convergence under a direct parameterization, i.e., policies are parameterized by $\pi(a|s) = \theta_{s,a}$, where $\theta_s \in \Delta_{\mathcal{A}}$, and $\theta \in \Delta_{\mathcal{A}}^{|\mathcal{S}|}$. At each iteration t, the algorithm collects a batch of on-policy data by executing the current policy $\pi^{(t)}$ in the environment. We denote by $\mathcal{D}^{(t)}$ the resulting set of trajectories collected at iteration t. DC-PPO updates the policy by minimizing the estimated loss in Equation 3 using EMDA as described in Algorithm 3 in Appendix D. To carry out our analysis, we make the following two assumptions.

Assumption 1. *Each state-action pair (s, a) appears infinitely many times in $\{\mathcal{D}^{(t)}\}$ almost surely.*

Assumption 2. *In each iteration t, all the states in $\mathcal{D}^{(t)}$ are distinct.*

Assumption 1 is a standard persistent exploration condition. It guarantees that every state–action pair is visited infinitely often, ensuring that no part of the state–action space is permanently neglected. This assumption is necessary to avoid degenerate cases where certain pairs are never observed, in which case consistent estimation or convergence guarantees are impossible. Assumption 2 is a technical simplification ensuring that there is no duplicate state within a batch, thus, avoiding additional dependencies between samples at each iteration t. The assumption is commonly satisfied when data are collected from long trajectories or large replay buffers, and it mainly serves to simplify the analysis rather than to restrict practical applicability. The following theorem, whose proof is reported in Appendix D, shows the global convergence of DC-PPO.

Theorem 1 (DC-PPO global convergence)**.** *Under Assumptions 1 and 2, we have for DC-PPO that $V^{\pi^{(t)}}(s) \to V^{\pi^\star}(s), \forall s \in \mathcal{S}$, as $t \to \infty$, almost surely.*

5 RELATED WORK

Implementing PPO requires careful tuning and its clipped objective has been shown outperforming vanilla policy gradient methods while achieving performance comparable to TRPO at a lower computational cost (Engstrom et al., 2020; Andrychowicz et al., 2021; Huang et al., 2022). A large body of work has sought to further improve PPO, primarily by modifying the objective's behavior in the right direction. Several methods revisit the clipping mechanism. Leaky PPO (Han et al., 2024) addresses gradient saturation outside the clipping region by allowing controlled gradient flow beyond the clipping boundary, leading to strong empirical performance. Similarly, PPO-RB

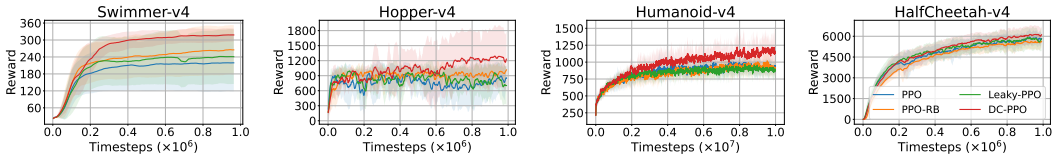


Figure 6: Performance comparisons of PPO, PPO-Leaky, PPO-RB, and DC-PPO (ours) across four selected MuJoCo environments.

(Wang et al., 2020) introduces a rollback-based correction to better enforce trust-region constraints and stabilize updates. SPO (Xie et al., 2025) strengthens the trust-region constraint by replacing the KL-divergence with a quadratic penalty. From our perspective, this simultaneously addresses both right and wrong direction updates, whereas our method explicitly targets wrong direction samples.

From a theoretical perspective, prior work has analyzed the convergence properties and limitations of PPO (Jin et al., 2024; Huang et al., 2024). Other approaches focus on strengthening the trust-region constraint, for example by replacing KL-divergence with alternative penalties or adaptive mechanisms (Meng et al., 2024; Garg et al., 2021; Wang et al., 2019). In contrast to these methods, which primarily control update magnitude or improve optimization in the right direction, our approach explicitly targets wrong-direction updates, a largely overlooked source of performance degradation.

6 Experiments

We compare DC-PPO against three baselines: PPO and its variants: Leaky PPO (Han et al., 2024) and PPO-RB (Wang et al., 2020). These two variants modify the objective function only in the right direction, where Leaky PPO (Han et al., 2024) claims to achieve the state-of-the-art. To evaluate the effectiveness of DC-PPO, we conduct extensive experiments on MuJoCo and Atari environments. Experimental details including the fixed values of α, β, and the environment-specific PPO configurations adapted from RL Baselines3 Zoo (Raffin, 2020) are provided in Appendix A.

Figure 6 shows training curves on four representative MuJoCo environments. Each curve reports the average episodic return over training, where evaluation is performed every 1,000 episodes using 10 independent rollouts. Results are averaged over 10 random seeds, with shaded regions indicating ± standard deviation, and smoothed using a moving-average window of size 10. DC-PPO consistently exhibits faster learning and higher final performance in Swimmer-v4, Hopper-v4, and Humanoid-v4. In HalfCheetah-v4, it follows a similar training trajectory to PPO but still achieves a higher final return. We highlight these four as representative as they form two classes – one where DC-PPO had significant gains and one where the performances are comparable. More training curves for MuJoCo and Atari environments can be found in Appendix B.1.

Table 3 provides quantitative comparisons of performance across seven MuJoCo environments. We report both the average return over the last 10 evaluations of training, reflecting the final converged performance, and the average of the top 10 evaluations, capturing the best performance achieved during training. For both metrics, DC-PPO consistently matches or outperforms PPO, Leaky PPO, and PPO-RB in most environments. Gains are significant in Hopper-v4, Swimmer-v4, and Humanoid-v4, with improvements of up to nearly 40% over PPO. In environments where it does not attain the best score (e.g., Ant-v4 and Reacher-v4), the difference remains small.

The improvements become more pronounced in Atari environments, as shown in Figure 8 in Appendix B.1. We observe consistent gains across most tasks, with particularly strong im-

Table 3: Performance comparisons on MuJoCo: final performance (last-10 average return) and peak performance (top-10 average return).

Last-10 average return				
Environment	PPO	PPO-RB	Leaky PPO	DC-PPO
HalfCheetah-v4	5833.58	5614.77	5802.79	**6110.72**
Hopper-v4	852.98	1015.99	705.10	**1230.66**
Swimmer-v4	219.23	265.16	241.02	**317.90**
Ant-v4	5713.90	**5888.53**	5735.24	5521.60
Reacher-v4	-4.76	-4.68	-4.73	**-4.64**
Humanoid-v4	932.01	914.02	874.24	**1154.80**
Walker2d-v4	3432.08	3362.96	3809.17	**3831.43**
Top-10 average return				
Environment	PPO	PPO-RB	Leaky PPO	DC-PPO
HalfCheetah-v4	6085.68	5768.28	6019.54	**6232.81**
Hopper-v4	998.72	1020.64	1009.99	**1299.70**
Swimmer-v4	220.62	269.12	243.86	**319.29**
Ant-v4	6439.54	**6563.56**	6469.42	6488.27
Reacher-v4	**-3.69**	-4.00	-3.89	-3.85
Humanoid-v4	1079.52	1091.90	1091.53	**1349.32**
Walker2d-v4	3563.23	3724.54	4022.19	**4209.62**

provements in Hero-v5, MsPacman-v5, and Seaquest-v5. Additionally, DC-PPO yields a statistically significant improvement over PPO with $p < 0.05$ under paired statistical test in Appendix B.2. Overall, these results demonstrate that clamping in the strictly wrong-direction updates leads to improved performance and stability across both continuous and discrete control domains, highlighting the general effectiveness and stability of DC-PPO.

Impact of clamping in DC-PPO As shown in Table 1, DC-PPO consistently reduces such updates across all environments. Notably, this reduction does not lead to performance degradation: DC-PPO matches or improves upon PPO in every task considered. At the same time, the magnitude of improvement varies across environments. In tasks such as Hopper-v4, Swimmer-v4, and Humanoid-v4, suppressing strictly wrong-direction updates leads to substantial performance gains. In contrast, in environments like Ant-v4 and Reacher-v4, the performance remains comparable to PPO despite a similar reduction in strictly wrong-direction updates. These results suggest that focusing on strictly wrong-direction updates is a safe and generally beneficial modification: it does not harm performance and can yield significant improvements when such updates interfere with learning.

