A DEEPER LOOK AT DISCOUNTING MISMATCH IN ACTOR-CRITIC ALGORITHMS

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the discounting mismatch in actor-critic algorithm implementations from a representation learning perspective. Theoretically, actor-critic algorithms usually have discounting for both actor and critic, *i.e.*, there is a γ^t term in the actor update for the transition observed at time t in a trajectory and the critic is a discounted value function. Practitioners, however, usually ignore the discounting (γ^t) for the actor while using a discounted critic. We investigate this mismatch in two scenarios. In the first scenario, we consider optimizing an undiscounted objective ($\gamma = 1$) where γ^t disappears naturally ($1^t = 1$). We then propose to interpret the discounting in critic in terms of a *bias-variance-representation* trade-off and provide supporting empirical results. In the second scenario, we consider optimizing a discounted objective ($\gamma < 1$) and propose to interpret the omission of the discounting in the actor update from an *auxiliary task* perspective and provide supporting empirical results.

1 INTRODUCTION

Actor-critic algorithms have enjoyed great success both theoretically (Williams, 1992; Sutton et al., 2000; Konda, 2002; Schulman et al., 2015a) and empirically (Mnih et al., 2016; Silver et al., 2016; Schulman et al., 2017; OpenAI, 2018). There is, however, a longstanding gap between the theory behind actor-critic algorithms and how practitioners implement them. Let $\gamma, \gamma_{\text{A}}$, and γ_{C} be the discount factors for defining the objective, updating the actor, and updating the critic respectively. Theoretically, no matter whether $\gamma = 1$ or $\gamma < 1$, we should always use $\gamma_{\text{A}} = \gamma_{\text{C}} = \gamma$ (Sutton et al., 2000; Schulman et al., 2015a) or at least keep $\gamma_{\text{A}} = \gamma_{\text{C}}$ if Blackwell optimality (Veinott, 1969; Weitzman, 2001) [1] is considered. Practitioners, however, usually use $\gamma_{\text{A}} = 1$ and $\gamma_{\text{C}} < 1$ in their implementations (Dhariwal et al., 2017; Caspi et al., 2017; Zhang, 2018; Kostrikov, 2018; Achiam, 2018; Liang et al., 2018; Stooke & Abbeel, 2019). Although this mismatch and its theoretical disadvantage have been recognized by Thomas (2014); Nota & Thomas (2020), whether and why it yields benefits in practice has not been systematically studied. In this paper, we empirically investigate this mismatch from a representation learning perspective. We consider two scenarios separately.

Scenario 1: *The true objective is undiscounted ($\gamma = 1$).* The theory prescribes to use $\gamma_{\text{A}} = \gamma_{\text{C}} = \gamma = 1$. Practitioners, however, usually use $\gamma_{\text{A}} = \gamma = 1$ but $\gamma_{\text{C}} < 1$, introducing *bias*. We explain this mismatch with the following hypothesis:

Hypothesis 1. $\gamma_{\text{C}} < 1$ *optimizes a bias-variance-representation trade-off.*

It is easy to see that $\gamma_{\text{C}} < 1$ reduces the variance in bootstrapping targets. Besides this, we further provide empirical evidence showing that when $\gamma_{\text{C}} < 1$, it may become easier to find a good representation compared to $\gamma_{\text{C}} = 1$. Consequently, although using $\gamma_{\text{C}} < 1$ introduces bias, it can facilitate representation learning. For our empirical study, we make use of recently introduced techniques, such fixed horizon temporal different learning (De Asis et al., 2019) and distributional reinforcement learning (Bellemare et al., 2017) to disentangle the various effects the discount factor has on the learning process.

[1] Blackwell optimality states that, in finite MDPs, there exists a $\gamma_0 < 1$ such that for all $\gamma \geq \gamma_0$, the optimal policies for the γ-discounted objective are the same.

Scenario 2: *The true objective function is discounted ($\gamma < 1$).* Theoretically, there is a γ^t term for the actor update on a transition observed at time t in a trajectory (Sutton et al., 2000; Schulman et al., 2015a). Practitioners, however, usually ignore this term while using a discounted critic, *i.e.*, $\gamma_{\text{A}} = 1$ and $\gamma_{\text{C}} = \gamma < 1$ are used. We explain this mismatch with the following hypothesis:

Hypothesis 2. *Using $\gamma_{\text{C}} = \gamma < 1$ and $\gamma_{\text{A}} = 1$ is effectively similar to using $\gamma_{\text{C}} = \gamma_{\text{A}} = \gamma < 1$ plus an auxiliary loss that sometimes facilitates representation learning.*

Our empirical study involves implementing the auxiliary task explicitly by using an additional policy for optimizing the difference term between the loss of $\gamma_{\text{A}} = 1$ and the loss of $\gamma_{\text{A}} < 1$. We also design new benchmarking environments where the sign of the reward function is flipped after a certain time step such that later transitions differ from earlier ones. In that setting, $\gamma_{\text{A}} = 1$ becomes harmful.

2 Background

Markov Decision Processes: We consider an infinite horizon MDP with a finite state space $\mathcal{S}$, a finite action space $\mathcal{A}$, a bounded reward function $r : \mathcal{S} \to \mathbb{R}$, a transition kernel $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, an initial state distribution μ_0, and a discount factor $\gamma \in [0, 1]$.[2] The initial state S_0 is sampled from μ_0. At time step t, an agent in state S_t takes action $A_t \sim \pi(\cdot|S_t)$, where $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the policy it follows. The agent then gets a reward $R_{t+1} \doteq r(S_t)$ and proceeds to the next state $S_{t+1} \sim p(\cdot|S_t, A_t)$. The return of the policy π at time step t is defined as $G_t \doteq \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i}$, which allows us to define the state value function $v_\pi^\gamma(S) \doteq \mathbb{E}[G_t|S_t = s]$ and the state-action value function $q_\pi^\gamma(s, a) \doteq \mathbb{E}[G_t|S_t = s, A_t = a]$. We consider episodic tasks where we assume there is an absorbing state $s^\infty \in \mathcal{S}$ such that $r(s^\infty) = 0$ and $p(s^\infty|s^\infty, a) = 1$ holds for any $a \in \mathcal{A}$. When $\gamma < 1$, v_π^γ and q_π^γ are always well defined. When $\gamma = 1$, to ensure v_π^γ and q_π^γ are well defined, we further assume finite expected episode length. Let T_s^π be a random variable denoting the first time step that an agent hits s^∞ when following π given $S_0 = s$. We assume $T_{\max} \doteq \sup_{\pi \in \Pi} \max_s \mathbb{E}[T_s^\pi] < \infty$, where π is parameterized by θ and Π is the corresponding function class. Similar assumptions are also used in stochastic shortest path problems (*e.g.*, Section 2.2 of Bertsekas & Tsitsiklis (1996)). In our experiments, all the environments have a hard time limit of 1000, i.e., $T_{\max} = 1000$. This is standard practice, classic RL environments also have an upper limit on their episode lengths (*e.g.* 27k in Bellemare et al. (2013, ALE)). Following Pardo et al. (2018), we add the (normalized) time step t in the state to keep the environment Markovian. We measure the performance of a policy π with $J_\gamma(\pi) \doteq \mathbb{E}_{S_0 \sim \mu_0}[v_\pi^\gamma(S_0)]$.

γ	define the objective
γ_{A}	update the actor
γ_{C}	update the critic

Table 1: Roles of the different discount factors

Vanilla Policy Gradient: Sutton et al. (2000) compute $\nabla_\theta J_\gamma(\pi)$ as

$$\nabla_\theta J_\gamma(\pi) \doteq \sum_s d_\pi^\gamma(s) \sum_a q_\pi^\gamma(s, a) \nabla_\theta \pi(a|s), \tag{1}$$

where $d_\pi^\gamma(s) \doteq \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s|\mu_0, p, \pi)$ for $\gamma < 1$ and $d_\pi^\gamma(s) \doteq \mathbb{E}[\sum_{t=0}^{T_{S_0}^\pi} \Pr(S_t = s|S_0, p, \pi)]$ for $\gamma = 1$.[3] Note d_π^γ remains well-defined for $\gamma = 1$ when $T_{\max} < \infty$. In order to optimize the policy performance $\nabla_\theta J_\gamma(\pi)$, one can follow (1) and, at time step t, update θ_t as

$$\theta_{t+1} \leftarrow \theta_t + \alpha \gamma_{\text{A}}^t q_\pi^{\gamma_{\text{C}}}(S_t, A_t) \nabla_\theta \log \pi(A_t|S_t), \tag{2}$$

where α is a learning rate. If we replace $q_\pi^{\gamma_{\text{C}}}$ with a learned value function, the update rule (2) becomes an actor-critic algorithm, where the actor refers to π and the critic refers to the learned approximation of $q_\pi^{\gamma_{\text{C}}}$. In practice, an estimate for $v_\pi^{\gamma_{\text{C}}}$ instead of $q_\pi^{\gamma_{\text{C}}}$ is usually learned. Theoretically, we should have $\gamma_{\text{A}} = \gamma_{\text{C}} = \gamma$. Practitioners, however, usually ignore the γ_{A}^t term in (2), and use $\gamma_{\text{C}} < \gamma_{\text{A}} = 1$. What this update truly optimizes remains an open problem (Nota & Thomas, 2020).

TRPO and PPO: To improve the stability of actor-critic algorithms, Schulman et al. (2015a) propose Trust Region Policy Optimization (TRPO), based on the performance improvement lemma:

[2] Following Schulman et al. (2015a), we consider $r : \mathcal{S} \to \mathbb{R}$ instead of $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ for simplicity.

[3] Sutton et al. (2000) do not explicitly define d_π^γ when $\gamma = 1$, which, however, can be easily deduced from Chapter 13.2 in Sutton & Barto (2018).

