We investigate this mismatch in two scenarios. In the first scenario, we consider optimizing an undiscounted objective. Actor-critic algorithms have enjoyed great success both theoretically and in practice (Konda [21], Schulman et al. [35], Sutton et al. [42], Williams [50]), and empirically (Mnih et al. [29], OpenAI [31], Schulman et al. [37], Silver et al. [38]). There is, however, a longstanding gap between the theory behind actor-critic algorithms and how practitioners implement them. Let $\gamma$, $\gamma_a$, and $\gamma_c$ 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 $\gamma_a = \gamma_c = \gamma$ (Schulman et al. [35], Sutton et al. [42]) or at least keep $\gamma_a = \gamma_c$ if Blackwell optimality (Veinott [48], Weitzman [49]) is considered. Practitioners, however, usually use $\gamma_a = 1$ and $\gamma_c < 1$ in their implementations (Achiam [1], Caspi et al. [9], Dhariwal et al. [12], Kostrikov [22], Liang et al. [27], Stooke and Abbeel [39], Zhang [54]). Although this mismatch and its theoretical disadvantage have been recognized by Nota and Thomas [30], Thomas [44], 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 $\gamma_a = \gamma_c = \gamma = 1$. Practitioners, however, usually use $\gamma_a = \gamma_c = \gamma = 1$ but $\gamma_c < 1$, introducing bias. We explain the benefits from this mismatch with the following hypothesis:

**Hypothesis 1.** $\gamma_c < 1$ optimizes a bias-variance-representation trade-off.

It is easy to see that $\gamma_c < 1$ reduces the variance in bootstrapping targets. We also provide empirical evidence showing that when $\gamma_c < 1$, it may become easier to find a good representation than when $\gamma_c = 1$ for reasons beyond the reduced variance. Consequently, although using $\gamma_c < 1$ introduces bias, it can facilitate representation learning. For our empirical study, we make use of fixed horizon temporal difference learning (De Asis et al. [11]) to disentangle the various effects of the discount factor on the learning process.

**Scenario 2. The true objective function is discounted: $\gamma = \gamma_c < 1$.**

Theoretically, there is a $\gamma^t$ term for the actor update on a transition observed at time $t$ in a trajectory (Konda [21], Schulman et al. [35], Sutton et al. [42]). Practitioners, however, usually ignore this term while using a discounted critic, i.e., $\gamma_a = \gamma_c = \gamma < 1$ over the unbiased setup (i.e., $\gamma_c = \gamma_a = \gamma = 1$) comes from improved representation learning.

Our empirical study involves implementing the difference between the biased and unbiased setup as an auxiliary task such that the difference contributes to the learning process through only representation learning. 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, the unbiased setup outperforms the biased setup.

2 BACKGROUND
Markov Decision Processes: We consider an infinite horizon MDP with a finite state space $S$, a finite action space $A$, a bounded reward function $r : S \rightarrow \mathbb{R}$, a transition kernel $p : S \times S \times A \rightarrow [0, 1]$, an initial state distribution $\mu_0$, and a discount factor $\gamma \in [0, 1].$ The initial state $S_0$ is sampled from $\mu_0$. At time step $t$, an agent in state $S_t$ takes action $A_t \sim \pi(S_t)$, where $\pi : A \times S \rightarrow \{0, 1\}$ is the policy it follows. The agent then gets a reward $R_{t+1} \equiv r(S_t)$ and proceeds to the next state $S_{t+1} \sim p(S_t, A_t)$. The return of the policy $\pi$ at time step $t$ is defined as

$$G_t \equiv \sum_{i=1}^{\infty} \gamma^{t-1} R_{t+i},$$

which allows us to define the state value function

$$v^\pi_t(s) \equiv \mathbb{E}[G_t | S_t = s],$$

and the state-action value function

$$q^\pi_t(s, a) \equiv \mathbb{E}[G_t | S_t = s, A_t = a].$$

We consider episodic tasks where we assume there is an absorbing state $s^\infty \in S$ such that $r(s^\infty) = 0$ and $p(s^\infty | s^\infty, a) = 1$ holds for any $a \in A$. When $\gamma < 1$, $v^\pi_t$ and $q^\pi_t$ are always well defined. When $\gamma = 1$, to ensure $v^\pi_t$ and $q^\pi_t$ are well defined, we further assume finite expected episode length. Let $\Gamma^\pi_t$ be a random variable denoting the first time step that an agent hits $s^\infty$ when following $\pi$ given $S_0 = s$. We assume $T_{\text{max}} = \sup_s \pi_{\text{eff}} \max_s \mathbb{E}[T^\pi_0] < \infty$, where $\pi$ is parameterized by $\theta$ and $\Pi$ is the corresponding function class. Similar assumptions are also used in stochastic shortest path problems (e.g., Section 2.2 of Bertsekas and Tsitsiklis [7]). In our experiments, all the environments have a hard time limit of 1000, i.e., $T_{\text{max}} = 1000$. This is standard practice: classic RL environments also have an upper limit on their episode lengths (e.g. 27k in Bellemare et al. [5, ALE]). Following Pardo et al. [33], we add the (normalized) time step $t$ in the state to keep the environment Markovian under the presence of the hard time limit. We measure the performance of a policy $\pi$ with