Role of β (separating noise from harmful updates) We further investigate the effect of the threshold parameter β, which defines the strictly wrong-direction region. Interestingly, tuning β does not necessarily decrease the overall proportion of wrong-direction samples compared to PPO; in fact, with the optimal β, this proportion can slightly increase (see Table 6 in Appendix B.5). Meanwhile, aggressively reducing β decreases the number of wrong-direction samples but consistently degrades performance (Figure 11 in Appendix B.4). These results highlight a key distinction: not all wrong-direction updates are harmful. Small deviations can be interpreted as stochastic noise inherent to policy optimization, arising from sampling variability and noisy advantage estimates. Penalizing all such deviations is overly restrictive and harms learning. Instead, DC-PPO focuses on strictly wrong-direction updates, which correspond to large and systematically detrimental deviations. As illustrated in Figure 11, a small value of β ($\beta = 0.01$) may result in slow convergence, as observed in Swimmer-v4. However, in Walker2d-v4, the same β value yields the best final performance. This suggests that there is no universal optimal value of β. Overall, intermediate values ($\beta \in [0.1, 0.2]$) provide the best trade-off between stability and learning speed (see Figure 11).

Effect of penalty strength α Smaller values of α only impose weak penalties and yield inferior performance, while excessively large ones can destabilize learning (see Figure 10 in Appendix B.3). Therefore, striking the right balance for α is crucial for the stability and effectiveness of DC-PPO.

Integration with right-direction modifications Consider variants that combine DC-PPO with right-direction modifications, including Leaky PPO and PPO-RB (denoted as DC-Leaky and DC-RB). As shown in Tables 7 and 8 in Appendix B.6, clamped variants improve any right-direction modifications. This highlights the effectiveness of explicitly targeting strictly wrong-direction updates: it provides a reliable improvement mechanism that does not rely on further tuning and isolates a key source of performance degradation that is not addressed by existing methods.

7 Conclusion

In this paper, we identified a fundamental limitation of PPO: *although its objective is designed to steer policy updates in the direction of the advantage, a significant portion of action probabilities still fail to do so in practice.* The proposed DC-PPO algorithm aims at explicitly addressing this issue by penalizing "strictly wrong direction" updates through a modified surrogate objective that introduces directional penalties only when updates deviate significantly from the intended direction. We empirically showed that this penalty can substantially reduce the frequency of harmful updates across environments, confirming that DC-PPO effectively suppresses wrong direction policy changes while preserving beneficial exploration. The observed reduction in the strictly wrong direction updates generally translates to improved empirical performance. We also showed that DC-PPO achieves substantial improvements across MuJoCo and Atari environments. Our theoretical analysis shows that DC-PPO produces ratio updates closer to 1 than standard PPO when initialized in strict wrong direction regions, thereby maintaining importance ratios within the trust-region more effectively. This theoretical insight provides a justification of the proposed method from the angle of trust-region.

The effectiveness of DC-PPO highlights an important research direction of focusing on the "wrong direction" regions, which have been largely overlooked in the literature. Future work could explore different penalty shapes to effectively penalize updates in the "wrong direction" and investigate how to adaptively adjust these penalties based on training dynamics.

AI USE STATEMENT

In this work, we used generative AI tools for no tasks requiring mandatory disclosure. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refine hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, interpreting results and generating synthetic data sets, cleaning and reformatting dataset, assisting with translation, supporting qualitative and thematic data analysis, are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code. We have reviewed all AI-assisted work. Any LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the complete hyperparameter configurations used in each one of the environments in section A, where we also provide a textual explanation of what each parameter represents. All experiments are implemented using Stable-Baselines3 (Raffin et al., 2021), with environment-specific PPO configurations adapted from RL Baselines3 Zoo (Raffin, 2020).

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal JĂŁzefowicz, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

Saurabh Garg, Joshua Zhanson, Emilio Parisotto, Adarsh Prasad, Zico Kolter, Zachary Lipton, Sivaraman Balakrishnan, Ruslan Salakhutdinov, and Pradeep Ravikumar. On proximal policy optimization's heavy-tailed gradients. In *International Conference on Machine Learning*, pp. 3610–3619. PMLR, 2021.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Xinchen Han, Hossam Afifi, Hassine Moungla, and Michel Marot. Leaky ppo: A simple and efficient rl algorithm for autonomous vehicles. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2024.

Nai-Chieh Huang, Ping-Chun Hsieh, Kuo-Hao Ho, and I-Chen Wu. Ppo-clip attains global optimality: Towards deeper understandings of clipping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12600–12607, 2024.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022.

Ruinan Jin, Shuai Li, and Baoxiang Wang. On stationary point convergence of PPO-clip. In *The Twelfth International Conference on Learning Representations*, 2024.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pp. 267–274, 2002.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

William Koch, Renato Mancuso, Richard West, and Azer Bestavros. Reinforcement learning for uav attitude control. *ACM Transactions on Cyber-Physical Systems*, 3(2):1–21, 2019.

Yuanzheng Li, Chaofan Yu, Mohammad Shahidehpour, Tao Yang, Zhigang Zeng, and Tianyou Chai. Deep reinforcement learning for smart grid operations: Algorithms, applications, and prospects. *Proceedings of the IEEE*, 111(9):1055–1096, 2023.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Adaptive step-size for policy gradient methods. *Advances in Neural Information Processing Systems*, 26, 2013.

Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research*, 22(268):1–8, 2021.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

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*, pp. 5026–5033. IEEE, 2012.

Yuhui Wang, Hao He, Xiaoyang Tan, and Yaozhong Gan. Trust region-guided proximal policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In *Uncertainty in artificial intelligence*, pp. 113–122. PMLR, 2020.

Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Forty-second International Conference on Machine Learning*, 2025.

Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 1112–1119, 2020.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

A DC-PPO HYPERPARAMETER CONFIGURATIONS

All of our experiments were implemented using the open-source Stable-Baselines3 library (Raffin et al., 2021), with hyperparameter configurations adapted from the RL Baselines3 Zoo repository (Raffin, 2020). We report them here.

The authors of Leaky-PPO and PPO-RB provide the optimal coefficients in the appendices of their respective works (Han et al., 2024; Wang et al., 2020).

Before reporting the results, we clarify several experimental details and definitions used in Tables 3. The flag Normalize = True indicates that both observations and rewards are normalized during training. The parameter number of workers denotes the number of parallel environments used during training. The MlpPolicy architecture refers to a multilayer perceptron policy and value network implemented in Stable-Baselines3, consisting of two hidden layers with 64 units each and `tanh` activations. Stable-Baselines3 also provides a `CnnPolicy`, which employs convolutional neural networks to extract features before passing them to the policy and value heads. The variable total steps specifies the total number of environment interactions during training, while horizon defines the number of rollout steps collected per environment before each policy update. Each policy update uses mini-batches of the specified batch size and is optimized for the number of gradient update epochs per update. Other hyperparameters follow the standard PPO setup. The final objective of PPO (Schulman et al., 2017) combines the clipped policy surrogate with a value function and an entropy bonus. Formally, the overall objective is

$$\mathcal{J}(\theta, \phi) = \hat{\mathbb{E}}_{s,a} \left[\mathcal{J}_{\text{PPO}}(\theta) - c_{\text{vf}} \mathcal{J}_{\text{vf}}(\phi) + c_{\text{ent}} \mathcal{J}_{\text{ent}}(\theta) \right], \tag{4}$$

where $\mathcal{J}_{\text{vf}}(\phi) = (V_\phi(s_t) - V_t^{\text{targ}})^2$ is a squared-error loss and $\mathcal{J}_{\text{ent}}(\theta)$ denotes the entropy bonus. The coefficients c_{vf} and c_{ent} balance the critic and entropy contributions. During optimization, gradients of $\mathcal{J}$ are clipped to a maximum norm to ensure stable learning dynamics. Hyperparameters γ and GAE parameter λ control temporal credit assignment via Generalized Advantage Estimation, ϵ specifies the clipping range for the ratio constraint 1.

A.1 OPTIMAL VALUES USED ACROSS ALL ENVIRONMENTS

Parameter	**Value**
Coefficient for DC-PPO (α)	3
Coefficient for DC-PPO (β)	ϵ (Optimal clip range for PPO)
Coefficient for Leaky-PPO (α)	0.01
Coefficient for PPO-RB (α)	0.02 (Humanoid), 0.3 (Other)

A.2 HYPERPARAMETERS FOR ATARI BENCHMARKS

Table 4: Detailed hyperparameters used for Atari benchmarks adopted from Xie et al. (2025).