Lemma 1. *(Theorem 1 in Schulman et al. (2015a)) For $\gamma < 1$ and any two policies π and π',*

$$J_\gamma(\pi') \geq J_\gamma(\pi) + \Big(\sum_s d_\pi^\gamma(s) \sum_a \pi'(a|s) Adv_\pi^\gamma(s,a) \Big) - \frac{4 \max_{s,a} |Adv_\pi^\gamma(s,a)| \gamma \epsilon(\pi,\pi')}{(1-\gamma)^2},$$

where $Adv_\pi^\gamma(s,a) \doteq \mathbb{E}_{s' \sim p(\cdot|s,a)}[r(s) + \gamma v_\pi^\gamma(s') - v_\pi^\gamma(s)]$ is the advantage, $\epsilon(\pi,\pi') \doteq \max_s D_{KL}(\pi(\cdot|s)||\pi'(\cdot|s))$, and D_{KL} refers to the KL divergence.

To facilitate our empirical study, we first make a theoretical contribution by extending Lemma 1 to the undiscounted setting. We have the following lemma:

Lemma 2. *Assuming $T_{\max} < \infty$, for $\gamma = 1$ and any two policies π and π',*

$$J_\gamma(\pi') \geq J_\gamma(\pi) + \Big(\sum_s d_\pi^\gamma(s) \sum_a \pi'(a|s) Adv_\pi^\gamma(s,a) \Big) - 4 \max_{s,a} |Adv_\pi^\gamma(s,a)| T_{\max}^2 \epsilon(\pi,\pi').$$

The proof of Lemma 2 is provided in the appendix. A practical implementation of Lemmas 1 and 2 is to compute a new policy θ via gradient ascent on the clipped objective:

$$L(\theta) \doteq \sum_{t=0}^{\infty} \gamma_{\text{A}}^t \min \Big\{ \tfrac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{\text{old}}}(A_t|S_t)} \text{Adv}_{\pi_{\theta_{\text{old}}}}^{\gamma_{\text{C}}}(S_t,A_t), \text{clip}(\tfrac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{\text{old}}}(A_t|S_t)}) \text{Adv}_{\pi_{\theta_{\text{old}}}}^{\gamma_{\text{C}}}(S_t,A_t) \Big\}, \tag{3}$$

where S_t and A_t are sampled from $\pi_{\theta_{\text{old}}}$, and $\text{clip}(x) \doteq \max(\min(x, 1+\epsilon), 1-\epsilon)$ with ϵ a hyperparameter. Theoretically, we should have $\gamma_{\text{A}} = \gamma_{\text{C}}$, but practical algorithms like Proximal Policy Optimization (Schulman et al., 2017, PPO) usually use $\gamma_{\text{C}} < \gamma_{\text{A}} = 1$.

Policy Evaluation: We now introduce several policy evaluation techniques we use in our empirical study. Let $\hat{v}$ be our estimate of v_π^γ. At time step t, Temporal Difference learning (TD, Sutton (1988)) updates $\hat{v}$ as $\hat{v}(S_t) \leftarrow \hat{v}(S_t) + \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}) - \hat{v}(S_t))$. Instead of the infinite horizon discounted return G_t, De Asis et al. (2019) propose to consider the H-step return $G_t^H \doteq \sum_{i=1}^{H} R_{t+i}$. Correspondingly, the H-step value function is defined as $v_\pi^H(s) \doteq \mathbb{E}[G_t^H | S_t = s]$. We let $\hat{v}^H$ be our estimate of v_π^H. At time step t, De Asis et al. (2019) use the following update rule to learn $\hat{v}^H$:

$$\hat{v}^i(S_t) \leftarrow \hat{v}^i(S_t) + \alpha(R_{t+1} + \hat{v}^{i-1}(S_{t+1}) - \hat{v}^i(S_t)) \quad (i = 1, \dots H), \tag{4}$$

where $\hat{v}^0(s) \doteq 0$. In other words, to learn $\hat{v}^H$, we need to learn $\{\hat{v}^i\}_{i=1,\dots,H}$ simultaneously. De Asis et al. (2019) call (4) *Fixed Horizon Temporal Difference* learning (FHTD).

As G_t is a random variable, Bellemare et al. (2017) propose to learn its full distribution instead of its expectation only, yielding the Distributional Reinforcement Learning (RL) paradigm. They use a categorical distribution with 51 atoms uniformly distributed in $[-V_{\max}, V_{\max}]$ to approximate the distribution of G_t, where $V_{\max}$ is a hyperparameter. In this paper, we refer to the corresponding policy evaluation algorithm as *C51*.

Methodology: We consider MuJoCo robot simulation tasks from OpenAI gym (Brockman et al., 2016) as our benchmark. Given its popularity in understanding deep RL algorithms (Henderson et al., 2017; Ilyas et al., 2018; Engstrom et al., 2019; Andrychowicz et al., 2020) and designing new deep RL algorithms (Fujimoto et al., 2018; Haarnoja et al., 2018), we believe our empirical results are relevant to most practitioners.

We choose PPO, a simple yet effective and widely used algorithm, as the representative actor-critic algorithm for our empirical study. PPO is usually equipped with generalized advantage estimation (Schulman et al., 2015b, GAE), which has a tunable hyperparameter $\hat{\gamma}$. The roles of γ and $\hat{\gamma}$ are similar. To reduce its confounding effect, we do not use GAE in our experiments, *i.e.*, the advantage estimation for our actor is simply the TD error $R_{t+1} + \gamma_{\text{C}} \hat{v}(S_{t+1}) - \hat{v}(S_t)$. The PPO pseudocode we follow is provided in Alg. 1 in the appendix and we refer to it as the default PPO implementation.

We use the standard architecture and optimizer across all tasks, in particular, the actor and the critic do not share layers. We conduct a thorough learning rate search in `Ant` for each algorithmic configuration (*i.e.*, a curve in a figure) and then use the same learning rate for all other tasks. When using FHTD and C51, we also include H and $V_{\max}$ in the grid search. All details are provided in the appendix. We report the average episode return of the ten most recent episodes against the number of interactions with the environment. Curves are averages over ten independent runs with shaded regions indicating standard errors. All our implementations and our Docker environment are publicly available for future research at "removed for peer review".

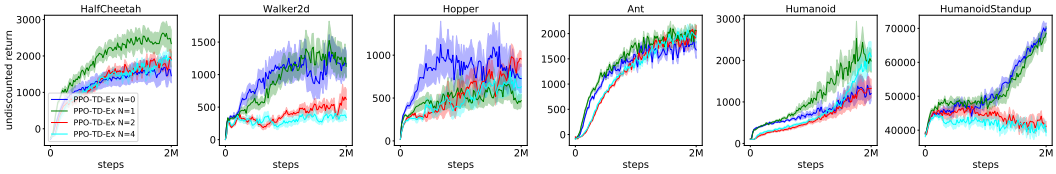


Figure 1: PPO-TD-Ex ($\gamma_{\text{C}} = 0.99$).

3 Optimizing the Undiscounted Objective (Scenario 1)

When our goal is to optimize the *undiscounted* objective $J_{\gamma=1}(\pi)$, one theoretically grounded option is to use $\gamma_{\text{A}} = \gamma_{\text{C}} = \gamma = 1$. By using $\gamma_{\text{A}} = 1$ and $\gamma_{\text{C}} < 1$, practitioners introduce *bias*. We first empirically confirm that introducing bias in this way indeed has empirical advantages. A simple first hypothesis is that $\gamma_{\text{C}} < 1$ leads to lower variance in Monte Carlo return bootstrapping targets than $\gamma_{\text{C}} = 1$, it thus optimizes a bias-variance trade-off. However, we further show that there are empirical advantages from $\gamma_{\text{C}} < 1$ that cannot uniquely be explained by this bias-variance trade-off, indicating that there are additional factors beyond variance. We then show empirical evidence identifying representation learning as an additional factor, leading to the *bias-variance-representation* trade-off from Hypothesis 1. All the experiments in this section use $\gamma_{\text{A}} = 1$.

Identifying bias-variance trade-off: To investigate the advantages of using $\gamma_{\text{C}} < 1$, we first test default PPO with $\gamma_{\text{C}} \in \{0.95, 0.97, 0.99, 0.995, 1\}$. We find that the best discount factor is always with $\gamma_{\text{C}} < 1$ and that $\gamma_{\text{C}} = 1$ usually leads to a performance drop (Figure 11 in the appendix). In default PPO, although the advantage is computed as the one-step TD error, the update target for updating the critic $\hat{v}(S_t)$ is almost always a Monte Carlo return. As there is no γ_{A}^t term in the actor update, we should theoretically use $\gamma_{\text{C}} = \gamma_{\text{A}} = 1$ when computing the Monte Carlo return, which usually leads to high variance. Consequently, a simple hypothesis for the empirical advantages of using $\gamma_{\text{C}} < 1$ is a bias-variance trade-off. We find, however, that there is more at play.

Beyond bias-variance trade-off: To reduce the effect of γ_{C} in controlling the variance, we benchmark PPO-TD (Algorithm 2 in the appendix). PPO-TD is the same as default PPO except that the critic is updated with one-step TD, *i.e.*, the update target for $\hat{v}(S_t)$ is now $R_{t+1} + \gamma_{\text{C}}\hat{v}(S_{t+1})$. Although Figure 12 in the appendix shows that PPO-TD ($\gamma_{\text{C}} = 1$) outperforms PPO ($\gamma_{\text{C}} = 1$) by a large margin, supporting the bias-variance trade-off hypothesis, Figure 13 suggests that for PPO-TD as well, $\gamma_{\text{C}} < 1$ is still preferable to $\gamma_{\text{C}} = 1$.