$$J_{\gamma}(\pi) \equiv \mathbb{E}_{S_0 \sim \mu_0}[v^\pi_0(S_0)].$$

Vanilla Policy Gradient: Sutton et al. [42] compute $\nabla \theta J_{\gamma}(\pi)$ as

$$\nabla \theta J_{\gamma}(\pi) \equiv \sum_s d^\pi_0(s) \sum_a q^\pi_0(s, a) \nabla \theta \log \pi(a | s),$$

where

$$d^\pi_0(s) \equiv \begin{cases} \sum_{t=0}^{\infty} \gamma^t \text{Pr}(S_t = s | \mu_0, p, \pi), & \gamma < 1, \\ \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \text{Pr}(S_t = s | S_0, p, \pi)], & \gamma = 1. \end{cases}$$

Note $d^\pi_0$ remains well-defined for $\gamma = 1$ when $T_{\text{max}} < \infty$. In order to optimize the policy performance $J_{\gamma}(\pi)$, one can follow (1) and, at time step $t$, update $\theta_t$ as

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla \theta J_{\gamma}(S_t, A_t) \nabla \theta \log \pi(A_t | S_t),$$

where $\alpha$ is a learning rate. If we replace $q^\pi_0$ with a learned value function, the update rule (2) becomes an actor-critic algorithm, where the actor refers to $\pi$ and the critic refers to the learned approximation of $q^\pi_0$. In practice, an estimate for $q^\pi_0$ instead of $q^\pi_0$ is usually learned. Theoretically, we should have $\gamma_a = \gamma_c = \gamma$. Practitioners, however, usually ignore the $\gamma_c$ term in (2), and use $\gamma_c < \gamma_a = 1$. What this update truly optimizes remains an open problem (Nata and Thomas [30]).

TRPO and PPO: To improve the stability of actor-critic algorithms, Schulman et al. [35] propose Trust Region Policy Optimization (TRPO), based on the performance improvement lemma:

**Lemma 2.1. (Theorem 1 in Schulman et al. [35]):** For $\gamma < 1$ and any two policies $\pi$ and $\pi'$, we have

$$J_{\gamma}(\pi') \geq J_{\gamma}(\pi) + \sum_s d^\pi_0(s) \sum_a \pi'(a | s) \text{Adv}^\pi_{\gamma}(s, a)$$

$$\geq (1 - \gamma^2 T_{\text{max}}^\pi_s) \text{KL}(\pi || \pi').$$

where

$$\text{Adv}^\pi_{\gamma}(s, a) \equiv \mathbb{E}_{s' \sim p(\cdot | s, a)}[r(s) + \gamma v^\pi_{\gamma}(s') - v^\pi_{\gamma}(s)].$$

is the advantage,

$$\epsilon(\pi, \pi') \equiv \max_s \mathbb{E}_{s' \sim p(\cdot | s, a)}[r(s) + \gamma v^\pi_{\gamma}(s') - v^\pi_{\gamma}(s)]$$

is the advantage,

$$\epsilon(\pi, \pi') \equiv \max_s \mathbb{E}_{s' \sim p(\cdot | s, a)}[r(s) + \gamma v^\pi_{\gamma}(s') - v^\pi_{\gamma}(s)],$$

and $D_{KL}$ refers to the KL divergence.

To facilitate our empirical study, we first make a theoretical contribution by extending Lemma 2.1 to the undiscounted setting. We have the following lemma:

**Lemma 2.2.** Assuming $T_{\text{max}} < \infty$, for $\gamma = 1$ and any two policies $\pi$ and $\pi'$, we have

$$J_{\gamma}(\pi') \geq J_{\gamma}(\pi) + \mathbb{E}[\text{Adv}^\pi_{\gamma}(s, a)|\text{Adv}^\pi_{\gamma}(s, a)|\pi'],$$

$$-\max_{\pi, \pi'} \text{Adv}^\pi_{\gamma}(s, a)|\pi'||\pi').$$

The proof of Lemma 2.2 is provided in the appendix. A practical implementation of Lemmas 2.1 and 2.2 is to compute a new policy $\theta$ via gradient ascent on the clipped objective:

$$L(\theta) \equiv \sum_{t=0}^{\infty} \gamma^t L_t(\theta, \theta_{\text{old}}),$$

where

$$L_t(\theta, \theta_{\text{old}}) \equiv \min_{\pi \in \Pi_{\text{eff}}} \pi(\theta(S_t) | \pi_{\text{old}}(S_t, A_t)) \pi(\theta(S_t) | \pi_{\text{old}}(S_t, A_t))$$