Hyperparameters	**Value**
Number of workers	8
Policy	CnnPolicy
Horizon	128
Batch size	1024
Total steps	1×10^7
Update epochs	4
Batch size	256
Discount factor Discount factor γ	0.99
GAE parameter λ	0.95
Clip range ϵ	0.2
Entropy loss coefficient c_2	0.01
Value loss coefficient c_1	0.5

A.3 ANT-V4

Parameter	Value
Normalize	True
Number of workers	1
Policy	MlpPolicy
Total steps	1×10^7
Batch size	32
Horizon	512
Discount factor γ	0.98
Learning rate	1.90609×10^{-5}
Entropy loss coefficient c_2	4.9646×10^{-7}
Clip range	0.1
Update epochs	10
GAE parameter λ	0.8
Max grad norm	0.6
Value loss coefficient c_1	0.677239

A.4 HALFCHEETAH-V4

Parameter	Value
Normalize	True
Number of workers	1
Policy	MlpPolicy
Total steps	1×10^6
Batch size	64
Horizon	512
Discount factor γ	0.98
Learning rate	2.0633×10^{-5}
Entropy loss coefficient c_2	4.01762×10^{-4}
Clip range	0.1
Update epochs	20
GAE parameter λ	0.92
Max grad norm	0.8
Value loss coefficient c_1	0.58096
Policy kwargs	log_std_init=-2, ortho_init=False, activation=ReLU, net_arch=pi=[256,256], vf=[256,256]

A.5 HOPPER-V4

Parameter	Value
Normalize	True
Number of workers	1
Policy	MlpPolicy
Total steps	1×10^6
Batch size	32
Horizon	512
Discount factor γ	0.999
Learning rate	9.80828×10^{-5}
Entropy loss coefficient c_2	2.29519×10^{-3}
Clip range	0.2
Update epochs	5
GAE parameter λ	0.99
Max grad norm	0.7
Value loss coefficient c_1	0.835671
Policy kwargs	log_std_init=-2, ortho_init=False, activation=ReLU, net_arch=pi=[256,256], vf=[256,256]

A.6 HUMANOID-V4

Parameter	Value
Normalize	True
Number of workers	1
Policy	MlpPolicy
Total steps	1×10^7
Batch size	256
Horizon	512
Discount factor γ	0.95
Learning rate	3.56987×10^{-5}
Entropy loss coefficient c_2	2.38306×10^{-3}
Clip range	0.3
Update epochs	5
GAE parameter λ	0.9
Max grad norm	2.0
Value loss coefficient c_1	0.431892
Policy kwargs	log_std_init=-2, ortho_init=False, activation=ReLU, net_arch=pi=[256,256], vf=[256,256]

A.7 REACHER-V4[2]

Parameter	Value
Normalize	True
Number of workers	1
Policy	MlpPolicy
Total steps	1×10^6
Batch size	32
Horizon	512
Discount factor γ	0.9
Learning rate	1.04019×10^{-4}
Entropy loss coefficient c_2	7.52585×10^{-8}
Clip range	0.3
Update epochs	5
GAE parameter λ	1.0
Max grad norm	0.9
Value loss coefficient c_1	0.950368

A.8 WALKER2D-V4

Parameter	Value
Normalize	True
Number of workers	1
Policy	MlpPolicy
Total steps	1×10^6
Batch size	32
Horizon	512
Discount factor γ	0.99
Learning rate	5.05041×10^{-5}
Entropy loss coefficient c_2	5.85045×10^{-4}
Clip range	0.1
Update epochs	20
GAE parameter λ	0.95
Max grad norm	1.0
Value loss coefficient c_1	0.871923

A.9 SWIMMER-V4

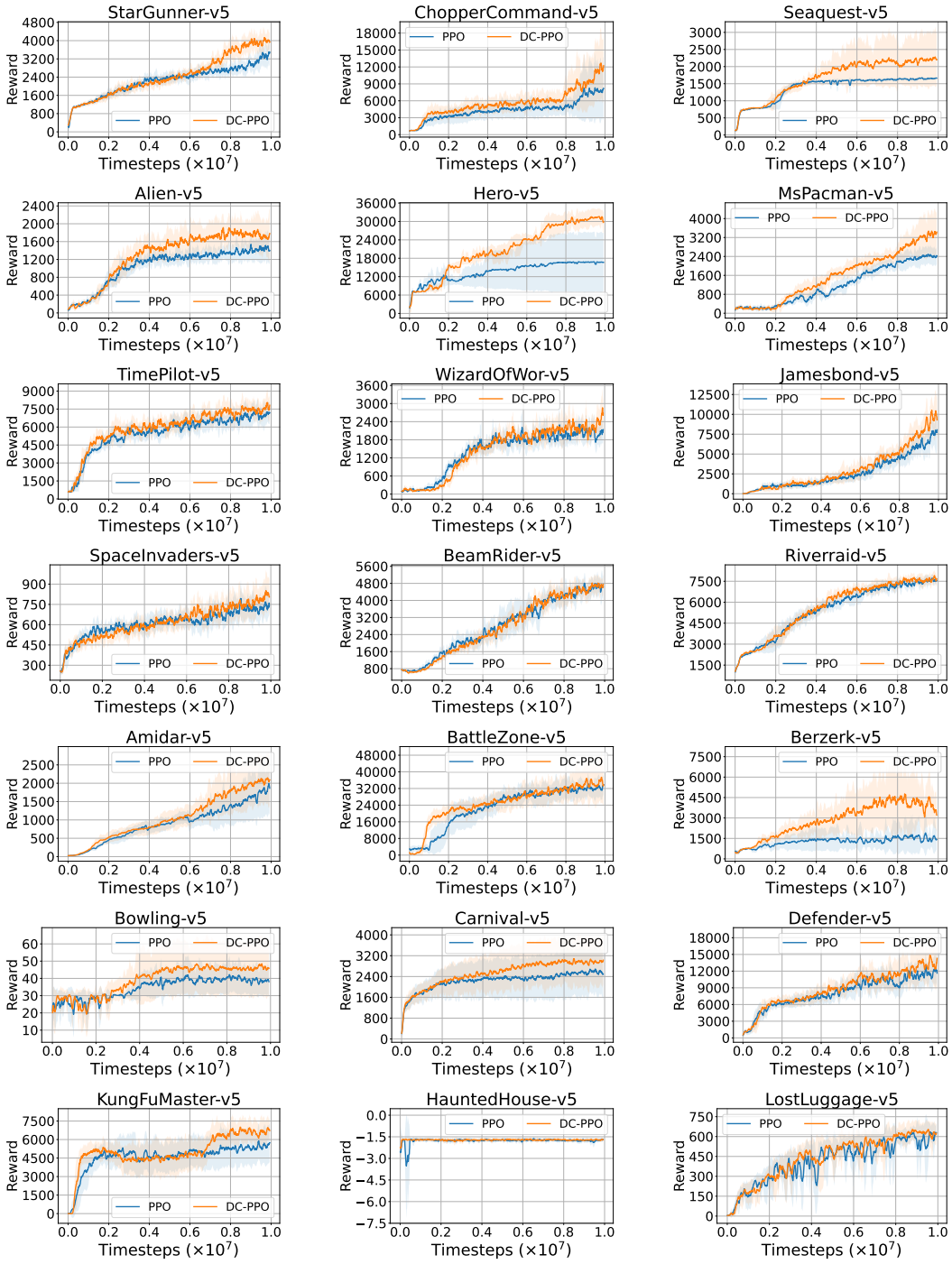
Parameter	Value
Number of workers	4
Policy	MlpPolicy
Total steps	1×10^6
Horizon	1024
Batch size	256
Update epochs	10
Discount factor γ	0.9999
Learning rate	6×10^{-4}
GAE parameter λ	0.98
Clip range	0.2
Entropy loss coefficient c_2	0.0
Value loss coefficient c_1	0.5
Max grad norm	0.5

[2]Not provided in RL-Baselines3-Zoo; we use the same configuration as Reacher-v2.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 PERFORMANCE OF DC-PPO COMPARED TO OTHER VARIANTS

The following figures present additional training results following the same evaluation protocol described in the main text (see Figure 6). Each curve reports the mean episodic return over 10 evaluation episodes, with shaded regions showing one standard deviation. The results are smoothed using a moving-average window of 10. These supplementary plots include additional environments and ablations over the α and β coefficients used in the DC-PPO loss formulation.