To further study this phenomenon, we benchmark PPO-TD-Ex (Algorithm 3 in the appendix), in which we provide N extra transitions to the critic by sampling multiple actions at any single state and using an averaged bootstrapping target. The update target for $\hat{v}(S_t)$ in PPO-TD-Ex is $\frac{1}{N}\sum_{i=0}^{N} R_{t+1}^i + \gamma_{\text{C}}\hat{v}(S_{t+1}^i)$. Here R_{t+1}^0 and S_{t+1}^0 refer to the original reward and successor state. To get R_{t+1}^i and S_{t+1}^i for $i \in \{1, \dots, N\}$, we first sample an action A_t^i from the sampling policy, then reset the environment to S_t, and finally execute A_t^i to get R_{t+1}^i and S_{t+1}^i. Importantly, we do not count those N extra transitions in the x-axis when plotting. The advantage for the actor update in PPO-TD-Ex is estimated with $R_{t+1}^0 + \hat{v}(S_{t+1}^0) - \hat{v}(S_t)$ regardless of γ_{C} to further control the influence of variance. Intuitively, if the quality of the estimate $\hat{v}(S_{t+1}^i)$ is good, providing more transitions this way should improve (or at least maintain) the overall performance. As shown by Figure 1, PPO-TD-Ex ($\gamma_{\text{C}} = 0.99$) roughly follows this intuition. However, surprisingly, providing extra data to PPO-TD-Ex ($\gamma_{\text{C}} = 1$) leads to a significant performance drop (Figure 2). This drop suggests that the larger variance from the randomness of S_{t+1} is not the only issue when using $\gamma_{\text{C}} = 1$ to train the critic. The quality of the estimate $\hat{v}$, at least in terms of making prediction on untrained states $\{S_{t+1}^i\}_{1,\dots,N}$, is lower when $\gamma_{\text{C}} = 1$ is used than $\gamma_{\text{C}} < 1$. In other words, the generalization of $\hat{v}$ is poor when $\gamma_{\text{C}} = 1$. The curves for PPO-TD-Ex ($\gamma_{\text{C}} = 0.995$) are a mixture of $\gamma_{\text{C}} = 0.99$ and $\gamma_{\text{C}} = 1$ and are provided in Figure 14.

In the undiscounted setting, we should theoretically have $R_{t+1} + \hat{v}(S_{t+1})$ as the update target for the critic. When $\gamma_{\text{C}} < 1$ is used instead, the update target becomes $R_{t+1} + \gamma_{\text{C}}\hat{v}(S_{t+1})$ and the variance resulting from the randomness of S_{t+1} becomes less pronounced. So here, γ_{C} trades off bias with variance, similar to that in Monte Carlo return bootstrapping targets in default PPO. We

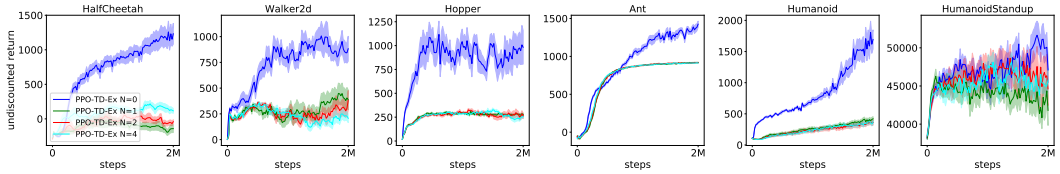


Figure 2: PPO-TD-Ex ($\gamma_{\text{C}} = 1$).

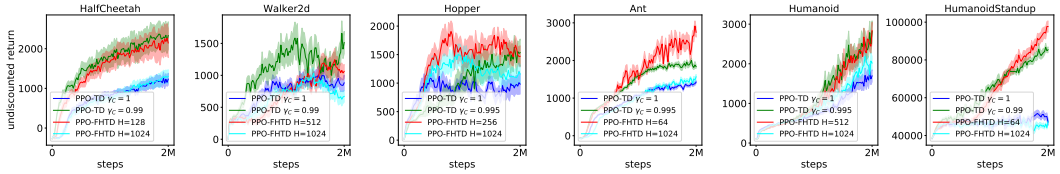


Figure 3: PPO-FHTD with the first parameterization. The best H and γ_{C} are used for each game.

refer to this effect of γ_{C} as *variance control*. However, γ_{C} can also affect the difficulty of learning a good estimate $\hat{v}$ for $v_\pi^{\gamma_{\text{C}}}$; we refer to this effect of γ_{C} as *learnability control* (Lehnert et al., 2018; Laroche & van Seijen, 2018; Romoff et al., 2019). Inspired by the poor generalization of $\hat{v}$ when $\gamma_{\text{C}} = 1$, we investigate learnability control mainly from the representation learning perspective. By representation learning, we refer to learning the bottom layers (backbone) of a neural network. The last layer of the neural network is then interpreted as a linear function approximator whose features are the output of the backbone. This interpretation of representation learning is widely used in the RL community, see *e.g.* Jaderberg et al. (2016); Chung et al. (2018); Veeriah et al. (2019).

Identifying bias-representation trade-off: To separate variance control and learnability control, ideally we should investigate the update target $R_{t+1}+\gamma_{\text{C},1}\hat{v}(S_{t+1})$, where $\hat{v}$ is trained to approximate $v_\pi^{\gamma_{\text{C},2}}$ and $\gamma_{\text{C},2} < \gamma_{\text{C},1} = 1$. Learning an estimate $\hat{v}$ for $v_\pi^{\gamma_{\text{C},2}}$, however, implies to use the update target $R_{t+1}+\gamma_{\text{C},2}\hat{v}(S_{t+1})$: the two effects of $\gamma_{\text{C},2}$ then get mixed again. To solve this dilemma, we consider the update target $R_{t+1} + \hat{v}^{H-1}(S_{t+1})$, where $\hat{v}^{H-1}(S_{t+1})$ is trained to approximate v_π^{H-1}, *i.e.*, we use FHTD to train the critic in PPO, which we refer to as PPO-FHTD (Algorithm 4 in the appendix). PPO-FHTD implements $\gamma_{\text{C},1} = 1$ directly, and manipulating H changes the horizon of the policy evaluation problem, which is also one of the effects of manipulating $\gamma_{\text{C},2}$.

We test two parameterizations for PPO-FHTD to investigate representation learning. In the first parameterization, to learn v_π^H, we parameterize $\{v_\pi^i\}_{i=1,\dots,H}$ as H different heads over the same representation layer (backbone). In the second parameterization, we always learn $\{v_\pi^i\}_{i=1,\dots,1024}$ as 1024 different heads over the same representation layer, whatever H we are interested in. To approximate v_π^H, we then simply use the output of the H-th head. Figure 10 in the appendix further illustrates the difference between the two parameterizations.

Figure 3 shows that by tuning H for FHTD, PPO-FHTD with the first parameterization matches or exceeds the performance of PPO-TD ($\gamma_{\text{C}} < 1$) in most tasks, and that the best H is always smaller than 1024. Theoretically, as long as we use an $H \geq T_{\max} = 1000$, we always have $v_\pi^H(s) \equiv v_\pi^{\gamma=1}(s)$. Figure 3 shows that the performance of PPO-FHTD ($H = 1024$) is very close to PPO-TD ($\gamma_{\text{C}} = 1$), indicating that learning $\{v_\pi^i\}_{i=1,\dots,1024}$ is not an additional overhead for the network in terms of learning $v_\pi^{H=1024}$, *i.e.*, increasing H does not pose additional challenges in terms of network capacity. However, Figure 4 suggests that for the second parameterization, $H = 1024$ is almost always among the best choices of H. Comparing Figures 3 and 4, we conclude that in the tested domains, learning v_π^H with different H requires different representations. This suggests that we can interpret the results in Figure 3 as a *bias-representation trade-off*. Using a larger H is less biased but representation learning *may* become harder due to the longer policy evaluation horizon. Consequently, an intermediate H achieves the best performance in Figure 3. As reducing H cannot bring in advantages in representation learning under the second parameterization, the less biased H, *i.e.*, the larger H, usually performs better in Figure 4. Overall, γ_{C} optimizes a *bias-representation* trade-off by changing the policy evaluation horizon H.

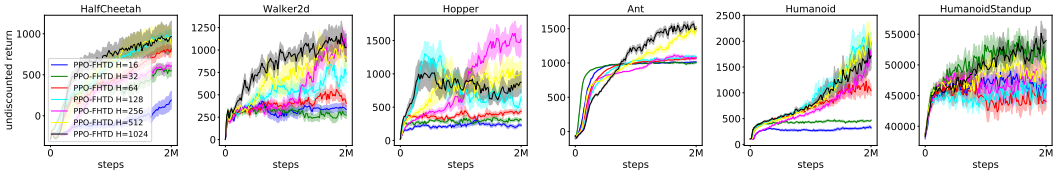


Figure 4: PPO-FHTD with the second parameterization.

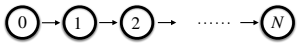


Figure 5: A simple MRP.

We further conjecture that representation learning may be harder for a longer horizon because good representations can become rarer. We provide a simulated example to support this. Consider policy evaluation on the simple Markov Reward Process (MRP) from Figure 5. We assume the reward for each transition is fixed and is randomly generated in $[0, 1]$. Let $x_s \in \mathbb{R}^K$ be the feature vector for a state s; we set its i-th component as $x_s[i] \doteq \tanh(\xi)$, where ξ is a random variable uniformly distributed in $[-2, -2]$. We chose this feature setup as we use $\tanh$ as the activation function in our PPO. We use $X \in \mathbb{R}^{N \times K}$ to denote the feature matrix. To create state aliasing (McCallum, 1997), which is common under function approximation, we first randomly split the N states into $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $|\mathcal{S}_1| = \alpha N$ and $|\mathcal{S}_2| = (1 - \alpha)N$, where α is the proportion of states to be aliased. Then for every $s \in \mathcal{S}_1$, we randomly select an $\hat{s} \in \mathcal{S}_2$ and set $x_s \leftarrow x_{\hat{s}}$. Finally, we add Gaussian noise $\mathcal{N}(0, 0.1^2)$ to each element of X. We use $N = 100$ and $K = 30$ in our simulation and report the normalized representation error (NRE) as a function of γ. For a feature matrix X, the NRE is computed analytically as $\text{NRE}(\gamma) \doteq \frac{\min_w ||Xw - v_\gamma||_2}{||v_\gamma||_2}$, where v_γ is the analytically computed true value function of the MRP. We report the results in Figure 6, where each data point is averaged over 10^4 randomly generated feature matrices (X) and reward functions. In this MRP, the average representation error becomes larger as γ increases, which suggests that learning a good representation under a large γ and state aliasing may be harder than with a smaller γ. We report the unnormalized representation error in Figure 16 in the appendix, where the trend is much clearer.

