Averaging n-step Returns Reduces Variance in Reinforcement Learning

Brett Daley [1 2] **Martha White** [1 2 3] **Marlos C. Machado** [1 2 3]

Abstract

Multistep returns, such as n-step returns and λ-returns, are commonly used to improve the sample efficiency of reinforcement learning (RL) methods. The variance of the multistep returns becomes the limiting factor in their length; looking too far into the future increases variance and reverses the benefits of multistep learning. In our work, we demonstrate the ability of compound returns—weighted averages of n-step returns—to reduce variance. We prove for the first time that any compound return with the same contraction modulus as a given n-step return has strictly lower variance. We additionally prove that this variance-reduction property improves the finite-sample complexity of temporal-difference learning under linear function approximation. Because general compound returns can be expensive to implement, we introduce two-bootstrap returns which reduce variance while remaining efficient, even when using minibatched experience replay. We conduct experiments showing that compound returns often increase the sample efficiency of n-step deep RL agents like DQN and PPO.

1. Introduction

Efficiently learning value functions is critical for reinforcement learning (RL) algorithms. Value-based RL methods (e.g., Watkins, 1989; Rummery & Niranjan, 1994; Mnih et al., 2015) encode policies implicitly in a value function, and policy evaluation is the principal mechanism of learning. Even when RL methods instead learn parametric policies, accurate value functions are needed to guide policy improvement (e.g., Silver et al., 2014; Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018) or to serve as a baseline (e.g., Barto et al., 1983; Sutton, 1984; Williams, 1992; Schulman et al., 2015a). Quicker policy evaluation is therefore useful to many RL algorithms.

One way to achieve faster value-function learning is through multistep returns, in which more than one reward following an action is used to construct a prediction target. Multistep returns are used extensively in deep RL (e.g. Schulman et al., 2015b; Mnih et al., 2016; Munos et al., 2016; Schulman et al., 2017; Hessel et al., 2018; Schrittwieser et al., 2020; Wurman et al., 2022; Chebotar et al., 2023; Schwarzer et al., 2023), where the value function is approximated by a neural network. In theory, multistep returns incorporate more information regarding future outcomes, leading to faster credit assignment and, in turn, faster learning. However, faster learning is not guaranteed in practice because looking farther into the future increases variance and can end up requiring more samples to estimate the mean. These two opposing factors must be balanced to achieve fast and stable learning. The most common multistep returns are n-step returns and λ-returns, both of which span between standard temporal-difference (TD; Sutton, 1988) learning ($n = 1$ or $\lambda = 0$) and Monte Carlo learning ($n \to \infty$ or $\lambda = 1$).

Implementation has generally been the main consideration in choosing which of these return estimators to use. When the value function is a lookup table or a linear parametric function, the λ-return is preferred for its efficient implementation using TD(λ) with eligibility traces (Sutton, 1988). However, in off-policy deep RL, the value function is a neural network trained with an experience replay buffer (Lin, 1992), so the extra bootstrapping performed by the λ-return becomes costly and n-step returns are more common. Although recent work has explored ways to efficiently train neural networks using replayed λ-returns (e.g., Munos et al., 2016; Harb & Precup, 2016; Daley & Amato, 2019), λ-returns generally remain more expensive or complex to implement than n-step returns.

Despite its more complicated implementation in deep RL, the λ-return is interesting because it averages many n-step returns together. In this paper, we show that this averaging reduces variance compared to n-step returns and that the variance reduction leads to faster learning. More generally, the λ-return is an example of a *compound* re-

[1]Department of Computing Science, University of Alberta, Edmonton, AB, Canada [2]Alberta Machine Intelligence Institute [3]Canada CIFAR AI Chair. Correspondence to: Brett Daley <brett.daley@ualberta.ca>.

turn, or a weighted average of two or more n-step returns (Sutton & Barto, 2018, Sec. 12.1). The λ-return is the most prevalent compound return, but others such as the γ-return (Konidaris et al., 2011) and the Ω-return (Thomas et al., 2015) have also been shown to be effective. While compound returns are known to be theoretically sound (Watkins, 1989, Sec. 7.2), they have not been extensively analyzed. Compound returns are ideally suited for deep RL because experiences are already stored in a replay memory, making it easy to compute several n-step returns cheaply and then average them together in any desired way.