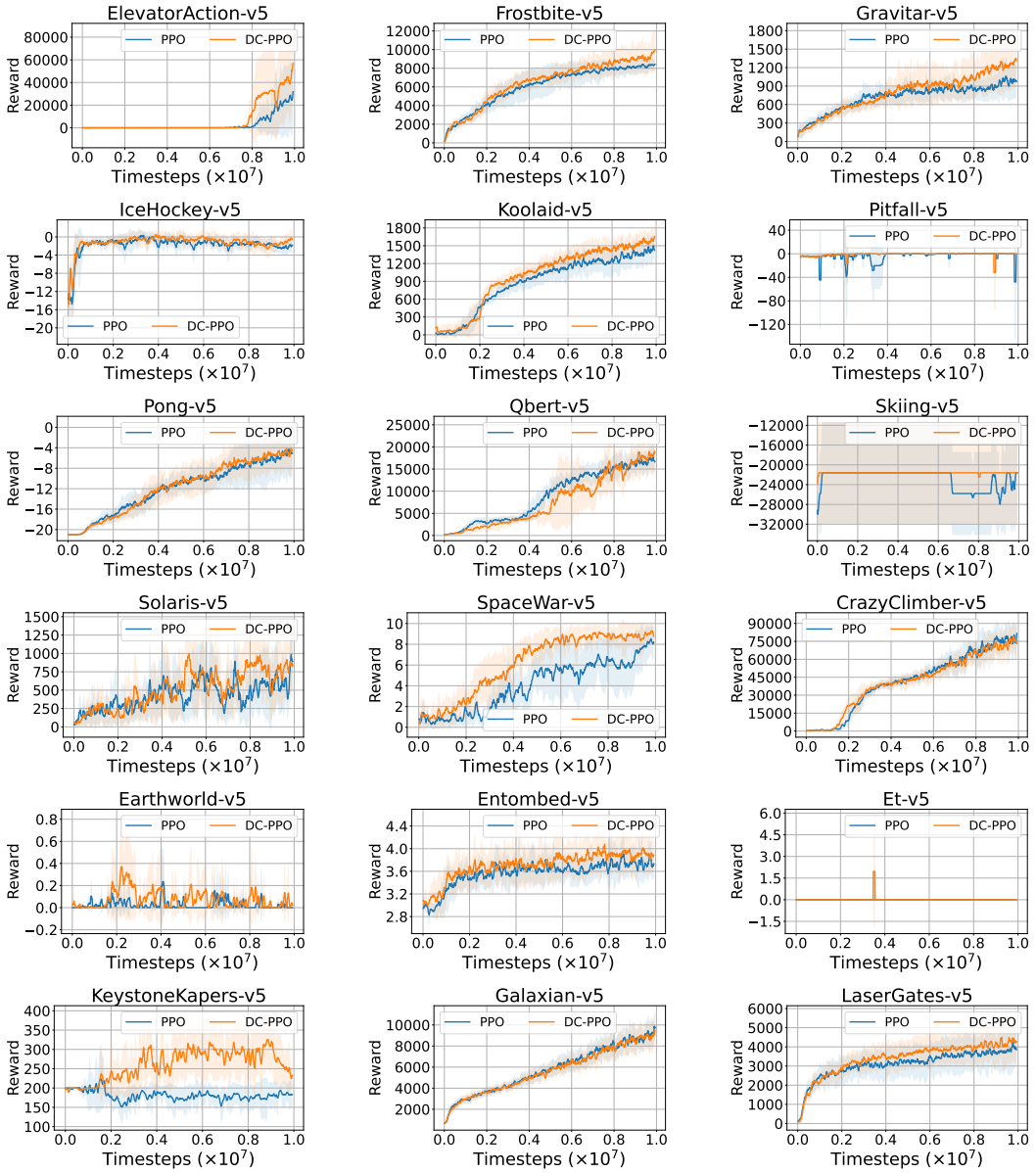


Figure 8: Training performance on Atari benchmarks, following Xie et al. (2025), we report the results with a fixed learning rate of 0.0001. The mean and standard deviation are shown across 5 random seeds.

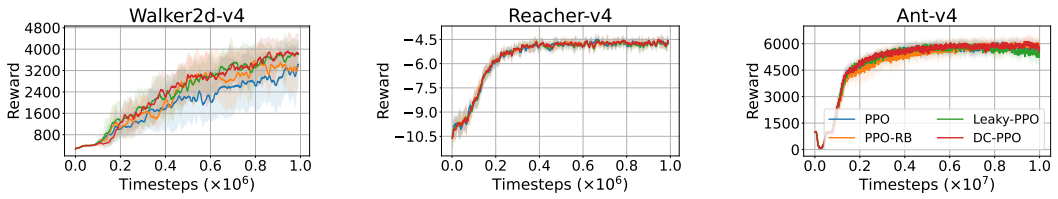


Figure 9: Performance comparisons of PPO, PPO-Leaky, PPO-RB, and DC-PPO (ours) across MuJoCo environments. The mean and standard deviation are shown across 10 random seeds.

B.2 Statistically significance improvement of DC-PPO compared with PPO

Following recent work (Agarwal et al., 2021) we measure the probability of DC-PPO's improvement compared to PPO. To compute the probability of improvement for a task m for algorithms X and Y with $N = 5$ and $K = 5$ runs respectively, we use the Mann-Whitney U-statistic (see Eq. A.2 in Appendix A.7 of Agarwal et al. (2021)). More specifically,

$$\Pr(X_m > Y_m) = \frac{1}{25} \sum_{i=1}^{5} \sum_{j=1}^{5} S(x_{m,i}, y_{m,j})$$

where $S(x, y) = 1$ if $y < x$, $S(x, y) = \frac{1}{2}$ if $y = x$, and 0 otherwise, and $x_{m,i}, y_{m,j}$ denote the average reward over the final 100 evaluation episodes of runs i and j on environment m, respectively.

Across 39 Atari environments, the estimated probability that a randomly selected DC-PPO run outperforms a randomly selected PPO run is 60.41%, with a 95% stratified-bootstrap confidence interval of [54.77%, 65.95%]. This improvement is statistically significant at the 95% confidence level and is unlikely to be explained solely by random-seed variability. DC-PPO obtains a probability of improvement above 50% on 29 of the 39 environments. Below we add a table summarizing the per-game probability that DC-PPO outperforms PPO.

Table 5: Per-game probability that DC-PPO outperforms PPO. Each probability is computed from all $5 \times 5 = 25$ cross-run comparisons.

Environment	$\Pr(\textbf{DC-PPO} > \textbf{PPO})$	**Environment**	$\Pr(\textbf{DC-PPO} > \textbf{PPO})$
StarGunner-v5	60%	Frostbite-v5	64%
ChopperCommand-v5	76%	Gravitar-v5	76%
Seaquest-v5	52%	IceHockey-v5	68%
Alien-v5	48%	Koolaid-v5	100%
Hero-v5	40%	Pitfall-v5	44%
MsPacman-v5	80%	Pong-v5	60%
TimePilot-v5	84%	Qbert-v5	44%
WizardOfWor-v5	92%	Skiing-v5	54%
Jamesbond-v5	60%	Solaris-v5	48%
SpaceInvaders-v5	56%	CrazyClimber-v5	24%
BeamRider-v5	64%	Earthworld-v5	60%
Riverraid-v5	52%	Entombed-v5	76%
AirRaid-v5	68%	Et-v5	50%
SpaceWar-v5	68%	Galaxian-v5	32%
Amidar-v5	52%	HauntedHouse-v5	36%
BattleZone-v5	28%	KeystoneKapers-v5	92%
Berzerk-v5	60%	KungFuMaster-v5	64%
Carnival-v5	68%	LaserGates-v5	64%
Defender-v5	64%	LostLuggage-v5	56%
ElevatorAction-v5	72%		
		Aggregate	**60.41%**

B.3 Implications of regularization strength in DC-PPO

Figure 10 illustrates the sensitivity of DC-PPO to the coefficient α across several MuJoCo environments. Overall, the performance remains relatively stable for moderate values of α, demonstrating the robustness of the method. In Swimmer-v4 and HalfCheetah-v4, higher α values mostly improve convergence speed and final rewards, indicating that stronger directional clamping can facilitate more decisive updates. Conversely, in Hopper-v4 and Walker2d-v4, overly large α occasionally leads to higher variance and slower stabilization, suggesting that moderate clamping (e.g., $\alpha = 3$–4) provides a better trade-off between exploration and stability. Overall, DC-PPO exhibits strong resilience to hyperparameter tuning.