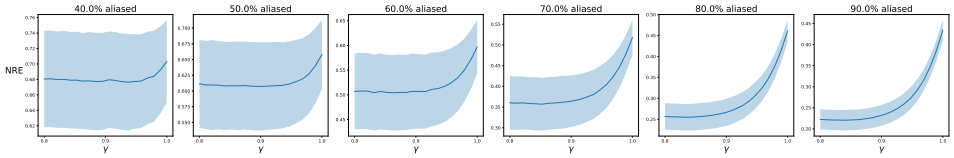


Figure 6: Normalized representation error as a function of the discount factor. Shaded regions indicate one standard derivation. [4]

Hypothesis 1 and the previous empirical study suggest that representation learning may be the main bottleneck of PPO-TD ($\gamma_{\text{C}} = 1$). To further support this, we benchmark PPO-C51 ($\gamma_{\text{C}} = 1$) (Algorithm 5 in the appendix), where the critic of PPO is trained with C51. C51 is usually considered to improve representation learning by implicitly providing auxiliary tasks (Bellemare et al., 2017; Munos, 2018; Petroski Such et al., 2019). Figure 7 shows that training the critic with C51 indeed leads to a performance improvement and PPO-C51 ($\gamma_{\text{C}} = 1$) sometimes outperforms PPO-TD ($\gamma_{\text{C}} < 1$) by a large margin. Figure 8 further shows that when $V_{\max}$ is optimized for PPO-C51, the benefit for using $\gamma_{\text{C}} < 1$ in PPO-C51 is less pronounced than that in PPO-TD, indicating the role of $\gamma_{\text{C}} < 1$ and distributional learning may overlap. Figures 3, 4, & 6, suggest that the overlapping is representation learning.

4 Optimizing the Discounted Objective (Scenario 2)

When our goal is to optimize the *discounted* objective $J_{\gamma<1}(\pi)$, theoretically we should have the γ_{A}^t term in the actor update and use $\gamma_{\text{C}} < 1$. Practitioners, however, usually ignore this γ_{A}^t (*i.e.*, set $\gamma_{\text{A}} = 1$), introducing *bias*. Figure 15 in the appendix shows that even if we use the *discounted* return

[4] The trend that NRE decreases as α increases is merely an artifact from how we generate v_γ.

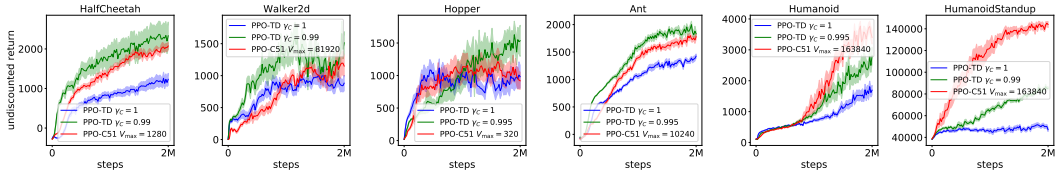


Figure 7: For PPO-C51, we set $\gamma_{\text{C}} = 1$.

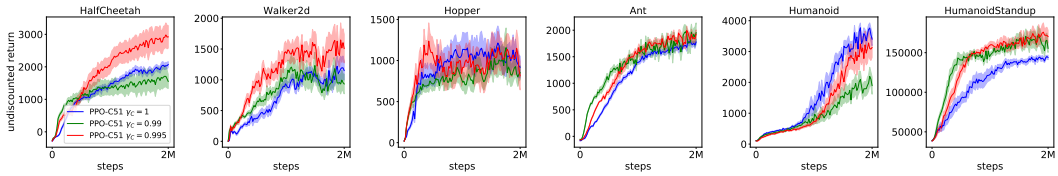


Figure 8: For each game, $V_{\max}$ is the same as the $V_{\max}$ in Figure 7.

as the performance metric, the biased implementation of PPO still outperforms the theoretically grounded implementation DisPPO in the domains we tested. Here PPO refers to the default PPO implementation where $\gamma_{\text{A}} = 1, \gamma_{\text{C}} = \gamma < 1$, and DisPPO (Alg. 6 in the appendix) adds the missing γ_{A}^t term in PPO by using $\gamma_{\text{A}} = \gamma_{\text{C}} = \gamma < 1$. We propose to interpret the empirical advantages of PPO over DisPPO with Hypothesis 2. For all experiments in this section, we use $\gamma_{\text{C}} = \gamma < 1$.

An auxiliary task perspective: The biased policy update implementation of (2) ignoring γ_{A}^t can be decomposed into two parts as $\Delta_t = \gamma^t \Delta_t + (1 - \gamma^t)\Delta_t$, where $\Delta_t \doteq q_\pi^{\gamma_{\text{C}}}(S_t, A_t)\nabla_\theta \log \pi(A_t|S_t)$. We propose to interpret the *difference term* between the biased implementation (Δ_t) and the theoretically grounded implementation ($\gamma^t \Delta_t$), *i.e.*, the $(1-\gamma^t) q_\pi^{\gamma_{\text{C}}}(S_t, A_t)\nabla_\theta \log \pi(A_t|S_t)$ term, as the gradient of an auxiliary objective with a dynamic weighting $1 - \gamma^t$. Let $J_{s,\mu}(\pi) \doteq \sum_a \pi(a|s) q_\mu^\gamma(s, a)$; we have $\nabla_\theta J_{s,\mu}(\pi)|_{\mu=\pi} = \mathbb{E}_{a\sim\pi(\cdot|s)}[q_\pi^\gamma(s, a)\nabla_\theta \log \pi(a|s)]$. This objective changes every time step (through μ). Inspired by the decomposition, we augment PPO with this auxiliary task, yielding AuxPPO (Algorithm 7 and Figure 10 in the appendix). In AuxPPO, we have two policies π and π' parameterized by θ and θ' respectively. The two policies are two heads over the same neural network backbone, where π is used for interaction with the environment and π' is the policy for the auxiliary task. AuxPPO optimizes θ and θ' simultaneously by considering the following joint loss

$$L(\theta,\theta') \doteq \sum_{t=0}^{\infty} \gamma^t \min\left\{\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{\text{old}}}(A_t|S_t)}\text{Adv}_{\pi_{\theta_{\text{old}}}}^{\gamma_{\text{C}}}(S_t, A_t), \text{clip}(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{\text{old}}}(A_t|S_t)})\text{Adv}_{\pi_{\theta_{\text{old}}}}^{\gamma_{\text{C}}}(S_t, A_t)\right\} +$$
$$\sum_{t=0}^{\infty}(1 - \gamma^t) \min\left\{\frac{\pi_{\theta'}(A_t|S_t)}{\pi_{\theta_{\text{old}}}(A_t|S_t)}\text{Adv}_{\pi_{\theta_{\text{old}}}}^{\gamma_{\text{C}}}(S_t, A_t), \text{clip}(\frac{\pi_{\theta'}(A_t|S_t)}{\pi_{\theta_{\text{old}}}(A_t|S_t)})\text{Adv}_{\pi_{\theta_{\text{old}}}}^{\gamma_{\text{C}}}(S_t, A_t)\right\},$$

where S_t and A_t are obtained by executing θ_{old}. We additionally synchronize θ' with θ periodically to avoid an off-policy learning issue.

Flipped rewards: Besides AuxPPO, we also design novel environments with flipped rewards to investigate Hypothesis 2. Recall we include the time step in the state, this allows us to simply create a new environment by defining a new reward function $r'(s, t) \doteq r(s)\mathbb{I}_{t\leq t_0} - r(s)\mathbb{I}_{t>t_0}$, where $\mathbb{I}$ is the indicator function. During an episode, within the first t_0 steps, this new environment is the same as the original one. After t_0 steps, the sign of the reward is flipped. We select t_0 such that γ^{t_0} is sufficiently small, *e.g.*, we define $t_0 \doteq \min_t\{\gamma^t < 0.05\}$.

Results: Figure 9 shows that in the original `Ant` environment, PPO is usually better than DisPPO for all tested discount factors, even if the performance metric is the *discounted* episodic return. Importantly, by using the difference term as an auxiliary task, AuxPPO is able to improve upon DisPPO. The improvement is clear for $\gamma \in \{0.995, 0.99, 0.97, 0.95\}$, where AuxPPO is able to roughly match the performance of PPO at the end of training. For $\gamma \in \{0.93, 0.9\}$, the improvement is not clear and we conjecture that this is because the learning of the π-head (the control head) in AuxPPO is much slower than the learning of π in PPO due to the γ_{C}^t term. In `Ant` with flipped rewards, DisPPO, however, always outperforms PPO by a large margin. The transitions after t_0 steps are not directly relevant when the performance metric is the discounted return. However, learning on those transitions can still improve representation learning provided that those transitions are similar

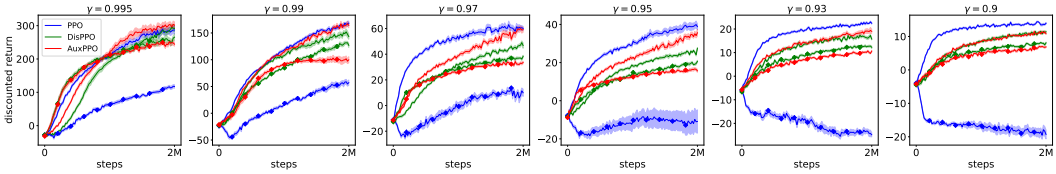


Figure 9: Curves without any marker are obtained in the original `Ant` environment. Diamond-marked curves are obtained in `Ant` with r'. [5]

to the earlier transitions, which is the case in the original `Ant` environment. PPO and AuxPPO, therefore, outperform DisPPO. However, when those transitions are much different from the earlier transitions, which is the case in `Ant` with flipped rewards, learning to control on them directly becomes distracting. PPO, therefore, is outperformed by DisPPO. Different from PPO, AuxPPO does not learn to control on later transitions. Provided that the network has enough capacity and the $1-\gamma_{\mathrm{C}}^t$ part does not dominate $L(\theta, \theta')$ much, the control head π_θ in AuxPPO will not be affected much by the unrelated transitions. The performance of AuxPPO is, therefore, similar to DisPPO. To summarize, Figure 9 suggests that using $\gamma_{\mathrm{A}} = 1$ is simply an *inductive bias* that *all transitions are equally important*. When this inductive bias is helpful for learning, $\gamma_{\mathrm{A}} = 1$ implicitly implements auxiliary tasks thus improving representation learning and the overall performance. When this inductive bias is detrimental, however, $\gamma_{\mathrm{A}} = 1$ can lead to significant performance drops.