In this paper, we formally investigate the variance of compound returns, deriving a variance model for arbitrary compound returns. Additionally, we prove that compound returns have a *variance-reduction property*: any compound return with the same contraction modulus as an n-step return has strictly lower variance if the TD errors are not perfectly correlated. As a corollary, there exists a λ-return for every n-step return with the same contraction modulus but lower variance, implying a better bias-variance trade-off.

Our theoretical results suggest one should prefer λ-returns over n-step returns, but these are expensive to implement for deep RL. Thus, we also introduce an efficient approximation called **Pi**ecewise **λ-R**eturn (PiLaR or Pilar). Pilar is computed by averaging just two n-step returns together—the most efficient compound return possible. The lengths of the n-step returns are chosen to put weights on the TD errors that are close to those assigned by TD(λ), thereby achieving similar variance reduction as the λ-return. We show that Pilars improve sample efficiency compared to n-step returns when used to train Deep Q-Network (DQN; Mnih et al., 2015), which is the foundation of many other off-policy deep RL methods. In on-policy methods like PPO (Schulman et al., 2017) where such an approximation is not necessary, we also show similar improvements for λ-returns, also known as Generalized Advantage Estimation in this context (Schulman et al., 2015b), over n-step returns.

2. Background

Value-based RL agents interact with their environments to iteratively improve estimates of their expected cumulative reward. By convention, the agent-environment interaction is modeled as a Markov decision process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, p, r)$. At time t, the agent observes the environment state, $S_t \in \mathcal{S}$, and selects an action, $A_t \in \mathcal{A}$, accordingly. The environment then transitions to a new state, $S_{t+1} \in \mathcal{S}$, with probability $p(S_{t+1} \mid S_t, A_t)$, and returns a scalar reward, $R_{t+1} \stackrel{\text{def}}{=} r(S_t, A_t, S_{t+1})$. We assume the agent samples each action with probability $\pi(A_t \mid S_t)$, where π is a stochastic policy.

In the standard prediction problem, the agent's goal is to learn a value function $v \colon \mathcal{S} \to \mathbb{R}$ that estimates the expected discounted return $v_\pi(s)$ attainable in each state s. Letting $G_t \stackrel{\text{def}}{=} \sum_{i=0}^{\infty} \gamma^i R_{t+1+i}$ be the observed Monte Carlo return at time t, where $\gamma \in [0, 1]$ is a discount factor and actions are implicitly sampled from π, then $v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}[G_t \mid S_t = s]$.

The basic learning operation of value-based RL is a *backup*, which has the general form

$$v(S_t) \leftarrow v(S_t) + \alpha_t \left(\hat{G}_t - v(S_t) \right) , \qquad (1)$$

where $\hat{G}_t$ is a return estimate (the *target*) and $\alpha_t \in (0, 1]$ is the step size. Substituting various estimators for $\hat{G}_t$ leads to different learning properties. For instance, the Monte Carlo return could be used, but it suffers from high variance, and delays learning until the end of an episode. To reduce the variance and delay, return estimates can *bootstrap* from the value function. Bootstrapping is the fundamental mechanism underlying TD learning (Sutton, 1988), and the extent to which the chosen estimator, $\hat{G}_t$, bootstraps is a major factor in learning performance.

The most basic multistep version of TD learning uses the n-step return as its target in Eq. (1):

$$G_t^{(n)} \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} \gamma^i R_{t+1+i} + \gamma^n v(S_{t+n}) . \qquad (2)$$

Bootstrapping introduces bias in the update, since generally $\mathbb{E}\left[v(S_{t+n}) \mid S_t\right] \neq v_\pi(S_{t+n})$ before convergence, but it greatly reduces variance. The case of $n = 1$ corresponds to the classic TD(0) algorithm, $v(S_t) \leftarrow v(S_t) + \alpha_t \delta_t$, where $\delta_t \stackrel{\text{def}}{=} R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$ is the TD error. However, bootstrapping after just one time step is slow because long-term reward information must propagate indirectly through the value function, requiring many behavioral repetitions before v approximates v_π well. Larger values of n consider more rewards per update and assign credit faster, but at the price of increased variance, with $n \to \infty$ reverting to the Monte Carlo return. The choice of n thus faces a bias-variance trade-off (Kearns & Singh, 2000), with intermediate values typically performing best in practice (Sutton & Barto, 2018, Sec. 7.1).