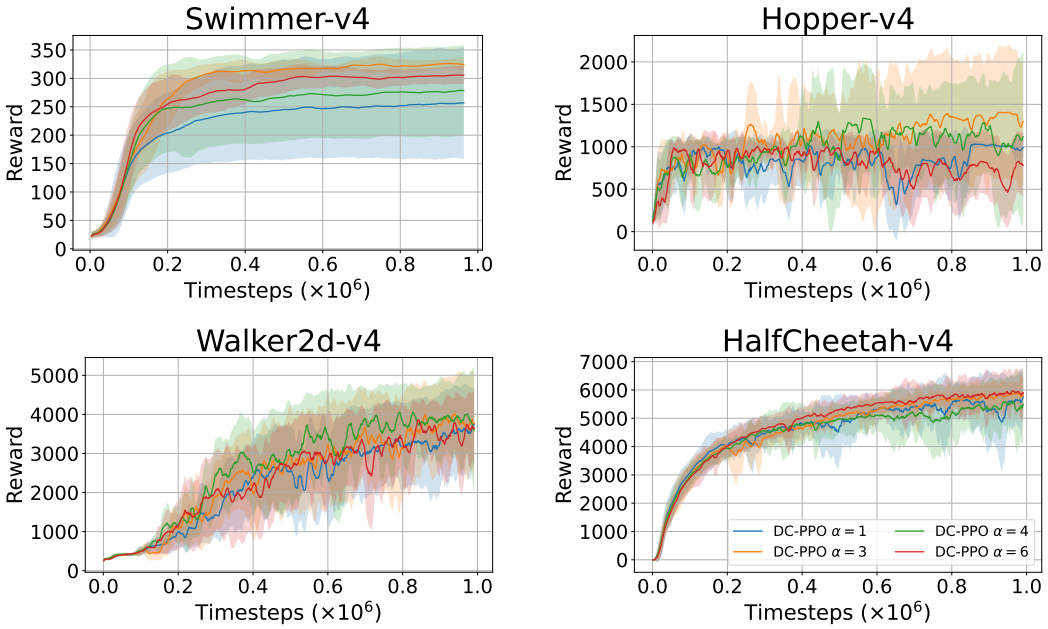


Figure 10: Performance comparisons with different coefficients of α across MuJoCo environments using the optimal coefficient β provided in section A averaged over 5 random seeds.

B.4 Implications of clamping strength in DC-PPO

Figure 11 illustrates the effect of varying the coefficient β in DC-PPO across multiple MuJoCo environments. Overall, β primarily controls the strength of directional correction. Smaller values (e.g., $\beta = 0.01$) impose stronger regularization, which can occasionally slow convergence or introduce mild instability, as observed in Swimmer-v4. In contrast, larger β values yield weaker directional constraints, effectively causing the method to behave increasingly like standard PPO, where ratio deviations are no longer further penalized. Across environments such as HalfCheetah-v4 and Swimmer-v4, intermediate values ($\beta \in [0.1, 0.2]$) generally provide the best balance between stability and learning speed. These results confirm that DC-PPO remains robust to a wide range of β settings, maintaining stable and competitive performance throughout.

In addition to the training curves, Table 6 presents a quantitative comparison of the proportion of ratio samples that fall within the wrong-direction bands for both positive and negative advantages. The proportions are computed over all state–action transitions (s, a) collected during training.

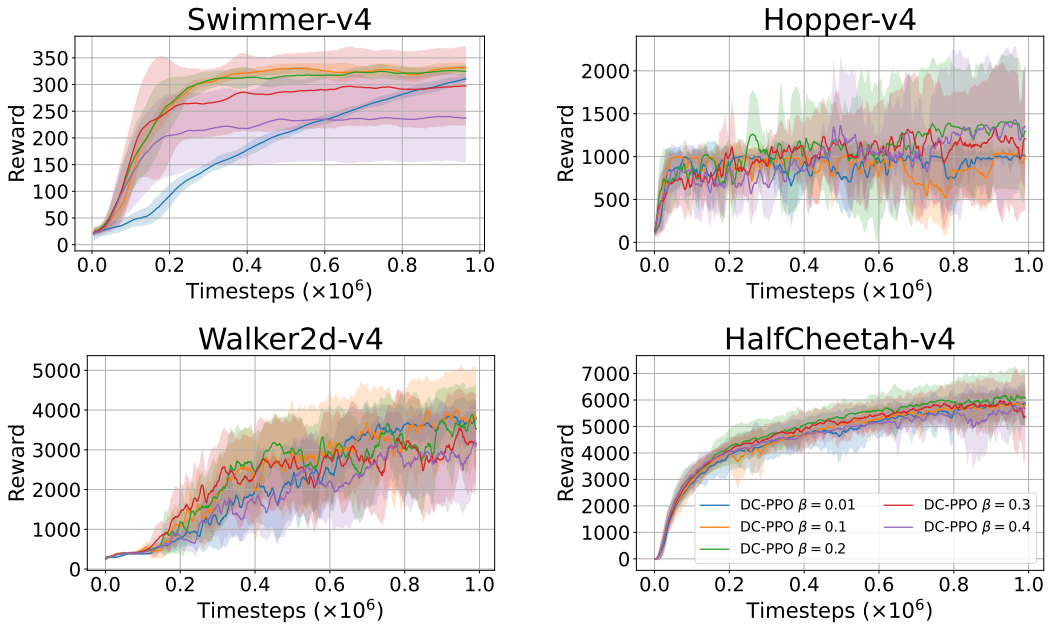


Figure 11: Performance with different β coefficients across MuJoCo environments using the optimal α from section A averaged over 5 random seeds.

B.5 IMPLICATIONS OF CLAMPING STRENGTH ON POSITIVE AND WRONG-DIRECTION SAMPLES

Table 6 compares the proportion of ratio samples that fall within the wrong-direction bands for both positive and negative advantages. Across most MuJoCo environments, DC-PPO (with $\beta = 0.01$) consistently reduces the frequency of wrong-direction updates compared to standard PPO. The overall mean confirms this trend—DC-PPO reduces wrong-direction updates by roughly 4% on average relative to PPO, demonstrating that directional clamping could effectively mitigate wrong-direction updates.

Table 6: Comparison between DC-PPO and PPO in terms of the proportion of samples in the wrong-direction band for positive ($\hat{A}(s_t, a_t) > 0$) and negative ($\hat{A}(s_t, a_t) < 0$) advantages across MuJoCo environments (with the optimal α, β given in section A). Bold indicates the lowest proportion of samples in wrong-direction band out of the three.

Environment	DC-PPO ($\beta = 0.01$)		DC-PPO (Optimal β)		PPO	
	$\hat{A}_t < 0$	$\hat{A}_t > 0$	$\hat{A}_t < 0$	$\hat{A}_t > 0$	$\hat{A}_t < 0$	$\hat{A}_t > 0$
HalfCheetah-v4	**0.2701**	**0.3109**	0.3126	0.3636	0.3092	0.3755
Hopper-v4	0.4484	**0.4115**	0.4017	0.4638	**0.3923**	0.4776
Swimmer-v4	0.4927	**0.4551**	0.4895	0.4772	**0.4682**	0.4931
Walker2d-v4	0.3487	0.3646	0.3474	0.3749	**0.3303**	**0.3610**
Ant-v4	**0.3432**	**0.3668**	0.3568	0.3901	0.3500	0.3866
Humanoid-v4	**0.3096**	**0.3595**	0.3402	0.4129	0.3355	0.4139
Reacher-v4	0.4644	**0.4578**	0.4369	0.4879	**0.4244**	0.4928
Overall mean	0.3824	**0.3895**	0.3836	0.4243	**0.3728**	0.4286

B.6 DC-PPO COMPARISON WITH DC-LEAKY AND DC-RB

Figure 12 presents the training performance of DC-PPO, DC-Leaky, and DC-RB, whereas Tables 7, 8 present the last and top-10 evaluations. Evidently, incorporating the Leaky-PPO objective or PPO-RB yields only marginal gains and, in several cases, degrades performance.