5 Related Work

The mismatch in actor-critic algorithm implementations has been previously studied. Thomas (2014) focuses on the natural policy gradient setting and shows that the biased implementation ignoring γ_{A}^t can be interpreted as the gradient of the average reward objective under a strong assumption that the state distribution is independent of the policy. Nota & Thomas (2020) prove that without this strong assumption, the biased implementation is *not* the gradient of any *stationary* objective. This does not contradict our auxiliary task perspective as our objective $J_{s,\mu}(\pi)$ changes at every time step. Nota & Thomas (2020) further provide a counterexample showing that following the biased gradient can lead to a policy of poor performance w.r.t. both discounted and undiscounted objectives. Both Thomas (2014) and Nota & Thomas (2020), however, focus on *theoretical disadvantages* of the biased gradient and regard ignoring γ_{A}^t as the source of the bias. We instead regard the introduction of $\gamma_{\mathrm{C}} < 1$ in the critic as the source of the bias in the undiscounted setting and investigate its *empirical advantages*, which are more relevant to practitioners. Moreover, our representation learning perspective for investigating this mismatch is to our knowledge novel.

Although we propose the *bias-variance-representation* trade-off, we do not claim that is all that γ affects. The discount factor also has many other effects (*e.g.*, Sutton (1995); Jiang et al. (2016); Laroche et al. (2017); Laroche & van Seijen (2018); Lehnert et al. (2018); Fedus et al. (2019); Van Seijen et al. (2019); Amit et al. (2020)), which we leave for future work. Besides facilitating representation learning, distributional RL also has other effects (see *e.g.*, Imani & White (2018); Lyle et al. (2019)), which we leave for future work as well.

6 Conclusion

In this paper, we investigate the longstanding mismatch between theorists and practitioners in actor-critic algorithms from a representation learning perspective. Although the theoretical understanding of policy gradient algorithms have recently been significantly advanced (Agarwal et al., 2019; Wu et al., 2020), this mismatch has drawn little attention. We hope our empirical study can help practitioners understand actor-critic algorithms better and therefore design more efficient actor-critic algorithms in the setting of deep RL, where representation learning emerges as a major consideration. We hope our empirical study can draw more attention to the mismatch, which could enable the community to finally close this longstanding gap.

[5] The other five environments are not well-suited for our empirical study. See Section B.1 for more details.

REFERENCES

Joshua Achiam. Spinning up in deep reinforcement learning. 2018.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. *arXiv preprint arXiv:1908.00261*, 2019.

Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. *arXiv preprint arXiv:2007.02040*, 2020.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific Belmont, MA, 1996.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Itai Caspi, Gal Leibovich, Gal Novik, and Shadi Endrawis. Reinforcement learning coach, December 2017. URL `https://doi.org/10.5281/zenodo.1134899`.

Wesley Chung, Somjit Nath, Ajin Joseph, and Martha White. Two-timescale networks for nonlinear value function approximation. In *International Conference on Learning Representations*, 2018.

Kristopher De Asis, Alan Chan, Silviu Pitis, Richard S Sutton, and Daniel Graves. Fixed-horizon temporal difference methods for stable reinforcement learning. *arXiv preprint arXiv:1909.03906*, 2019.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

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*, 2019.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. *arXiv preprint arXiv:1811.02553*, 2018.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. *arXiv preprint arXiv:1806.04613*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Nan Jiang, Satinder P Singh, and Ambuj Tewari. On structural properties of mdps that bound loss due to shallow planning. In *IJCAI*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vijay R Konda. *Actor-critic algorithms*. PhD thesis, Massachusetts Institute of Technology, 2002.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

Romain Laroche and Harm van Seijen. In reinforcement learning, all objective functions are not equal. 2018.

Romain Laroche, Mehdi Fatemi, Joshua Romoff, and Harm van Seijen. Multi-advisor reinforcement learning. *arXiv preprint arXiv:1704.00756*, 2017.

Lucas Lehnert, Romain Laroche, and Harm van Seijen. On value function representation of long horizon problems. In *AAAI Conference on Artificial Intelligence*, 2018.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, 2018.

Clare Lyle, Marc G Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

R McCallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, 1997.

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 *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Rémi Munos. Distributional reinforcement learning. Invited talk at European Workshop on Reinforcement Learning `https://ewrl.files.wordpress.com/2018/10/distributional_rl.pdf`, 2018.

Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 2020.

OpenAI. Openai five. `https://openai.com/five/`, 2018.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*, 2018.

Felipe Petroski Such, Vashisht Madhavan, Rosanne Liu, Rui Wang, Pablo Samuel Castro, Yulun Li, Jiale Zhi, Ludwig Schubert, Marc G. Bellemare, Jeff Clune, and et al. An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Aug 2019. doi: 10.24963/ijcai.2019/452. URL `http://dx.doi.org/10.24963/ijcai.2019/452`.

Joshua Romoff, Peter Henderson, Ahmed Touati, Emma Brunskill, Joelle Pineau, and Yann Ollivier. Separating value functions across time-scales. *arXiv preprint arXiv:1902.01883*, 2019.

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*, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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*, 2016.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

Richard S Sutton. Td models: Modeling the world at a mixture of time scales. In *Machine Learning Proceedings 1995*. Elsevier, 1995.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction (2nd Edition)*. MIT press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

Philip Thomas. Bias in natural actor-critic algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Harm Van Seijen, Mehdi Fatemi, and Arash Tavakoli. Using a logarithmic mapping to enable lower discount factors in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L Lewis, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. In *Advances in Neural Information Processing Systems*, 2019.

Arthur F Veinott. Discrete dynamic programming with sensitive discount optimality criteria. *The Annals of Mathematical Statistics*, 1969.

Martin L Weitzman. Gamma discounting. *American Economic Review*, 2001.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Yue Wu, Weitong Zhang, Pan Xu, and Quanquan Gu. A finite time analysis of two time-scale actor critic methods. *arXiv preprint arXiv:2005.01350*, 2020.

Shangtong Zhang. Modularized implementation of deep rl algorithms in pytorch. `https://github.com/ShangtongZhang/DeepRL`, 2018.

A PROOF OF LEMMA 2

Proof. The proof is based on Appendix B in Schulman et al. (2015a), where perturbation theory is used to prove the performance improvement bound (Lemma 1). To simplify notation, we use a vector and a function interchangeably, *i.e.*, we also use r and μ_0 to denote the reward vector and the initial distribution vector. $J(\pi)$ and $d_\pi(s)$ are shorthand for $J_\gamma(\pi)$ and $d_\pi^\gamma(s)$ with $\gamma = 1$. All vectors are *column* vectors.

Let $\mathcal{S}^+$ be the set of states excluding s^∞, *i.e.*, $\mathcal{S}^+ \doteq \mathcal{S}/\{s^\infty\}$, we define $P_\pi \in \mathbb{R}^{|\mathcal{S}^+|\times|\mathcal{S}^+|}$ such that $P_\pi(s,s') \doteq \sum_a \pi(a|s)p(s'|s,a)$. Let $G \doteq \sum_{t=0}^\infty P_\pi^t$. According to standard Markov chain theories, $G(s,s')$ is the expected number of times that s' is visited before s^∞ is hit given $S_0 = s$. $T_{\max} < \infty$ implies that G is well-defined and we have $G = (I - P_\pi)^{-1}$. Moreover, $T_{\max} < \infty$ also implies $\forall s, \sum_{s'} G(s,s') \le T_{\max}$, *i.e.*, $||G||_\infty \le T_{\max}$. We have $J(\pi) = \mu_0^\top G r$.