Another type of multistep return is the λ-return, used by TD(λ) algorithms (Sutton, 1988), which is equivalent to an exponentially weighted average of n-step returns:

$$G_t^\lambda \stackrel{\text{def}}{=} (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)} ,$$

where $\lambda \in [0, 1]$. The λ-return is just one way to average n-step returns together, but any weighted average is possible. Such averages are known as compound returns (Sutton & Barto, 2018, Sec. 12.1), formally expressed as

$$G_t^{\boldsymbol{c}} \stackrel{\text{def}}{=} \sum_{n=1}^{\infty} c_n G_t^{(n)}, \tag{3}$$

where $\boldsymbol{c}$ is an infinite sequence of nonnegative weights with the constraint $\sum_{n=1}^{\infty} c_n = 1$. Eq. (3) is a strict generalization of all of the aforementioned return estimators; however, it technically constitutes a *compound* return if and only if at least two of the weights are nonzero—otherwise, it reduces to an n-step return. All choices of weights that sum to 1 are equally valid in the sense that using their compound returns as the target in Eq. (1) will converge to v_π under the standard step-size criteria of Robbins & Monro (1951). However, in practice, the choice of weights greatly affects the bias and variance of the estimator and, as a consequence, the empirical rate of convergence to v_π. Furthermore, different choices of weights will vary in the amount of computation needed; sparse weights require less bootstrapping and therefore less computation. A principal contribution of this paper is to shed light on the trade-offs between these factors and to develop new compound returns that favorably balance these considerations (see Section 4).

Many RL agents are control agents, meaning they do not just predict v_π for a fixed policy, but also use these estimates to update the behavior policy during training. One way to do this is Q-learning (Watkins, 1989). Rather than learning a state-value function v, the agent learns an action-value function $q \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that estimates the expected return $q_*(s, a)$ earned by following an optimal policy after taking action a in state s. The estimated value of a state is therefore $v(s) = \max_{a \in \mathcal{A}} q(s, a)$, and all of the previously discussed return estimators apply after making this substitution. Backups are conducted as before, but now operate on $q(S_t, A_t)$ instead of $v(S_t)$.

Learning is off-policy in this setting, since the agent now predicts returns for a greedy policy that differs from the agent's behavior. Any multistep return is therefore biased unless the expectation is explicitly corrected, e.g., by importance sampling (Kahn & Harris, 1951). However, it is common practice to ignore this bias in deep RL, and recent research has even suggested that doing so is more effective with both n-step returns (Hernandez-Garcia & Sutton, 2019) and λ-returns (Daley & Amato, 2019; Kozuno et al., 2021), the latter of which become Peng's Q(λ) (Peng & Williams, 1996). For these reasons, we also forgo off-policy corrections in our work.

3. Variance Analysis

Our main goal in this section is to derive conditions for when a compound return reduces variance compared to a given n-step return. We call this the *variance-reduction property* (see Theorem 3.7). An important consequence of this property is that when these conditions are met and both

returns are chosen to have the same contraction modulus, the compound return needs fewer samples than the n-step return to converge (see Theorem 3.9).

The outline of this section is as follows. We first derive variance models for n-step returns and compound returns under generalized assumptions extending prior work (Section 3.1) and then develop a way to pair different return families by equating their worst-case bias (Section 3.2). This sets up our principal theoretical result, the variance-reduction property of compound returns (Section 3.3). We also conduct a finite-time analysis of TD learning with compound returns to show that this variance reduction does improve theoretical sample complexity (Section 3.4). Finally, we experimentally demonstrate that λ-returns—an instance of compound returns—outperform n-step returns in a prediction task as indicated by our theory (Section 3.5).