Here $S_t$ and $A_t$ are sampled from $\pi_{\text{old}}$, and

$$\text{clip}(x) \equiv \max(\min(x, 1 + \epsilon), 1 - \epsilon)$$

with $\epsilon$ a hyperparameter. Theoretically, we should have $\gamma_a = \gamma_c$, but practical algorithms like Proximal Policy Optimization (PPO, Schulman et al. [37]) usually use $\gamma_c < \gamma_a = 1$.

**Policy Evaluation:** We now introduce several policy evaluation techniques we use in our empirical study. Let $\hat{\pi}$ be our estimate of $\pi_{\text{old}}$. At time step $t$, Temporal Difference learning (TD, Sutton [40]) updates $\hat{\pi}$ as

$$\hat{\pi}(S_t) \leftarrow \hat{\pi}(S_t) + \alpha(R_{t+1} + \gamma \hat{\pi}(S_{t+1}) - \hat{\pi}(S_t)).$$

Instead of the infinite horizon discounted return $G_t$, De Asis et al. [11] propose to consider the $H$-step return

$$G^H_t \equiv \sum_{i=1}^{H} R_{t+i}.$$
Correspondingly, the $H$-step value function is defined as
\[ v^H_i(s) = \mathbb{E}[Q^H_i(S_T)|S_t = s]. \]

We let \( \hat{v}^H \) be our estimate of \( v^H_i \). At time step \( t \), De Asis et al. [11] use the following update rule to learn \( \hat{v}^H_i \). For \( i = 1, \ldots, H \):
\[ \hat{v}^i(S_t) \leftarrow \hat{v}^i(S_t) + \alpha(R_{t+1} + \gamma^{i-1}(S_{t+1}) - \hat{v}^i(S_t)), \]
where \( \hat{v}^0(s) \equiv 0 \). In other words, to learn \( \hat{v}^H_i \), we need to learn \( \{\hat{v}^i\}_{i=1}^H \) simultaneously. De Asis et al. [11] call (3) Fixed Horizon Temporal Difference learning (FHTD).

**Methodology:** We consider MuJoCo (Todorov et al. [45]) robot simulation tasks from OpenAI gym (Brockman et al. [8]) as our benchmark. Given its popularity in understanding deep RL algorithms (Andrychowicz et al. [4], Engstrom et al. [13], Henderson et al. [17], Ilyas et al. [18]) and designing new deep RL algorithms (Fujimoto et al. [15], Haarnoja et al. [16]), 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 (GAE, Schulman et al. [36]), which has a tunable hyperparameter \( \gamma \). The roles of \( \gamma \) and \( \lambda \) 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 \( R_{t+1} + \gamma(S_{t+1}) - \hat{v}(S_t) \). The PPO pseudocode we follow is provided 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 grid search for the learning rate of each algorithmic configuration (i.e., for every curve in all figures). All experimental 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 30 independent runs with shaded regions indicating standard errors. All our implementations and our Docker environment are publicly available for future research.\(^3\)

### 3 OPTIMIZING THE UNDISCOUNTED OBJECTIVE (SCENARIO 1)

In this scenario, the goal is to optimize the undiscounted objective \( J_{y=1}(\pi) \). This scenario is related to most practitioners as they usually use the undiscounted return as the performance metric (Haarnoja et al. [16], Mnih et al. [29], Schulman et al. [37]). One theoretically grounded option is to use \( \gamma_a = \gamma = 1 \). By using \( \gamma_a = 1 \) and \( \gamma = 1 \), practitioners introduce bias. We first empirically confirm that introducing bias in this way indeed has empirical advantages. A simple first hypothesis for this is that \( \gamma = 1 \) leads to lower variance in Monte Carlo return bootstrapping targets than \( \gamma = 1 \); it thus optimizes a bias-variance trade-off. However, we further show that there are empirical advantages from \( \gamma > 0 \) that cannot be explained solely 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 \( \gamma_a = 1 \).