Let $G' \doteq (I - P_{\pi'})^{-1}$, we have

$$J(\pi') - J(\pi) = \mu_0^\top (G' - G) r.$$

Let $\Delta \doteq P_{\pi'} - P_\pi$, we have

$$G'^{-1} - G^{-1} = -\Delta,$$

Left multiply by G' and right multiply by G,

$$\begin{aligned} G - G' &= -G'\Delta G, \\ G' &= G + G'\Delta G \quad \text{(Expanding } G' \text{ in RHS recursively)} \\ &= G + G\Delta G + G'\Delta G\Delta G. \end{aligned}$$

So we have

$$J(\pi') - J(\pi) = \mu_0^\top G\Delta G r + \mu_0^\top G'\Delta G\Delta G r.$$

It is easy to see $\mu_0^\top G = d_\pi^\top$ and $Gr = v_\pi$. So

$$\begin{aligned} \mu_0^\top G\Delta G r &= d_\pi^\top \Delta v_\pi \\ &= \sum_s d_\pi(s) \sum_{s'} \Big(\sum_a \pi'(a|s)p(s'|s,a) - \sum_a \pi(a|s)p(s'|s,a) \Big) v_\pi(s') \\ &= \sum_s d_\pi(s) \sum_a (\pi'(a|s) - \pi(a|s)) \sum_{s'} p(s'|s,a) v_\pi(s') \\ &= \sum_s d_\pi(s) \sum_a (\pi'(a|s) - \pi(a|s)) \Big(r(s) + \sum_{s'} p(s'|s,a) v_\pi(s') - v_\pi(s) \Big) \\ &\qquad (\textstyle\sum_a (\pi'(a|s) - \pi(a|s)) f(s) = 0 \text{ holds for any } f \text{ that dependes only on } s) \\ &= \sum_s d_\pi(s) \sum_a \pi'(a|s) \mathrm{Adv}_\pi(s,a). \\ &\qquad (\textstyle\sum_a \pi(a|s)\mathrm{Adv}_\pi(s,a) = 0 \text{ by Bellman equation}) \end{aligned}$$

We now bound $\mu_0^\top G'\Delta G\Delta G r$. First,

$$\begin{aligned} |(\Delta G r)(s)| &= |\sum_{s'} \Big(\sum_a \pi'(a|s) - \pi(a|s) \Big) p(s'|s,a) v_\pi(s')| \\ &= |\sum_a \Big(\pi'(a|s) - \pi(a|s) \Big) \Big(r(s) + \sum_{s'} p(s'|s,a) v_\pi(s') - v_\pi(s) \Big)| \\ &= |\sum_a \Big(\pi'(a|s) - \pi(a|s) \Big) \mathrm{Adv}_\pi(s,a)| \\ &\le 2 \max_s \mathrm{D}_{TV}(\pi'(\cdot|s), \pi(\cdot|s)) \max_{s,a} |\mathrm{Adv}_\pi(s,a)|, \end{aligned}$$

where D_{TV} is the total variation distance. So

$$||\Delta G r||_\infty \leq 2 \max_s \mathrm{D}_{TV}(\pi'(\cdot|s), \pi(\cdot|s)) \max_{s,a} |\mathrm{Adv}_\pi(s,a)|.$$

Moreover, for any vector x,

$$|(\Delta x)(s)| \leq 2 \max_s \mathrm{D}_{TV}(\pi'(\cdot|s), \pi(\cdot|s)) ||x||_\infty,$$
$$||\Delta x||_\infty \leq 2 \max_s \mathrm{D}_{TV}(\pi'(\cdot|s), \pi(\cdot|s)) ||x||_\infty.$$

So

$$\begin{aligned}
||\Delta||_\infty &\leq 2 \max_s \mathrm{D}_{TV}(\pi'(\cdot|s), \pi(\cdot|s)), \\
|\mu_0^\top G' \Delta G \Delta G r| &\leq ||\mu_0^\top||_1 ||G'||_\infty ||\Delta||_\infty ||G||_\infty ||\Delta G r||_\infty \\
&\leq 4 T_{\max}^2 \max_s \mathrm{D}_{TV}^2(\pi'(\cdot|s), \pi(\cdot|s)) \max_{s,a} |\mathrm{Adv}_\pi(s,a)| \\
&\leq 4 T_{\max}^2 \max_s \mathrm{D}_{KL}(\pi(\cdot|s) || \pi'(\cdot|s)) \max_{s,a} |\mathrm{Adv}_\pi(s,a)|,
\end{aligned}$$

which completes the proof. □

Note this perturbation-based proof of Lemma 2 holds only for $r : \mathcal{S} \to \mathbb{R}$. For $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, we can turn to the coupling-based proof as Schulman et al. (2015a), which, however, complicates the presentation and deviates from the main purpose of this paper. We, therefore, leave it for future work.

B Experiment Details

B.1 Methodology

We use `HalfCheetah`, `Walker`, `Hopper`, `Ant`, `Humanoid`, and `HumanoidStandup` as our benchmarks. We exclude other tasks as we find PPO plateaus quickly there. The tasks we consider have a hard time limit of 1000. Following Pardo et al. (2018), we add time step information into the state, *i.e.*, there is an additional scalar $t/1000$ in the observation vector. Following Achiam (2018), we estimate the KL divergence between the current policy θ and the sampling policy θ_{old} when optimizing the loss (3). When the estimated KL divergence is greater than a threshold, we stop updating the actor and update only the critic with current data. We use Adam (Kingma & Ba, 2014) as the optimizer and perform grid search for the initial learning rates of Adam optimizers. Let α_A and $\alpha_C \doteq \beta\alpha_A$ be the learning rates for the actor and critic respectively. For each algorithmic configuration (*i.e.*, a curve in a figure), we tune $\alpha_A \in \{0.125, 0.25, 0.5, 1, 2\} \times 3 \cdot 10^{-4}$ and $\beta \in \{1, 3\}$ with grid search in `Ant` with 3 independent runs maximizing the average return of the last 100 training episodes. In particular, $\alpha_A = 3 \cdot 10^{-4}$ and $\beta = 3$ is roughly the default learning rates for the PPO implementation in Achiam (2018). We then run this algorithmic configuration with the best α_A and α_C in all tasks. Overall, we find after removing GAE, smaller learning rates are preferred. When we use FHTD, we additionally consider $H \in \{16, 32, 64, 128, 256, 512, 1024\}$ in the grid search. When we use C51, we additionally consider $V_{\max} \in \{20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240, 81920, 163840, 327680\}$ in the grid search.

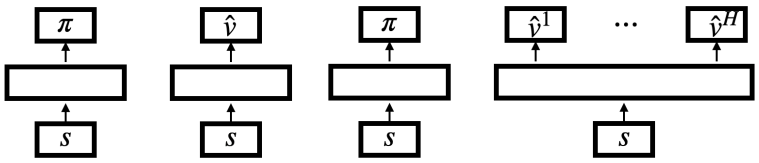
In the discounted setting, we consider only `Ant` and its variant. For `Walker2d`, `Hopper`, and `Humanoid`, we find the average episode length of all algorithms are smaller than t_0, *i.e.*, the flipped reward rarely takes effects. For `HumanoidStandup` and `HalfCheetah`, the episode length is fixed to 1000. Consequently, the $1 - \gamma_{\text{C}}^t$ part in $L(\theta, \theta')$ dominates the representation learning (the training of the backbone) during training, making the environments ill-posed for investigating AuxPPO. To summarize, other five environments are not well-suited for the purpose of our empirical study.

B.2 Algorithm Details

The pseudocode of all implemented algorithms are provide in Algorithms 1 - 7 with their architectures illustrated in Figure 10. For hyperparameters that are not included in the grid search, we use

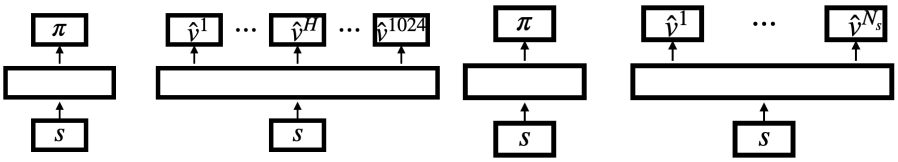
the same value as Dhariwal et al. (2017); Achiam (2018). In particular, for the rollout length, we set $K = 2048$. For the optimization epochs, we set $K_{opt} = 320$. For the minibatch size, we set $B = 64$. For the maximum KL divergence, we set $KL_{target} = 0.01$. We clip $\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ into $[-0.2, 0.2]$. We use $N_s = 51$ supports for PPO-C51.

We use two-hidden-layer neural networks for function approximation. Each hidden layer has 64 hidden units and a $\tanh$ activation function. The output layer of the actor network has a $\tanh$ activation function and is interpreted as the mean of an isotropic Gaussian distribution, whose standard derivation is a global state-independent variable as suggested by Schulman et al. (2015a).



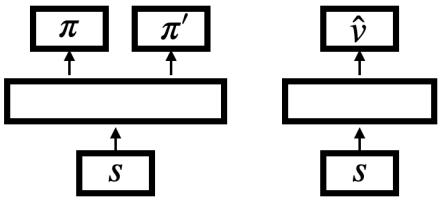
(a) Architecture of PPO, PPO-TD, PPO-TD-Ex, DisPPO

(b) The first parameterization of PPO-FHTD



(c) The second parameterization of PPO-FHTD

(d) Architecture of PPO-C51



(e) Architecture of AuxPPO

Figure 10: Architectures of the algorithms

Algorithm 1: PPO

Input:
θ, ψ: parameters of $\pi, \hat{v}$
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold

$S_0 \sim \mu_0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta$
  for $i = 0, \ldots, K-1$ **do**
    $A_i \sim \pi_{\theta_{old}}(\cdot|S_i)$
    Execute A_i, get R_{i+1}, S_{i+1}
    if S_{i+1} *is a terminal state* **then**
      $m_i \leftarrow 0, S_{i+1} \sim \mu_0$
    else
      $m_i \leftarrow 1$
    end
  end
  $G_K \leftarrow \hat{v}(S_K)$
  for $i = K-1, \ldots, 0$ **do**
    $G_i \leftarrow R_{i+1} + \gamma_{\mathrm{C}} m_i G_{i+1}$
    $\mathrm{Adv}_i \leftarrow R_{i+1} + \gamma_{\mathrm{C}} m_i \hat{v}_\psi(S_{i+1}) - \hat{v}_\psi(S_i)$
    Store $(S_i, A_i, G_i, \mathrm{Adv}_i)$ in M
  end
  Normalize Adv_i in M as $\mathrm{Adv}_i \leftarrow \frac{\mathrm{Adv}_i - \mathrm{mean}(\{\mathrm{Adv}_i\})}{\mathrm{std}(\{\mathrm{Adv}_i\})}$
  for $o = 1, \ldots, K_{opt}$ **do**
    Sample a minibatch $\{(S_i, A_i, G_i, \mathrm{Adv}_i)\}_{i=1,\ldots,B}$ from M
    $L(\psi) \leftarrow \frac{1}{2B} \sum_{i=1}^{B} (\hat{v}_\psi(S_i) - G_i)^2$ `/* No gradient through` G_i `*/`
    $L(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \min\{\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \mathrm{Adv}_i, \mathrm{clip}(\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \mathrm{Adv}_i\}$
    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i|S_i) - \log \pi_\theta(A_i|S_i) < KL_{target}$ **then**
      Perform one gradient update to θ maximizing $L(\theta)$ with Adam
    end
  end
end