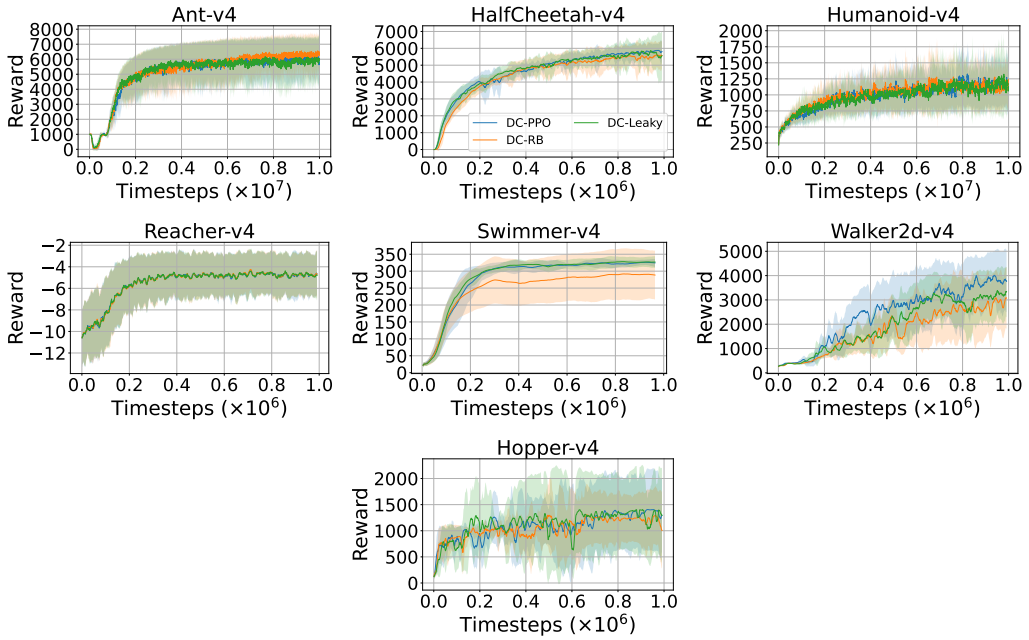


Figure 12: Performance comparisons of DC-PPO, DC-Leaky, and DC-RB across MuJoCo environments.

Table 7: Average of the last-10 evaluations across MuJoCo environments.

Environment	PPO	PPO-RB	Leaky PPO	DC-PPO	DC-RB	DC-Leaky
HalfCheetah-v4	5753.40	5494.25	5468.16	**5816.94**	5569.56	5567.40
Hopper-v4	937.63	1006.24	525.76	**1296.55**	1004.28	1295.42
Swimmer-v4	256.83	289.36	296.24	324.42	288.49	**326.99**
Ant-v4	5813.77	5969.68	5973.11	5763.29	**6317.02**	5868.17
Reacher-v4	-4.78	-4.67	-4.72	**-4.66**	-4.72	-4.80
Humanoid-v4	900.60	929.90	914.12	1137.30	**1241.08**	1118.55
Walker2d-v4	3750.60	3078.21	3842.29	**3765.47**	3073.59	3336.82

Table 8: Average of the top-10 evaluations across MuJoCo environments.

Environment	PPO	PPO-RB	Leaky PPO	DC-PPO	DC-RB	DC-Leaky
HalfCheetah-v4	5945.76	5617.86	5840.40	5951.39	5756.71	**5991.84**
Hopper-v4	1017.51	1016.92	1009.07	1413.65	1401.22	**1421.22**
Swimmer-v4	259.19	294.28	300.20	327.82	294.70	**330.69**
Ant-v4	6596.15	6738.53	6560.68	6673.82	**6860.43**	6632.02
Reacher-v4	-3.31	-3.44	-3.66	-3.42	**-3.40**	-3.51
Humanoid-v4	1119.66	1244.01	1054.83	1486.52	1496.36	**1507.39**
Walker2d-v4	4024.42	3953.52	4214.24	**4294.54**	3203.04	3805.75

Variants that apply clamping in the wrong direction consistently yield the stronger performance, outperforming standard PPO, PPO-RB, and Leaky-PPO across all environments, as shown in tables 7 and 8, under both evaluation metrics. While DC-PPO achieves the strongest last-10 evaluation performance, the combined variants often surpass standard DC-PPO and attain higher top-10 evaluation

rewards. Across most MuJoCo environments incorporating clamping in the wrong direction led to improved performance overall.

C PROOF OF LEMMA 1

Proof. Since $t \in \Omega_T$, we either have that $w_{\theta_0}(s_t, a_t) < 1 - \beta$ and $\hat{A}(s_t, a_t) > 0$ or $w_{\theta_0}(s_t, a_t) > 1 + \beta$ and $\hat{A}(s_t, a_t) < 0$. We will show the result for the first case, as the second case is analogous. For simplicity of the proof, we define

$$\phi(\gamma) = w_{\theta_0 + \gamma \nabla \mathcal{J}_{\text{DC-PPO}}(\theta_0)}(s_t, a_t) - w_{\theta_0 + \gamma \nabla \mathcal{J}_{\text{PPO}}(\theta_0)}(s_t, a_t).$$

Then, by computing its gradient, we get

$$\begin{aligned} \phi'(0) &= \nabla w_{\theta_0}(s_t, a_t)^\top \left(\nabla \mathcal{J}_{\text{DC-PPO}}(\theta_0) - \nabla \mathcal{J}_{\text{PPO}}(\theta_0) \right) \\ &= (\alpha - 1) \frac{1}{|\Omega_T|} \sum_{t' \in \Omega_T} \langle \nabla w_{\theta_0}(s_t, a_t), \nabla w_{\theta_0}(s_{t'}, a_{t'}) \rangle \hat{A}(s_{t'}, a_{t'}) \end{aligned}$$

where the equality follows from the definition of $\mathcal{J}_{\text{DC-PPO}}$. Since $\alpha > 1$ and the assumption, we get $\phi'(0) > 0$. Hence, there exists $\bar{\gamma} > 0$ such that for all $\gamma \in (0, \bar{\gamma})$, we have $\phi(\gamma) > \phi(0)$. This implies that

$$w_{\theta_1^{\text{PPO}}}(s_t, a_t) < w_{\theta_1^{\text{DC-PPO}}}(s_t, a_t). \tag{5}$$

Moreover, from the continuity of $w_\theta(s_t, a_t)$ in θ there exists $\bar{\delta} > 0$ such that for all $\delta \in (0, \bar{\delta})$ we have $|w_{\theta_0 + \delta \hat{u}}(s_t, a_t) - w_{\theta_0}(s_t, a_t)| < 1 - w_{\theta_0}(s_t, a_t)$ for $\hat{u} = \frac{\nabla \mathcal{J}_{\text{DC-PPO}}(\theta_0)}{\|\nabla \mathcal{J}_{\text{DC-PPO}}(\theta_0)\|}$[3]. This implies that $w_{\theta_0 + \delta \hat{u}}(s_t, a_t) < 1$. For all $\gamma \in (0, \min(\bar{\gamma}, \delta \|\nabla \mathcal{J}_{\text{DC-PPO}}(\theta_0)\|))$, by combining with equation 5 yields that $w_{\theta_1^{\text{PPO}}}(s_t, a_t) < w_{\theta_1^{\text{DC-PPO}}}(s_t, a_t) < 1$. Therefore

$$|w_{\theta_1^{\text{DC-PPO}}}(s_t, a_t) - 1|^2 < |w_{\theta_1^{\text{PPO}}}(s_t, a_t) - 1|^2,$$

which proves the desired result. □

D DC-PPO GLOBAL CONVERGENCE PROOF

D.1 PSEUDO CODE OF THE ALGORITHMS

Algorithm 2 Tabular DC-PPO

1: **Initialization:** policy $\pi^{(0)} = \pi(\theta^{(0)})$, initial state distribution μ, step size of EMDA η, number of EMDA iterations $K^{(t)}$
2: **Output:** Learned policy $\pi^{(\infty)}$
3: **for** $t = 0, 1, \ldots$ **do**
4: Collect a set of trajectories $t \in \mathcal{D}^{(t)}$ under policy $\pi^{(t)} = \pi(\theta^{(t)})$
5: Find $A^{(t)}$ by a policy evaluation method
6: Compute $\hat{L}^{(t)}(\theta)$ based on $A^{\pi^{(t)}}$ and the collected samples in $\mathcal{D}^{(t)}$
7: Update the policy by $\theta^{(t+1)} = \text{EMDA-tabular}\big(\hat{L}^{(t)}(\theta), \eta, K^{(t)}, \mathcal{D}^{(t)}, \theta^{(t)}\big)$
8: **end for**

[3] $\hat{u}$ is well defined by the assumption.