**Bias-variance trade-off:** To investigate the advantages of using \( \gamma < 1 \), we first test default PPO with \( \gamma \in \{0.95, 0.97, 0.99, 0.995, 1\} \). We find that the best discount factor is always with \( \gamma = 0.99 \) and that \( \gamma = 1 \) usually leads to a performance drop (Figure 1). 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 always a Monte Carlo return, i.e., the sum of the undiscounted return \( R_t \). Here \( \gamma \) has a gating effect in controlling the variance of the Monte Carlo return: when \( \gamma = 1 \), and therefore \( \gamma = 1 \), the randomness from \( R_t \) contributes less to the variance of the Monte Carlo return. In this paper, we refer to this gating effect as variance control. As the objective is undiscounted and we use \( \gamma_a = \gamma = 1 \), theoretically we should also use \( \gamma_a = 1 \) when computing the Monte Carlo return if we do not want to introduce bias. By using \( \gamma < 1 \), bias is introduced. The variance of the Monte Carlo return is, however, also reduced. Consequently, a simple hypothesis for the empirical advantage of using \( \gamma < 1 \) is that it optimizes a bias-variance trade-off. We find, however, that there is more at play.

**Beyond bias-variance trade-off:** To make other possible effects of \( \gamma \) pronounced, it is desirable to reduce the variance control effect of \( \gamma \). To this end, we benchmark PPO-TD (pseudocode 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 \( R_{t+1} + \gamma_v(S_{t+1}) - \hat{v}(S_t) \). In this update target, \( \gamma_v \) gates the randomness from only the immediately successor state \( S_{t+1} \). By contrast, in the original Monte Carlo update target, \( \gamma_v \) gates the randomness of all future states and rewards. Figure 2 shows that PPO-TD (\( \gamma_v = 1 \)) outperforms PPO (\( \gamma_v = 1 \)) in four games. This indicates that PPO-TD might be less vulnerable to the large variance in critic update targets introduced by using \( \gamma_v = 1 \) than default PPO. Figure 3 suggests, however, that even for PPO-TD, \( \gamma_v < 1 \) is still preferable to \( \gamma_v = 1 \).

Of course, in PPO-TD, \( \gamma_v \) still has the variance control effect, though not as pronounced as that in default PPO. To make other possible effects of \( \gamma_v \) more pronounced, we benchmark PPO-TD-Ex (pseudocode 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
\[ \frac{1}{N_{i=0}^{N}} \sum_{i=0}^{N} R_{i+t+1}^{0} + \gamma_v \hat{v}(S_{i+t+1}^{i}) \]
where \( R_{i+t+1}^{0} \) and \( S_{i+t+1}^{i} \) refer to the original reward and successor state. To get \( R_{i+t+1}^{0} \) and \( S_{i+t+1}^{i} \) for \( i \in \{1, \ldots, N\} \), we first sample an action \( A_i^{t} \) from the sampling policy, then reset the environment to \( S_t \), and finally execute \( A_i^{t} \) to get \( R_{i+t+1}^{i} \) and \( S_{i+t+1}^{i} \). The advantage for the actor update in PPO-TD-Ex is estimated with \( R_{i+t+1}^{0} + \gamma_v \hat{v}(S_{i+t+1}^{i}) - \hat{v}(S_t) \) regardless of \( \gamma_v \) to further depress its variance control effect. Importantly, we do not count those \( N \) extra transitions in the x-axis when plotting. If we use the true value function instead of \( \hat{v} \), \( N \geq 1 \) should always outperform \( N = 0 \) as the additional transitions help reduce variance (assuming \( \gamma_v \) is fixed). However, in practice we have only \( \hat{v} \), which is not trained on the extra successor states \( \{S_{i+t+1}^{i}\}_{i=1}^{N} \).

So the quality of the prediction \( \hat{v}(S_{i+t+1}^{i}) \) depends mainly on the generalization of \( \hat{v} \). Consequently, increasing \( N \) risks potential erroneous prediction \( \hat{v}(S_{i+t+1}^{i}) \). That being said, though not guaranteed to improve the performance, when the prediction \( \hat{v}(S_{i+t+1}^{i}) \) is decent,
increasing $N$ should at least not lead to a performance drop. As shown by Figure 4, PPO-TD-Ex ($\gamma_c = 0.99$) roughly follows this intuition. However, surprisingly, providing any extra transition this way to PPO-TD-Ex ($\gamma_c = 1$) leads to a significant performance drop in 4 out of 6 tasks (Figure 5). This drop suggests that the quality of the critic $\hat{v}$, at least in terms of making prediction on untrained states $S_{t+1}, \ldots, S_{t+N}$, is lower when $\gamma_c = 1$ is used than $\gamma_c < 1$. In other words, the generalization of $\hat{v}$ becomes poorer when $\gamma_c$ is increased from 0.99 to 1. The curves for PPO-TD-Ex ($\gamma_c = 0.995$) are a mixture of $\gamma_c = 0.99$ and $\gamma_c = 1$ and are provided in the appendix. The limited generalization could imply that representation learning becomes harder when $\gamma_c$ is increased. By representation learning, we refer to learning the lower 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., Chung et al. [10], Jaderberg et al. [19], Veeriah et al. [47].