Algorithm 2: PPO-TD

Input:
θ, ψ: parameters of $\pi, \hat{v}$
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold

$S_0 \sim \mu_0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta$
  for $i = 0, \ldots, K-1$ **do**
    $A_i \sim \pi_{\theta_{old}}(\cdot|S_i)$
    Execute A_i, get R_{i+1}, S_{i+1}
    if S_{i+1} *is a terminal state* **then**
      $m_i \leftarrow 0, S_{i+1} \sim \mu_0$
    else
      $m_i \leftarrow 1$
    end
  end
  for $i = K-1, \ldots, 0$ **do**
    $\text{Adv}_i \leftarrow R_{i+1} + \gamma_{\text{C}} m_i \hat{v}_\psi(S_{i+1}) - \hat{v}_\psi(S_i)$
    $S'_i \leftarrow S_{i+1}, r_i \leftarrow R_{i+1}$
    Store $(S_i, A_i, m_i, r_i, S'_i, \text{Adv}_i)$ in M
  end
  Normalize Adv_i in M as $\text{Adv}_i \leftarrow \frac{\text{Adv}_i - \text{mean}(\{\text{Adv}_i\})}{\text{std}(\{\text{Adv}_i\})}$
  for $o = 1, \ldots, K_{opt}$ **do**
    Sample a minibatch $\{(S_i, A_i, m_i, r_i, S'_i, \text{Adv}_i)\}_{i=1,\ldots,B}$ from M
    $y_i \leftarrow r_i + \gamma_{\text{C}} m_i \hat{v}_\psi(S'_i)$
    $L(\psi) \leftarrow \frac{1}{2B} \sum_{i=1}^{B} (\hat{v}_\psi(S_i) - y_i)^2$ `/* No gradient through` y_i `*/`
    $L(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \min\{\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \text{Adv}_i, \text{clip}(\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \text{Adv}_i\}$
    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i|S_i) - \log \pi_\theta(A_i|S_i) < KL_{target}$ **then**
      Perform one gradient update to θ maximizing $L(\theta)$ with Adam
    end
  end
end

Algorithm 3: PPO-TD-Ex

Input:
θ, ψ: parameters of $\pi, \hat{v}$
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold
N: number of extra transitions
p, r: transition kernel and reward function of the oracle

$S_0 \sim \mu_0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta$
  for $i = 0, \ldots, K-1$ **do**
    for $j = 0, \ldots, N$ **do**
      $A_i^j \sim \pi_{\theta_{old}}(\cdot|S_i), R_{i+1}^j \leftarrow r(S_i, A_i^j), S_{i+1}^j \sim p(\cdot|S_i, A_i^j)$
      if S_{i+1}^j *is a terminal state* **then**
        $m_i^j \leftarrow 0, S_{i+1}^j \sim \mu_0$
      else
        $m_i^j \leftarrow 1$
      end
    end
    $S_{i+1} \leftarrow S_{i+1}^0$
  end
  for $i = K-1, \ldots, 0$ **do**
    $\text{Adv}_i \leftarrow R_{i+1}^0 + \gamma_{\text{C}} m_i^0 \hat{v}_\psi(S_{i+1}^0) - \hat{v}_\psi(S_i^0)$
    for $j = 0, \ldots, N$ **do**
      $S_i'^j \leftarrow S_{i+1}^j$
    end
    Store $(\{S_i^j, A_i^j, m_i^j, r_i^j, S_i'^j\}_{j=0,\ldots,N}, \text{Adv}_i)$ in M
  end
  Normalize Adv_i in M as $\text{Adv}_i \leftarrow \frac{\text{Adv}_i - \text{mean}(\{\text{Adv}_i\})}{\text{std}(\{\text{Adv}_i\})}$
  for $o = 1, \ldots, K_{opt}$ **do**
    Sample a minibatch $\{(\{S_i^j, A_i^j, m_i^j, r_i^j, S_i'^j\}_{j=0,\ldots,N}, \text{Adv}_i)\}_{i=1,\ldots,B}$ from M
    $y_i \leftarrow \frac{1}{N+1} \sum_{j=0}^{N} r_i^j + \gamma_{\text{C}} m_i^j \hat{v}_\psi(S_i'^j)$
    $L(\psi) \leftarrow \frac{1}{2B} \sum_{i=1}^{B} (\hat{v}_\psi(S_i^0) - y_i)^2$ /* No gradient through y_i */
    $L(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \min\{\frac{\pi_\theta(A_i^0|S_i^0)}{\pi_{\theta_{old}}(A_i^0|S_i^0)} \text{Adv}_i, \text{clip}(\frac{\pi_\theta(A_i^0|S_i^0)}{\pi_{\theta_{old}}(A_i^0|S_i^0)}) \text{Adv}_i\}$
    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i^0|S_i^0) - \log \pi_\theta(A_i^0|S_i^0) < KL_{target}$ **then**
      Perform one gradient update to θ maximizing $L(\theta)$ with Adam
    end
  end
end

Algorithm 4: PPO-FHTD

Input:
θ, ψ: parameters of $\pi, \{\hat{v}^j\}_{j=1,\dots,H}$
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold

$S_0 \sim \mu_0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta$
  for $i = 0, \dots, K-1$ **do**
    $A_i \sim \pi_{\theta_{old}}(\cdot|S_i)$
    Execute A_i, get R_{i+1}, S_{i+1}
    if S_{i+1} *is a terminal state* **then**
      $m_i \leftarrow 0, S_{i+1} \sim \mu_0$
    else
      $m_i \leftarrow 1$
    end
  end
  for $i = K-1, \dots, 0$ **do**
    $\text{Adv}_i \leftarrow R_{i+1} + m_i \hat{v}^H_\psi(S_{i+1}) - \hat{v}^H_\psi(S_i)$
    $S'_i \leftarrow S_{i+1}, r_i \leftarrow R_{i+1}$
    Store $(S_i, A_i, m_i, r_i, S'_i, \text{Adv}_i)$ in M
  end
  Normalize Adv_i in M as $\text{Adv}_i \leftarrow \frac{\text{Adv}_i - \text{mean}(\{\text{Adv}_i\})}{\text{std}(\{\text{Adv}_i\})}$
  for $o = 1, \dots, K_{opt}$ **do**
    Sample a minibatch $\{(S_i, A_i, m_i, r_i, S'_i, \text{Adv}_i)\}_{i=1,\dots,B}$ from M
    for $j = 1, \dots, H$ **do**
      $y^j_i \leftarrow r_i + m_i \hat{v}^{j-1}_\psi(S'_i))$ /* $\hat{v}^0(S'_i) \equiv 0$ */
    end
    $L(\psi) \leftarrow \frac{1}{2B} \sum_{i=1}^{B} \sum_{j=1}^{H} (\hat{v}^j_\psi(S_i) - y^j_i)^2$ /* No gradient through y^j_i */
    $L(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \min\{\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \text{Adv}_i, \text{clip}(\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \text{Adv}_i\}$
    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i|S_i) - \log \pi_\theta(A_i|S_i) < KL_{target}$ **then**
      Perform one gradient update to θ maximizing $L(\theta)$ with Adam
    end
  end
end

Algorithm 5: PPO-C51

Input:
θ, ψ: parameters of $\pi, \{\hat{v}^j\}_{j=1,\dots,N_s}$ with N_s being the number of supports and $\hat{v}^j$ being the probability of each support
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold

$\Delta_z \doteq \frac{2V_{\max}}{N_s - 1}, \{z_j \doteq -V_{\max} + (j-1)\Delta_z : j = 1, \dots, N_s\}$ `// Define the supports`
$S_0 \sim \mu_0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta$
  for $i = 0, \dots, K-1$ **do**
    $A_i \sim \pi_{\theta_{old}}(\cdot|S_i)$
    Execute A_i, get R_{i+1}, S_{i+1}
    if S_{i+1} *is a terminal state* **then**
      $m_i \leftarrow 0, S_{i+1} \sim \mu_0$
    else
      $m_i \leftarrow 1$
    end
  end
  for $i = K-1, \dots, 0$ **do**
    $\text{Adv}_i \leftarrow R_{i+1} + m_i \gamma_{\text{C}} \sum_{j=1}^{N_s} \hat{v}^j_\psi(S_{i+1}) z_j - \sum_{j=1}^{N_s} \hat{v}^j_\psi(S_i) z_j$
    $S'_i \leftarrow S_{i+1}, r_i \leftarrow R_{i+1}$
    Store $(S_i, A_i, m_i, r_i, S'_i, \text{Adv}_i)$ in M
  end
  Normalize Adv_i in M as $\text{Adv}_i \leftarrow \frac{\text{Adv}_i - \text{mean}(\{\text{Adv}_i\})}{\text{std}(\{\text{Adv}_i\})}$
  for $o = 1, \dots, K_{opt}$ **do**
    Sample a minibatch $\{(S_i, A_i, m_i, r_i, S'_i, \text{Adv}_i)\}_{i=1,\dots,B}$ from M
    for $i = 1, \dots, B$ **do**
      for $j = 1, \dots, N_s$ **do**
        $z^i_j \leftarrow r_i + m_i \gamma_{\text{C}} z_j$
      end
    end
    for $j = 1, \dots, N_s$ **do**
      $y^i_j \leftarrow \sum_{k=1}^{N_s} [1 - \frac{|[z^i_j]^{V_{\max}}_{-V_{\max}} - z_j|}{\Delta_z}]^1_0 \hat{v}^k_\psi(S'_i)$ `/* ` $[x]^u_l \doteq \min(\max(x, l), u)$ `*/`
    end
    $L(\psi) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{N_s} -y^i_j \log \hat{v}^j_\psi(S_i)$ `/* No gradient through` y^i_j `*/`
    $L(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \min\{\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \text{Adv}_i, \text{clip}(\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \text{Adv}_i\}$
    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i|S_i) - \log \pi_\theta(A_i|S_i) < KL_{target}$ **then**
      Perform one gradient update to θ maximizing $L(\theta)$ with Adam
    end
  end
end