3.1. Characterizing Compound-Return Variance

We start by developing reasonable models for the variance of n-step returns and compound returns, building on the work of Konidaris et al. (2011). We note that our real quantity of interest in this section is the conditional variance of the backup error $\hat{G}_t - v(S_t)$. The degree of this quantity's deviation from its expected value is what ultimately impacts the performance of value-based RL methods in Eq. (1). Nevertheless, this turns out to be the same as the conditional variance of the return $\hat{G}_t$, since $v(S_t)$ contributes no randomness: $\mathrm{Var}[\hat{G}_t - v(S_t) \mid S_t] = \mathrm{Var}[\hat{G}_t \mid S_t]$. This equivalence allows us to interchange the variances of a return and its error, depending on which is more computationally convenient.

Modeling a compound return's variance is challenging because it typically requires making assumptions about how the variance of n-step returns increase as a function of n, as well as how strongly correlated different n-step returns are. If these assumptions are too strong, the derived variances fail to reflect reality and lead to poorly informed algorithmic choices. For instance, consider the following compound return, which we call a *two-bootstrap return*:

$$(1-c)G_t^{(n_1)} + c G_t^{(n_2)}, \ c \in (0,1), \ n_1 < n_2. \tag{4}$$

Let $\sigma_n^2 \stackrel{\text{def}}{=} \mathrm{Var}[G_t^{(n)} \mid S_t]$. The variance of the above is

$$(1-c)^2 \sigma_1^2 + c^2 \sigma_2^2 + 2c(1-c)\sigma_1 \sigma_2 \rho,$$

where $\rho = \mathrm{Corr}[G_t^{(n_1)}, G_t^{(n_2)} \mid S_t]$. To evaluate this expression, it is tempting to assume either $\rho = 0$ to remove the covariance term, or $\rho = 1$ because both returns are generated from the same trajectory. However, neither assumption is fully correct. We can see this more clearly by

decomposing the returns into their constituent TD errors:

$$G_t^{(n_1)} = v(S_t) + \sum_{i=0}^{n_1-1} \gamma^i \delta_{t+i}\,,$$

$$G_t^{(n_2)} = v(S_t) + \sum_{i=0}^{n_1-1} \gamma^i \delta_{t+i} + \sum_{i=n_1}^{n_2-1} \gamma^i \delta_{t+i}\,.$$

Because the returns share the first n_1 TD errors, averaging them as in Eq. (4) can only reduce the variance from the last $n_2 - n_1$ TD errors. Two different n-step returns are therefore neither uncorrelated nor perfectly correlated; they consist of various random elements, some of which are shared and cannot be averaged, and others which are not shared and can be averaged. Models that fail to account for this partial averaging lead to poor variance predictions.

To be accurate, our variance analysis should therefore start with the TD error as the fundamental unit of randomness within a return. In the following analysis, we generalize the n-step return variance assumptions made by Konidaris et al. (2011, Sec. 3) to obtain an expression for the covariance between two TD errors. We assume that there exist $\kappa \geq 0$ and $\rho \in [0, 1]$ such that the following hold:

Assumption 3.1. Each TD error has uniform variance: $\mathrm{Var}[\delta_{t+i} \mid S_t] = \kappa,\ \forall\, i \geq 0$.

Assumption 3.2. TD errors are uniformly correlated: $\mathrm{Corr}[\delta_{t+i}, \delta_{t+j} \mid S_t] = \rho,\ \forall\, i \geq 0, \forall\, j \neq i$.

We must make some assumptions about the TD errors because not much can be said otherwise about the variance of a return estimate for an arbitrary MDP. For example, it is possible to contrive problems in which 1-step TD learning actually has much higher variance than Monte Carlo estimation. Consider an MDP with reward zero everywhere and a value function initialized with random numbers; the Monte Carlo return would not have variance but TD bootstrapping would. However, it is generally *expected* that 1-step TD has much lower variance than Monte Carlo, because summing more TD errors (i.e., random variables) together will very likely increase practical variance. This intuition motivates our assumptions, as we want a variance model that captures this behavior while remaining tractable for analysis. We further discuss the justification and historical context of these assumptions in Appendix A.