In PPO-TD, the bootstrapping target for training $\hat{v}(S_t)$ is $R_{t+1} + \gamma_c \hat{v}(S_{t+1})$, where $\gamma_c$ has two roles. First, it gates the randomness from $S_{t+1}$, which is the aforementioned variance control. Second, it affects the value function $v^\gamma_c$, that we want to approximate via changing the horizon of the policy evaluation problem, which could possibly affect the difficulty of learning a good estimate $\hat{v}$ for $v^\gamma_c$ directly, not through the variance, which we refer to as learnability.
control (see, e.g., Laroche and van Seijen [25], Lehnert et al. [26], Romoff et al. [34]). Both roles can be responsible for the increased difficulty in representation learning when γ_c is increased. In the rest of this section, we provide empirical evidence showing that the changed difficulty in representation learning, resulting directly from the changed horizon of the policy evaluation problem, is at play when using the γ_c < 1, which, together with the previously established bias-variance trade-off, suggests that a bias-variance-representation trade-off is at play when practitioners use γ_c < 1.

Bias-representation trade-off: To further disentangle the variance control effect and learnability control effect of γ_c, we use FHTD to train the critic in PPO, which we refer to as PPO-FHTD (pseudo-code in the appendix). PPO-FHTD always uses γ_c = 1 regardless of H. The critic update target in PPO-TD is {R_{t+1} + γ_c \hat{v}(S_{t+1})}, whose variance is
\[
\text{Var}(R_{t+1} + γ_c \hat{v}(S_{t+1})|S_t) = \text{Var}(r(S_t) + γ_c \hat{v}(S_{t+1})|S_t) = γ^2_c \text{Var}(\hat{v}(S_{t+1})|S_t).
\]

By contrast, the critic update target in PPO-FHTD for \( \hat{v}^i(S_t) \) is {R_{t+1} + \hat{v}^{i-1}(S_{t+1})}, with variance:
\[
\text{Var}(R_{t+1} + \hat{v}^{i-1}(S_{t+1})|S_t) = \text{Var}(\hat{v}^{i-1}(S_{t+1})|S_t).
\]

On the one hand, manipulating H in FHTD changes the horizon of the policy evaluation problem, which corresponds to the role of learnability control of γ_c. On the other hand, manipulating H does not change the multiplier proceeding the variance term (i.e. (4) and (5)) and thus separates variance control from the learnability control.

We test two parameterizations for PPO-FHTD to investigate representation learning. In the first parameterization, to learn \( v_H^\pi \), we parameterize \( \{v_H^\pi\}_{i=1,\ldots,H} \) as H different heads over the same representation layer (backbone). In the second parameterization, we always learn \( \{v_H^\pi\}_{i=1,\ldots,1024} \) as 1024 different heads over the same representation layer, regardless of what H we are interested in. To approximate \( v_H^\pi \), we then simply use the output of the H-th head. Figure 8 further illustrates the difference between the two parameterizations.

Figure 6 shows that with the first parameterization, the best H for PPO-FHTD is usually smaller than 1024. Figure 7, however, suggests that for the second parameterization, H = 1024 is almost always among the best choices of H. Comparing Figures 3 and 7 shows that the performance of PPO-FHTD (H = 1024) is close to the performance of PPO-TD (γ_c = 1) as expected, since for any parameterization.
\[ H \geq T_{\text{max}} = 1000, \text{always have } \psi(H) \equiv v_{\pi}^{Y_1}(s). \] This performance similarity suggests that learning \( \{v_{\pi}^H\}_{i=1}^{1024} \) 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. Then, comparing Figures 6 and 7, we conclude that in the tested domains, learning \( v_{\pi}^H \) with different \( H \) requires different representations. This suggests that we can interpret the results in Figure 6 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 6. 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 7. Overall, \( \gamma_c \) optimizes a bias-representation trade-off by changing the policy evaluation horizon \( H \).