Algorithm 6: DisPPO

Input:
θ, ψ: parameters of $\pi, \hat{v}$
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold

$S_0 \sim \mu_0, t \leftarrow 0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta$
  for $i = 0, \ldots, K-1$ **do**
    $A_i \sim \pi_{\theta_{old}}(\cdot|S_i), t_i \leftarrow t$
    Execute A_i, get R_{i+1}, S_{i+1}
    if S_{i+1} *is a terminal state* **then**
      $m_i \leftarrow 0, S_{i+1} \sim \mu_0, t \leftarrow 0$
    else
      $m_i \leftarrow 1, t \leftarrow t+1$
    end
  end
  $G_K \leftarrow \hat{v}(S_K)$
  for $i = K-1, \ldots, 0$ **do**
    $G_i \leftarrow R_{i+1} + \gamma_{\mathrm{C}} m_i G_{i+1}$
    $\mathrm{Adv}_i \leftarrow R_{i+1} + \gamma_{\mathrm{C}} m_i \hat{v}_\psi(S_{i+1}) - \hat{v}_\psi(S_i)$
    Store $(S_i, A_i, G_i, \mathrm{Adv}_i, t_i)$ in M
  end
  Normalize Adv_i in M as $\mathrm{Adv}_i \leftarrow \frac{\mathrm{Adv}_i - \mathrm{mean}(\{\mathrm{Adv}_i\})}{\mathrm{std}(\{\mathrm{Adv}_i\})}$
  for $o = 1, \ldots, K_{opt}$ **do**
    Sample a minibatch $\{(S_i, A_i, G_i, \mathrm{Adv}_i, t_i)\}_{i=1,\ldots,B}$ from M
    $L(\psi) \leftarrow \frac{1}{2B} \sum_{i=1}^{B} (\hat{v}_\psi(S_i) - G_i)^2$ `/* No gradient through` G_i `*/`
    $L(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^{B} \gamma_{\mathrm{A}}^{t_i} \min\{\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \mathrm{Adv}_i, \mathrm{clip}(\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \mathrm{Adv}_i\}$
    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i|S_i) - \log \pi_\theta(A_i|S_i) < KL_{target}$ **then**
      Perform one gradient update to θ maximizing $L(\theta)$ with Adam
    end
  end
end

Algorithm 7: AuxPPO

Input:
θ, θ', ψ: parameters of $\pi, \pi', \hat{v}$
α_A, α_C: Initial learning rates of the Adam optimizers for θ, ψ
K, K_{opt}, B: rollout length, number of optimization epochs, and minibatch size
KL_{target}: maximum KL divergence threshold

$S_0 \sim \mu_0, t \leftarrow 0$
while *True* **do**
  Initialize a buffer M
  $\theta_{old} \leftarrow \theta, \theta' \leftarrow \theta$
  for $i = 0, \ldots, K-1$ **do**
    $A_i \sim \pi_{\theta_{old}}(\cdot|S_i), t_i \leftarrow t$
    Execute A_i, get R_{i+1}, S_{i+1}
    if S_{i+1} *is a terminal state* **then**
      $m_i \leftarrow 0, S_{i+1} \sim \mu_0, t \leftarrow 0$
    else
      $m_i \leftarrow 1, t \leftarrow t+1$
    end
  end
  $G_K \leftarrow \hat{v}(S_K)$
  for $i = K-1, \ldots, 0$ **do**
    $G_i \leftarrow R_{i+1} + \gamma_{\mathrm{C}} m_i G_{i+1}$
    $\mathrm{Adv}_i \leftarrow R_{i+1} + \gamma_{\mathrm{C}} m_i \hat{v}_\psi(S_{i+1}) - \hat{v}_\psi(S_i)$
    Store $(S_i, A_i, G_i, \mathrm{Adv}_i, t_i)$ in M
  end
  Normalize Adv_i in M as $\mathrm{Adv}_i \leftarrow \frac{\mathrm{Adv}_i - \mathrm{mean}(\{\mathrm{Adv}_i\})}{\mathrm{std}(\{\mathrm{Adv}_i\})}$
  for $o = 1, \ldots, K_{opt}$ **do**
    Sample a minibatch $\{(S_i, A_i, G_i, \mathrm{Adv}_i, t_i)\}_{i=1,\ldots,B}$ from M
    $L(\psi) \leftarrow \frac{1}{2B} \sum_{i=1}^{B} (\hat{v}_\psi(S_i) - G_i)^2$ `/* No gradient through` G_i `*/`

$$L(\theta, \theta') \leftarrow \frac{1}{B} \sum_{i=1}^{B} \gamma_{\mathrm{C}}^{t_i} \min\{\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \mathrm{Adv}_i, \mathrm{clip}(\frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \mathrm{Adv}_i\} + (1 - \gamma_{\mathrm{C}}^{t_i}) \min\{\frac{\pi_{\theta'}(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)} \mathrm{Adv}_i, \mathrm{clip}(\frac{\pi_{\theta'}(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}) \mathrm{Adv}_i\}$$

    Perform one gradient update to ψ minimizing $L(\psi)$ with Adam
    if $\frac{1}{B} \sum_{i=1}^{B} \log \pi_{\theta_{old}}(A_i|S_i) - \log \pi_\theta(A_i|S_i) < KL_{target}$ **then**
      Perform one gradient update to θ, θ' maximizing $L(\theta, \theta')$ with Adam
    end
  end
end

C ADDITIONAL EXPERIMENTAL RESULTS

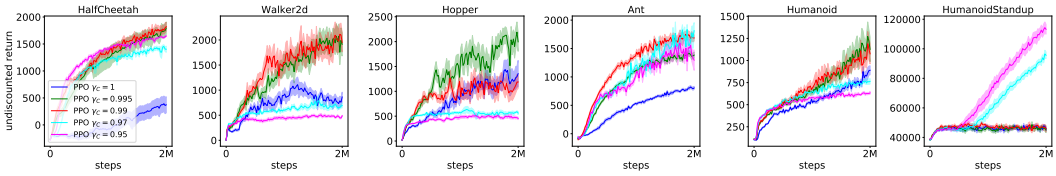


Figure 11: The default PPO implementation with different discount factors.

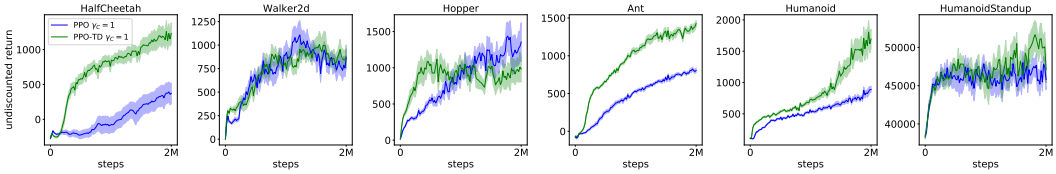


Figure 12: Comparison between PPO and PPO-TD when $\gamma_c = 1$.

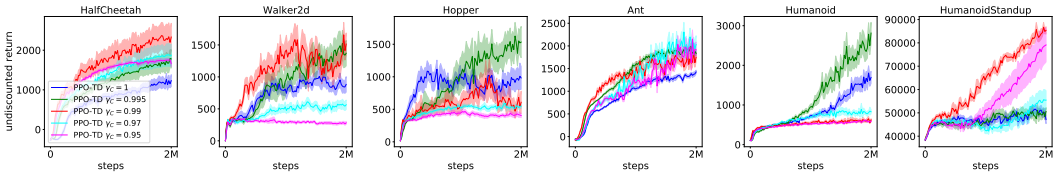


Figure 13: PPO-TD with different discount factors.

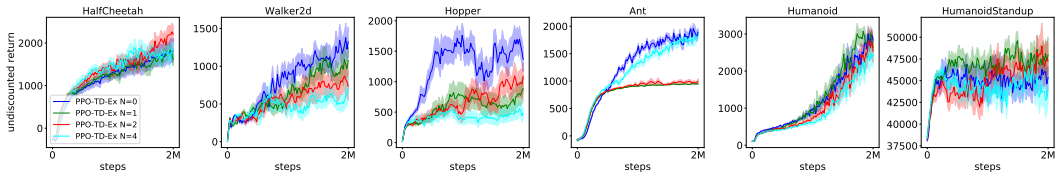


Figure 14: PPO-TD-Ex ($\gamma_{\text{C}} = 0.995$).

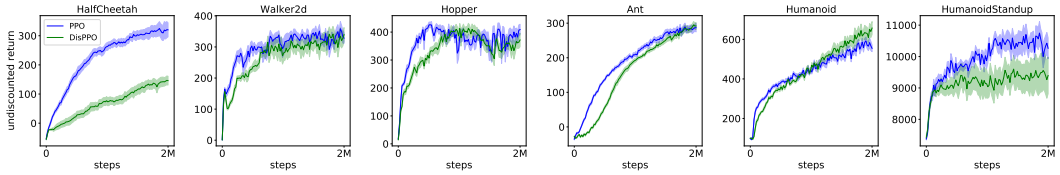


Figure 15: Comparison between PPO and DisPPO with $\gamma = 0.995$

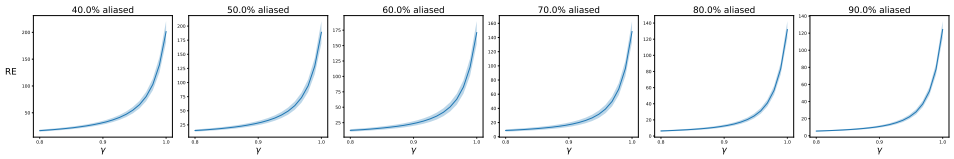


Figure 16: Unnormalized representation error (RE) as a function of the discount factor. Shaded regions indicate one standard derivation. RE is computed analytically as $\text{RE}(X, \gamma) \doteq \min_w ||Xw - v_\gamma||_2$