Now, we can unify Assumptions 3.1 and 3.2 as

$$\mathrm{Cov}[\delta_{t+i}, \delta_{t+j} \mid S_t] = ((1-\rho)\mathbf{1}_{i=j} + \rho)\kappa\,, \quad (5)$$

where $\mathbf{1}_{i=j} = 1$ if $i = j$, and is 0 otherwise. In the proposition below, we derive a variance model for the n-step return by decomposing the return into a sum of discounted TD errors and then adding up the pairwise covariances given by Eq. (5). For brevity, we define a function $\Gamma_k(n)$ to represent

partial sums of a geometric series with common ratio γ^k. That is, $\Gamma_k(n) = (1 - \gamma^{kn}) / (1 - \gamma^k)$ for $\gamma < 1$, and $\Gamma_k(n) = n$ for $\gamma = 1$.

Proposition 3.3. *The variance of an n-step return is*

$$\mathrm{Var}[G_t^{(n)} \mid S_t] = (1 - \rho)\,\Gamma_2(n)\,\kappa + \rho\,\Gamma_1(n)^2\,\kappa\,.$$

Proof. See Appendix C.1. □

Our n-step variance model linearly interpolates between an optimistic case where TD errors are uncorrelated ($\rho = 0$) and a pessimistic case where TD errors are maximally correlated ($\rho = 1$). In the maximum-variance scenario of $\gamma = 1$, we have $\Gamma_1(n) = \Gamma_2(n) = n$, so the model becomes $(1 - \rho)n\kappa + \rho n^2\kappa$, i.e., it interpolates between linear and quadratic functions. In Appendix B, we demonstrate that these bounds are consistent with empirical data in real environments, even when our assumptions do not fully hold.

Eq. (5) enables us to go beyond n-step returns and calculate variances for arbitrary compound returns. We accomplish this by again decomposing the return into a weighted sum of TD errors (see the next lemma) and then applying our assumptions to derive a compound variance model in the following proposition.

Lemma 3.4. *A compound error can be written as a weighted summation of TD errors:*

$$G_t^{\boldsymbol{c}} - v(S_t) = \sum_{i=0}^{\infty} \gamma^i h_i \delta_{t+i}\,, \ \text{where } h_i \stackrel{\text{def}}{=} \sum_{n=i+1}^{\infty} c_n\,.$$

Proposition 3.5. *The variance of a compound return is*

$$\mathrm{Var}[G_t^{\boldsymbol{c}} \mid S_t] = (1-\rho)\sum_{i=0}^{\infty} \gamma^{2i} h_i^2 \kappa + \rho \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} h_i h_j \kappa\,.$$

Proofs. See Appendices C.2 and C.3. □

The cumulative weights $(h_i)_{i=0}^{\infty}$ fully specify the variance of a compound return. For instance, the λ-return assigns a cumulative weight of $h_i = \sum_{n=i+1}^{\infty} (1-\lambda)\lambda^{n-1} = \lambda^i$ to the TD error δ_{t+i}, which matches the TD(λ) algorithm (Sutton, 1988). Substituting this weight into Proposition 3.5 and solving the geometric series yields the variance for the λ-return, which we show in Appendix C.4.

3.2. Error-Reduction Property and Effective n-step

Proposition 3.5 provides a method for calculating variance, but we would still like to show that compound returns reduce variance relative to n-step returns. To do this, we first need a way to relate two returns in terms of their expected performance. This is because low variance by itself is not sufficient for fast learning; for example, the 1-step return has very low variance, but learns slowly.