We further conjecture that representation learning may be harder for a longer horizon because the volume of all of good representations can become smaller. We provide a simulated example to support this. Consider policy evaluation on the simple Markov Reward Process (MRP) from Figure 10. We assume the reward for each transition is fixed, which 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 \( x_s[i] \equiv \tan (\xi) \), where \( \xi \) is a random variable uniformly distributed in \([-\pi, \pi]\). We choose this setup as we use \( \pi \) 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 [28]), which is common under function approximation, we first randomly split the \( N \) states into \( S_1 \) and \( S_2 \) such that \( |S_1| = \alpha N \) and \( |S_2| = (1 - \alpha)N \), where \( \alpha \) is the proportion of states to be aliased. Then for every \( s \in S_1 \), we randomly select an \( \hat{s} \in S_2 \) 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 \( \gamma \). For a feature matrix \( X \), the NRE is computed analytically as

\[
\text{NRE}(\gamma) \equiv \min_w \frac{\|Xw - v_{\gamma}\|_2}{\|v_{\gamma}\|_2},
\]

where \( v_{\gamma} \) is the analytically computed true value function of the MRP. We report the results in Figure 9, 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 \( \gamma \) increases, which suggests that, in this MRP, the volume of good representations (e.g., representations whose normalized representation error are smaller than some threshold) becomes smaller under a larger \( \gamma \) than that under a smaller \( \gamma \).

Importantly, in this MRP experiment, we compute all the quantities analytically so no variance in involved within a single trial. Consequently, representation error is a property of \( v_{\gamma} \) itself. We report the unnormalized representation error in the appendix, where we report the trend is much clearer.

Overall, though we do not claim that there is a monotonic relationship between the discount factor and the difficulty of representation learning, our empirical study suggests that representation learning is a key factor at play in the misuse of the discounting in actor-critic algorithms, beyond the widely recognized bias-variance trade-off.

### 4 Optimizing the Discounted Objective (Scenario 2)

When our goal is to optimize the discounted objective \( J_{Y<1}(\pi) \), theoretically we should have the \( Y_1^c \) term in the actor update and use \( \gamma_c < 1 \). Practitioners, however, usually ignore this \( Y_1^c \) term (i.e., set \( \gamma_c = 1 \)), introducing bias (see, e.g., the default PPO in the Appendix). By adding this missing \( Y_1^c \) term back (i.e., setting \( \gamma_c = \gamma < 1 \)), we end up with an unbiased implementation, which we refer to as DistPPO (pseudocode in the Appendix). Figure 11, however, shows that even if we use the discounted return as the performance metric, the biased implementation of PPO still outperforms the theoretically grounded unbiased implementation DistPPO in some tasks. \(^6\) We propose to interpret the empirical advantages of PPO over DistPPO with Hypothesis 2. For all experiments in this section, we use \( \gamma_c = \gamma < 1 \).

**An auxiliary task perspective:** The biased policy update implementation of (2) ignoring \( Y_1^c \) can be decomposed into two parts as

\[
J_{\pi}^{\gamma}(S_t, A_t) \equiv \log \pi(A_t|S_t) + (1 - \gamma^t) \log \pi(A_t|S_t).
\]

We propose to interpret the difference term between the biased implementation \( J_{\pi}^{\gamma}(S_t, A_t) \) and the theoretically grounded implementation \( J_{\pi}^{\gamma}(S_t, A_t) \), i.e., the \((1 - \gamma^t) \log \pi(A_t|S_t)\) term, as the gradient of an auxiliary objective with a dynamic weighting \( 1 - \gamma^t \). Let

\[
J_{s, \mu}(\pi) \equiv \sum_s \mu(s)q_{\mu}^\gamma(s, a),
\]

we have

\[
\nabla_\theta J_{s, \mu}(\pi)_{|\mu=\pi} = \mathbb{E}_{a \sim \pi(s)}[q_{\pi}^\gamma(s, a)\nabla_\theta \log \pi(a|s)].
\]

This objective changes every time step (through \( \mu \)). Inspired by the decomposition, we augment PPO with this auxiliary task, yielding AuxPPO (pseudocode in the appendix). In AuxPPO, we have two policies \( \pi \) and \( \pi^a \) parameterized by \( \theta \) and \( \theta^a \) respectively. The

---

\(^6\)See the appendix for more details about task selection.

\(^6\)In this scenario, by a task we mean the combination of a game and a discount factor.
two policies are two heads over the same neural network backbone, where \( \pi \) is used for interaction with the environment and \( \pi' \) is the policy for the auxiliary task. AuxPPO optimizes \( \theta \) and \( \theta' \) simultaneously by considering the following joint loss

\[
L(\theta, \theta') = \sum_{t=0}^{\infty} \gamma^t L_t(\theta, \theta_{\text{old}}) + (1 - \gamma^t) L_t(\theta', \theta_{\text{old}}),
\]

where \( S_t \) and \( A_t \) are obtained by executing \( \pi_{\text{old}} \). We additionally synchronize \( \theta' \) with \( \theta \) periodically to avoid an off-policy learning issue. By contrast, the objectives for PPO and DisPPO are

\[
\sum_{t=0}^{\infty} \gamma^t L_t(\theta, \theta_{\text{old}}) + (1 - \gamma^t) L_t(\theta, \theta_{\text{old}})
\]

and

\[
\sum_{t=0}^{\infty} \gamma^t L_t(\theta, \theta_{\text{old}})
\]

respectively. Figure 12 further illustrates the architecture of AuxPPO.