Algorithm 3 EMDA-tabular($L(\theta), \eta, K, \mathcal{D}, \theta_{\text{init}}$)

Input: Objective $L(\theta)$, step size η, number of iterations K, dataset $\mathcal{D}$, initial parameter θ_{init}
Initialization: $\tilde{\theta}^{(0)} = \theta_{\text{init}}$, $\hat{\theta} = \theta_{\text{init}}$
Output: Learned parameter $\tilde{\theta}$
for $k = 0, \ldots, K-1$ **do**
  for each state s in $\mathcal{D}$ **do**
    Find $g_{s,a}^{(k)} = \frac{\partial L(\theta)}{\partial \theta_{s,a}}\Big|_{\theta=\tilde{\theta}^{(k)}}$ for each a
    Let $w_s = \left(e^{-\eta g_{s,1}^{(k)}}, \ldots, e^{-\eta g_{s,|\mathcal{A}|}^{(k)}}\right)$
    $\tilde{\theta}_s^{(k+1)} = \frac{1}{\langle w_s, \tilde{\theta}_s^{(k)} \rangle}\left(w_s \circ \tilde{\theta}_s^{(k)}\right)$
  end for
end for

Definition 1. *Let π_1, π_2 be two policies. Then π_1 improves upon π_2 ($\pi_1 \geq \pi_2$) if and only if $V^{(\pi_1)}(s) \geq V^{(\pi_2)}(s) \quad \forall s \in S$, and π_1 strictly improves upon π_2 ($\pi_1 > \pi_2$) if and only if $\pi_1 \geq \pi_2$ and there exists $s \in S$ such that $V^{(\pi_1)}(s) > V^{(\pi_2)}(s)$.*

Lemma 2 (Performance difference lemma Kakade & Langford (2002))**.** *Given any two policies π_1 and π_2, we have*

$$V^{\pi_1}(s) - V^{\pi_2}(s) = \frac{1}{1-\gamma} \sum_{s' \in S} d_s^{\pi_1}(s') \sum_{a \in \mathcal{A}} \pi_1(a|s') A^{\pi_2}(s', a)$$

Lemma 3 (Adopted from Lemma 12 in Huang et al. (2024) for PPO)**.** *In iteration t, suppose $\pi^{(t)}$ is strictly positive, for all state and action pairs, i.e., $\forall (s,a),\ \pi^{(t)}(a|s) > 0$, then under DC-PPO in algorithm 2, $\pi^{(t+1)}$ satisfies that, $\pi^{(t+1)} \geq \pi^{(t)}$ and $\forall (s,a),\ \pi^{(t+1)}(a|s) > 0$.*

Proof. We first show that $\forall (s,a),\ \pi^{(t+1)}(a|s) > 0$.

Let $(s,a) \in S \times \mathcal{A}$.

For ease of exposition, we define $g_{s,a}^{(k)} := -\frac{\partial \mathcal{L}(\theta)}{\partial \theta_{s,a}}\big|_{\theta=\hat{\theta}^{(k)}}$ and $w^{(k)} := \left(e^{-\eta g_{s,1}^{(k)}}, \ldots, e^{-\eta g_{s,|\mathcal{A}|}^{(k)}}\right)$ as shown in algorithm 3.

By the update rule in algorithm 3,

$$\pi^{(t+1)}(a|s) = \frac{\prod_{k=0}^{K^{(t)}-1} \exp\left(-\eta g_{s,a}^{(k)}\right)}{\prod_{k=0}^{K^{(t)}-1} \langle w_s^{(k)}, \tilde{\theta}_s^{(k)} \rangle} \pi^{(t)}(a|s). \tag{6}$$

If $(s,a) \in \mathcal{D}^{(t)}$, then $\pi^{(t)}(a|s) > 0$, and since for every k the vectors $w_s^{(k)}$ and $\tilde{\theta}_s^{(k)}$ are positive, it follows that $\pi^{(t+1)}(a|s) > 0$. In the complementary case, $\pi^{(t)}(a|s)$ is not modified, and because $\pi^{(t)}(a|s) > 0$, we also have $\pi^{(t+1)}(a|s) > 0$.

In eq. (6) $g_{(s,a)}^{(k)}$ can be written as,

$$g_{(s,a)}^{(k)} = \begin{cases} -\alpha \frac{\text{sign}(A^{\pi^{(t)}}(s,a))}{\pi^{(t)}(a|s)}, & \text{if } \tilde{\theta}_{s,a}^{(k)} \in \Theta_t, (s,a) \in \mathcal{D}^{(t)} \\ -\frac{\text{sign}(A^{\pi^{(t)}}(s,a))}{\pi^{(t)}(a|s)}, & \text{if } (\frac{\tilde{\theta}_{s,a}^{(k)}}{\pi^{(t)}(a|s)} - 1)\text{sign}(A^{\pi^{(t)}}(s,a)) < \epsilon, \tilde{\theta}_{s,a}^{(k)} \notin \Theta_t, (s,a) \in \mathcal{D}^{(t)} \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where

$$\Theta_t = \left\{\theta \mid \left|\frac{\theta}{\pi^{(t)}(a|s)} - 1\right| > \beta \wedge \left(\frac{\theta}{\pi^{(t)}(a|s)} - 1\right) A^{\pi^{(t)}}(s,a) < 0\right\}.$$

By Lemma 2 it is sufficient to show that $\sum_{a\in\mathcal{A}} \pi^{(t+1)}(a|s)A^{\pi^{(t)}}(s,a) \geq 0 \quad \forall s \in S$ to have $\pi^{(t+1)} \geq \pi^{(t)}$.

For all $(s,a) \in \mathcal{D}^{(t)}$ with positive advantage, we have $\prod_{k=0}^{K^{(t)}-1} \exp(\underbrace{-\eta g_{(s,a)}^{(k)}}_{>0}) > 1$, and similarly for all $(s,a) \in \mathcal{D}^{(t)}$ with negative advantage $\prod_{k=0}^{K^{(t)}-1} \exp(\underbrace{-\eta g_{(s,a)}^{(k)}}_{<0}) < 1$, furthermore since $\sum_{a\in\mathcal{A}} \pi^{(t)}(a|s)A^{\pi^{(t)}}(s,a) = 0$ for all $s \in S$, we have,

$$\sum_{a\in\mathcal{A}} \pi^{(t+1)}(a|s)A^{\pi^{(t)}}(s,a)$$
$$= \frac{1}{\prod_{k=0}^{K^{(t)}-1}\langle w_s^{(k)}, \tilde{\theta}_s^{(k)}\rangle} \sum_{a\in\mathcal{A}} \left(\prod_{k=0}^{K^{(t)}-1} \exp\left(-\eta g_{s,a}^{(k)}\right) \right) \pi^{(t)}(a|s)A^{\pi^{(t)}}(s,a) \geq 0.$$

□

Recall theorem 1 . Under Assumptions 1 and 2, and DC-PPO Objective we have $V^{\pi^{(t)}}(s) \to V^{(\pi^\star)}(s)$ as $t \to \infty \; \forall s \in S$, almost surely.

Proof. The following proof is similar in spirit to the arguments in Huang et al. (2024); see Lemmas 12–14. First of all, we note that by lemma 3 the limits $V^{(\infty)}(s), Q^{(\infty)}(s,a), A^{(\infty)}(s,a)$ exist for all trajectory sampling. Moreover, by Lemma 3 we know that $V^{\pi^{(t+1)}}(s) \geq V^{\pi^{(t)}}(s) \quad \forall s \in S$. Furthermore, by the properties of bounded reward and discounted setting, $-\frac{R_{\max}}{1-\gamma} \leq V^{\pi^{(t)}}(s) \leq \frac{R_{\max}}{1-\gamma}$. Since $\{V^{\pi^{(t)}}(s) : t \geq 1\}$ is a monotonic and bounded series, it convergences, i.e. $V^{\pi^{(t)}}(s) \to V^{\pi^{(\infty)}}(s)$ for each $s \in S$. Similarly, $Q^{\pi^{(t)}}(s,a) \to Q^{\pi^{(\infty)}}(s,a)$, and thus it must hold $A^{\pi^{(t)}}(s,a) \to A^{\pi^{(\infty)}}(s,a)$, we left to prove that $V^{\pi^{(\infty)}}(s) = V^{(\pi^\star)}(s) \quad \forall s \in S$. We proceed by showing that $\forall (s,a),\ A^{\pi^{(\infty)}}(s,a) \leq 0$, which is sufficient to prove convergence to optimal policy, this can be inferred by lemma 2 if $\forall (s,a),\ A^{\pi}(s,a) \leq 0$ for some policy π then for any different policy π' we have $V^{\pi'}(s) - V^{\pi}(s) \leq 0 \quad \forall s \in S$.

For convenience we define the sets

$$\mathcal{A}_+^\infty(s) = \{a \in \mathcal{A} : A^{\pi^{(\infty)}}(s,a) > 0\},$$
$$\mathcal{A}_-^\infty(s) = \{a \in \mathcal{A} : A^{\pi^{(\infty)}}(s,a) < 0\},$$
$$\mathcal{A}_0^\infty(s) = \{a \in \mathcal{A} : A^{\pi^{(\infty)}}(s,a) = 0\}.$$

Assume that the set $\mathcal{A}_+^\infty(s)$ is non-empty for some $s \in S$, and let $\Delta = \min_{a\in\mathcal{A}_+^\infty(s)} A^{\pi^{(\infty)}}(s,a) > 0$. We select any arbitrary $a' \in \mathcal{A}_+^\infty(s)$, then by limit rules there exists T such that for all $t \geq T'$ we have $A^{\pi^{(t)}}(s,a') > \frac{\Delta}{2}$.

By the update rule in algorithm 3 we have,

$$\pi^{(t+1)}(a' \mid s) = \frac{\prod_{k=0}^{K^{(t)}-1} \exp(-\eta g_{s,a'}^{(k)})\pi^{(t)}(a' \mid s)}{\prod_{k=0}^{K^{(t)}-1}\langle w_s^{(k)}, \tilde{\theta}_s^{(k)}\rangle}$$
$$\geq \frac{\pi^{(t)}(a' \mid s)\exp(-\eta g_{s,a'}^{(0)})}{\pi^{(t)}(a' \mid s)\exp(-\eta g_{s,a'}^{(0)}) + 1} \tag{8}$$
$$= \frac{\pi^{(t)}(a' \mid s)\exp\left(\eta/\pi^{(t)}(a' \mid s)\right)}{\pi^{(t)}(a' \mid s)\exp\left(\eta/\pi^{(t)}(a' \mid s)\right) + 1}$$
$$\geq \frac{e\eta}{e\eta + 1} := \psi, \tag{9}$$

where equation 8 holds due to $\tilde{\theta}^{(k)}_{s,a'}$ being non-decreasing under assumption 2 and by the definition of $g^{(k)}_{s,a}$ in eq. (7). eq. (9) follows since $f(z) = \frac{ze^{\eta/z}}{ze^{\eta/z}+1}$ has a unique minimizer at $z = \eta$.

If $\mathcal{A}^{\infty}_0(s)$ is non-empty, then there exists some T'' such that for all $t \geq T''$, we get $|A^{\pi^{(t)}}(s,a)| < \frac{\psi\Delta}{4}$. Since $\sum_{a\in\mathcal{A}} \pi^{(t)}(a|s)A^{\pi^{(t)}}(s,a) = 0$, then for all $t > \max(T', T'')$ we have,

$$\begin{aligned}\sum_{a\in\mathcal{A}^{\infty}_{-}(s)} \pi^{(t)}(a \mid s) &\geq \frac{\sum_{a\in\mathcal{A}^{\infty}_{+}(s)} \pi^{(t)}(a \mid s)A^{\pi^{(t)}}(s,a) + \sum_{a\in\mathcal{A}^{\infty}_{0}(s)} \pi^{(t)}(a \mid s)A^{\pi^{(t)}}(s,a)}{\max_{a\in\mathcal{A}^{-}_{+}(s)}|A^{\pi^{(t)}}(s,a)|} \\ &\geq \frac{\psi(\Delta/2) - (\psi\Delta/4)}{\frac{2R_{\max}}{1-\gamma}} \\ &= \frac{\psi\Delta}{\frac{8R_{\max}}{1-\gamma}}.\end{aligned} \tag{10}$$

We have now shown that $\sum_{a\in\mathcal{A}^{\infty}_{-}(s)} \pi^{(t)}(a \mid s)$ is bounded below by a positive constant; in the next step, we prove that $\sum_{a\in\mathcal{A}^{\infty}_{-}(s)} \pi^{(t)}(a \mid s) \to 0$, which yields a contradiction.

Since $\sum_{a\in\mathcal{A}} \pi^{(\infty)}(a \mid s)A^{\pi^{(\infty)}}(s,a) = 0$, there exists at least one action $a'' \in \mathcal{A}$ such that $\pi^{(\infty)}(a'' \mid s) > 0$ and $A^{\pi^{(\infty)}}(s,a'') < 0$. Fix any $a' \in \mathcal{A}^{\infty}_{+}(s)$ and such an action $a'' \in \mathcal{A}^{\infty}_{-}(s)$. By Assumption 2, at each iteration at most one action per state appears in $\mathcal{D}^{(t)}$, and therefore the following three cases exhaust all possible configurations of (s,a') and (s,a'') in $\mathcal{D}^{(t)}$. As before we take T large enough such that for all $t \geq T$ we have $A^{\pi^{(t)}}(s,a'') < 0$ and $A^{\pi^{(t)}}(s,a') > 0$.

Case 1: $(s,a') \in \mathcal{D}^{(t)},\ (s,a'') \notin \mathcal{D}^{(t)}$.

By the EMDA-tabular (algorithm 3) and eq. (6), we have

$$\frac{\pi^{(t+1)}(a'' \mid s)}{\pi^{(t+1)}(a' \mid s)} = \frac{\pi^{(t)}(a'' \mid s)}{\pi^{(t)}(a' \mid s)} \prod_{k=0}^{K^{(t)}-1} \exp\left(\eta g^{(k)}_{s,a'}\right) \leq \frac{\pi^{(t)}(a'' \mid s)}{\pi^{(t)}(a' \mid s)} \exp(-\eta) < 1, \tag{11}$$

where the last inequality holds by eq. (7), $a' \in \mathcal{A}^{\infty}_{+}(s)$, and $\pi^{(t)}(a' \mid s) \leq 1$.

Case 2: $(s,a') \notin \mathcal{D}^{(t)},\ (s,a'') \in \mathcal{D}^{(t)}$.

In this case by the EMDA-tabular (algorithm 3) we have $-g^{(0)}_{s,a''} < 0$ and $-g^{(k)}_{s,a''} \leq 0$ for all $k \in \{1,\dots,K^{(t)}\}$. Therefore,

$$\frac{\pi^{(t+1)}(a'' \mid s)}{\pi^{(t+1)}(a' \mid s)} < \frac{\pi^{(t)}(a'' \mid s)}{\pi^{(t)}(a' \mid s)}. \tag{12}$$

Case 3: $(s,a') \notin \mathcal{D}^{(t)},\ (s,a'') \notin \mathcal{D}^{(t)}$.

Under the EMDA-tabular, since neither (s,a') nor (s,a'') belongs to $\mathcal{D}^{(t)}$, the update does not affect their relative weights. Consequently, the action probability ratio between a' and a'' remains unchanged. This holds even though the individual probabilities $\pi^{(t)}(a'' \mid s)$ and $\pi^{(t)}(a' \mid s)$ may still change due to updates on a third action $a''' \neq a', a''$ with $(s,a''') \in \mathcal{D}^{(t)}$. Formally,

$$\frac{\pi^{(t+1)}(a'' \mid s)}{\pi^{(t+1)}(a' \mid s)} = \frac{\pi^{(t)}(a'' \mid s)}{\pi^{(t)}(a' \mid s)}. \tag{13}$$

By assumption 1, each state–action pair (s,a) appears in $\{\mathcal{D}^{(t)}\}$ infinitely often. By assumption 2, at most one action per state appears in $\mathcal{D}^{(t)}$ in each iteration. Therefore, for any $a' \in \mathcal{A}^{\infty}_{+}(s)$ and

$a'' \in \mathcal{A}_{-}^{\infty}(s)$, Case 1 occurs infinitely often whenever $(s, a') \in \mathcal{D}^{(t)}$, and Case 2 occurs infinitely often whenever $(s, a'') \in \mathcal{D}^{(t)}$. Combining the cases, we conclude that

$$\frac{\pi^{(t)}(a'' \mid s)}{\pi^{(t)}(a' \mid s)} \longrightarrow 0 \quad \text{as } t \to \infty,$$

for every $a'' \in \mathcal{A}_{\infty}^{-}(s)$ and any $a' \in \mathcal{A}_{\infty}^{+}(s)$.

This yields a contradiction. Indeed, by equation 9 we have that for all sufficiently large t, $\pi^{(t)}(a' \mid s) \geq \frac{e\eta}{e\eta+1} > 0$. Moreover, we have shown that for every $a'' \in \mathcal{A}_{-}^{\infty}(s)$, $\pi^{(t)}(a'' \mid s)/\pi^{(t)}(a' \mid s) \to 0$, hence $\sum_{a'' \in \mathcal{A}_{-}^{\infty}(s)} \pi^{(t)}(a'' \mid s) \longrightarrow 0$. This contradicts the lower bound equation 10, which shows that $\sum_{a'' \in \mathcal{A}_{-}^{\infty}(s)} \pi^{(t)}(a'' \mid s)$ is bounded below by a positive constant for all sufficiently large t. Therefore, $\mathcal{A}_{+}^{\infty}(s)$ must be empty for every $s \in S$, i.e., $A^{\pi^{(\infty)}}(s, a) \leq 0$ for all (s, a).

□