**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, which allows us to create a new environment by simply defining a new reward function

\[
r'(s, t) = r(s)I_{t \leq t_0} - r(s)I_{t > t_0},
\]

where \( I \) 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 \( \gamma^{t_0} \) is sufficiently small, e.g., we define \( t_0 \) \( \leq \) min\( \{ \gamma^t < 0.05 \} \). With this criterion for selecting \( t_0 \), the later transitions (i.e., transitions after \( t_0 \) steps) have little influence on the evaluation objective, the discounted return. Consequently, the later transitions affect the overall learning process mainly through representation learning. DisPPO rarely makes use of the later transitions due to the \( \gamma^t \) term in the gradient update. AuxPPO makes use of the later transitions only through representation learning (i.e., through the training of \( \pi' \)). PPO exploits the later transitions for representation learning and the later transitions also affect the control policy of PPO directly.

**Results:** When we consider the original environments, Figure 11 shows that in 8 out of 12 tasks, PPO outperforms DisPPO, even if the performance metric is the discounted episodic return. In all those 8 tasks, by using the difference term as an auxiliary task, AuxPPO is able to improve upon DisPPO. In 5 out of those 8 tasks, AuxPPO is able to roughly match the performance of PPO at the end of training. For \( \gamma \in \{0.95, 0.93, 0.9\} \) in Art, the improvement of AuxPPO is not clear and we conjecture that this is because learning of the \( \pi \)-head (the control head) in AuxPPO is much slower than the learning of \( \pi \) in PPO due to the \( \gamma^t \) term. Overall, this suggests that the benefit of PPO over DisPPO comes mainly from representation learning.

When we consider the environments with flipped rewards, PPO is outperformed by DisPPO and AuxPPO by a large margin in 10 out of 12 tasks. The transitions after \( t_0 \) steps are not directly relevant when the performance metric is the discounted return. However, learning on those transitions may still improve representation learning provided that those transitions are similar to the earlier transitions, which is the case in the original environments. PPO and AuxPPO, therefore, outperform DisPPO. However, when those transitions are much different from the earlier transitions, which is the case in the environments with flipped rewards, updating the control policy \( \pi_0 \) directly based on those transitions becomes distracting. DisPPO, therefore, outperforms PPO. Unlike PPO, AuxPPO does not update the control policy \( \pi_0 \) on later transitions directly. Provided that the network has enough capacity, the irrelevant transitions do not affect the control policy \( \pi_0 \) in AuxPPO much. The performance of AuxPPO is, therefore, similar to that of DisPPO.

To summarize, Figure 11 suggests that using \( \gamma_s = 1 \) is simply an inductive bias that all transitions are equally important. When this inductive bias is helpful for learning, \( \gamma_s = 1 \) implicitly implements auxiliary tasks thus improving representation learning and the overall performance. When this inductive bias is detrimental, however, \( \gamma_s = 1 \) can lead to significant performance drops. AuxPPO appears to be a safe choice that does not depend much on the correctness of this inductive bias.

5 RELATED WORK

The mismatch in actor-critic algorithm implementations has been previously studied. Thomas [44] focuses on the natural policy gradient setting and shows that the biased implementation ignoring \( \gamma_s \) 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 and Thomas [30] 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 \( J_{\gamma, \mu}(\pi) \) changes at every time step. Nota and Thomas [30] further provide a counterexample showing that following the biased gradient can lead to a poorly performing policy w.r.t. both discounted and undiscounted objectives. Both Thomas [44] and Nota and Thomas [30], however, focus on theoretical disadvantages of the biased gradient and regard ignoring \( \gamma_s \) as
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., Amit et al. [3], Fedus et al. [14], Jiang et al. [20], Laroche et al. [23], Laroche and van Seijen [25], Lehnert et al. [26], Sutton [41], Van Seijen et al. [46]), the analysis of which we leave for future work. In Scenario 1, using γc < 1 helps reduce the variance. Variance reduction in RL itself is an active research area (see, e.g., Papini et al. [32], Xu et al. [52], Yuan et al. [53]). Investigating those variance reduction techniques with γc = 1 is another possibility for future work. Recently, Bengio et al. [6] study the effect of the bootstrapping parameter λ in TD(λ) in generalization.

Figure 11: Curves without any marker are obtained in the original Ant / HalfCheetah. Diamond-marked curves are obtained in Ant / HalfCheetah with r′.

Our work studies the effect of the discount factor γ in representation learning in the context of the misuse of the discounting in actor-critic algorithms, sharing a similar spirit of Bengio et al. [6].

6 CONCLUSION

In this paper, we investigated the longstanding mismatch between theory and practice in actor-critic algorithms from a representation learning perspective. Although the theoretical understanding of policy gradient algorithms has recently advanced significantly (Agarwal et al. [2], Wu et al. [51]), this mismatch has drawn little attention. We proposed to understand this mismatch from a bias-representation trade-off perspective and an auxiliary task perspective for two different scenarios. 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, as well as draw more attention to the mismatch, which could enable the community to finally close this longstanding gap.

ACKNOWLEDGMENTS

We thank Geoffrey J. Gordon, Marc-Alexandre Cote, Bei Peng, and Dipendra Misra for the insightful discussion. Part of this work was done during SZ’s internship at Microsoft Research Montreal. SZ is also funded by the Engineering and Physical Sciences Research Council (EPSRC). This project has received funding from the European Research Council under the European Union’s Horizon 2020 research and innovation programme (grant agreement number 637713). Part of the experiments was made possible by a generous equipment grant from NVIDIA.

REFERENCES

[1] Joshua Achiam. 2018. Spinning Up in Deep Reinforcement Learning. (2018).
[2] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. 2019. Optimality and approximation with policy gradient methods in markov decision processes. arXiv preprint arXiv:1908.00261 (2019).
[3] Ron Amit, Ron Meir, and Kamal Ciosek. 2020. Discount Factor as a Regularizer in Reinforcement Learning. arXiv preprint arXiv:2007.02040 (2020).
[4] Marcin Andrychowicz, Anton Rachkov, Piotr Stanezyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hasenot, Matthieu Geist, Olivier Pietquin,
Marcin Michalski, et al. 2020. 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. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research 47 (jun 2015), 253–279.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. 2020. Interference and Generalization in Temporal Difference Learning. arXiv preprint arXiv:2003.06350 (2020).

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

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

Iai Caspi, Gal Leibovich, Gal Novik, and Shadi Endravisz. 2017. Reinforcement Learning Coach. https://doi.org/10.5281/zenodo.1134899

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

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

Pratulla Dhariwal, Christopher Hesse, Alex Nichol, Matthias Plappert, Alex Radford, John Schulman, Szymon Sidor, Yuhua Wu, and Peter Zhoukho. 2017. OpenAI Baselines. https://github.com/openai/baselines.

Logan Engstrom, Andrew Ilyas, Shihabi Santurkar, Dimitris Tsipras, Furdaus Janoos, Larry Rudolph, and Aleksander Madry. 2019. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. In International Conference on Learning Representations.

William Fedus, Carles Galeda, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. 2019. Hyperbolic discounting and learning over multiple horizons. arXiv preprint arXiv:1902.09477 (2019).

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01298 (2018).

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

Andrew Ilyas, Logan Engstrom, Shihabi Santurkar, Dimitris Tsipras, Furdaus Janoos, Larry Rudolph, and Aleksander Madry. 2018. A Closer Look at Deep Policy Gradients. arXiv preprint arXiv:1811.02553 (2018).

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

Nan Jiang, Satinder P Singh, and Ambuj Tewari. 2016. On Structural Properties of MDPs that Bound Loss Due to Shallow Planning. In Proceedings of the 33rd International Conference on Machine Learning.

Richard S Sutton. 2018. Alogarithmic Mapping to Enable Lower Discount Factors in Reinforcement Learning. AAMAS 2022, May 9–13, 2022, Online. AAMAS 2022, May 9–13, 2022, Online.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning.

Chris Note and Philip S. Thomas. 2020. Is the Policy Gradient a Gradient? In Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. CoRR.

OpenAI. 2018. OpenAI Five. https://openai.com/five/.

Matteo Papini, Damiano Binaghi, Giuseppe Canocano, Matteo Pirotta, and Marcello Restelli. 2018. Stochastic variance-reduced policy gradient. arXiv preprint arXiv:1806.05618 (2018).

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

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

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. 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. 2016. Mastering the game of Go with deep neural networks and tree search. Nature (2016).

Stanisław Ulam. 1971. 1940. Lectures on Games of Strategy. International Congress of Mathematicians.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. 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. 2016. Mastering the game of Go with deep neural networks and tree search. Nature (2016).

Stanisław Ulam. 1971. 1940. Lectures on Games of Strategy. International Congress of Mathematicians.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. 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. 2016. Mastering the game of Go with deep neural networks and tree search. Nature (2016).

Stanisław Ulam. 1971. 1940. Lectures on Games of Strategy. International Congress of Mathematicians.