In the discounted setting, a good candidate for expected learning speed is the return's contraction modulus. The contraction modulus is the constant factor by which the maximum value-function error between v and v_π is guaranteed to be reduced. When the contraction modulus is less than 1, the return estimator exhibits an *error-reduction property*: i.e., the maximum error decreases on average with every backup iteration of Eq. (1). This property is commonly used in conjunction with the Banach fixed-point theorem to prove that v eventually converges to v_π (see, e.g., Bertsekas & Tsitsiklis, 1996, Sec. 4.3). The error-reduction property of compound returns was first identified by Watkins (1989, Sec. 7.2) and is expressed formally as

$$\max_{s \in \mathcal{S}} |\mathbb{E}\left[G_t^{\boldsymbol{c}} \mid S_t = s\right] - v(s)|$$

$$\leq \left(\sum_{k=1}^{\infty} c_k \gamma^k \right) \max_{s \in \mathcal{S}} |v(s) - v_\pi(s)| . \quad (6)$$

The contraction modulus is the coefficient on the right-hand side: $\sum_{k=1}^{\infty} c_k \gamma^k$, a weighted average of each individual n-step return's contraction modulus, γ^n. We can compare compound returns to n-step returns that have equivalent error-reduction properties by solving the equation $\gamma^n = \sum_{k=1}^{\infty} c_k \gamma^k$ for n. We call this the *effective n-step* of the compound return, since the compound return reduces the worst-case error as though it were an n-step return whose length is the solution to the previous equation (see Proposition 3.6 below). In undiscounted settings, we cannot directly equate contraction moduli like this because they all become $\sum_{k=1}^{\infty} c_k = 1$, but we can still solve the limit as $\gamma \to 1$ to define the effective n-step.

Proposition 3.6 (Effective n-step of compound return)**.** *Let $G_t^{\boldsymbol{c}}$ be any compound return and let*

$$n = \begin{cases} \log_\gamma \left(\sum_{k=1}^{\infty} c_k \gamma^k \right), & \text{if } 0 < \gamma < 1, \\ \sum_{k=1}^{\infty} c_k k, & \text{if } \gamma = 1. \end{cases} \quad (7)$$

When n is an integer, $G_t^{\boldsymbol{c}}$ shares the same bound in Eq. (6) as the n-step return $G_t^{(n)}$.

Proof. See Appendix C.5. □

With or without discounting, an n-step return and a compound return that satisfy Eq. (7) have the same error-reduction property. We refer to the quantity $\sum_{k=1}^{\infty} c_k k$ as the *center of mass* (COM) of a return, since it is the first moment of the weight distribution over n-step returns. Intuitively, this represents the average length into the future considered by the return, and is the undiscounted analog of the log contraction modulus.

3.3. The Variance-Reduction Property

With the previous definitions, we are now ready to formalize the *variance-reduction property* of compound returns.

Theorem 3.7 (Variance-reduction property of compound returns)**.** *Let $G_t^{(n)}$ be any n-step return and let $G_t^{\boldsymbol{c}}$ be any compound return with the same effective n-step: i.e., $\boldsymbol{c}$ satisfies Proposition 3.6. The inequality $\mathrm{Var}[G_t^{\boldsymbol{c}} \mid S_t] \leq \mathrm{Var}[G_t^{(n)} \mid S_t]$ always holds, and is strict whenever TD errors are not perfectly correlated ($\rho < 1$).*

Proof. See Appendix C.6. □

Theorem 3.7 shows that whenever a compound return has the same contraction modulus ($\gamma < 1$) or COM ($\gamma = 1$) as an n-step return, it has lower variance as long as the TD errors are not perfectly correlated. Perfect correlation between all TD errors would be unlikely to occur except in contrived, maximum-variance MDPs; thus, compound returns reduce variance in most cases. Crucially, variance reduction is achieved for any type of weighted average—although the magnitude of reduction does depend on the specific choice of weights. The exact amount, in terms of κ, can be calculated by subtracting the compound variance from the n-step variance for a given contraction modulus or COM. As an example, we bound the variance reduction of a λ-return in the following corollary.

Corollary 3.8 (Variance reduction of λ-return)**.** *The magnitude of variance reduction for a λ-return is bounded by*

$$0 \leq \mathrm{Var}[G_t^n \mid S_t] - \mathrm{Var}[G_t^\lambda \mid S_t] \leq (1 - \rho)\frac{\lambda}{1 - \lambda^2}\kappa .$$

This magnitude is monotonic in γ and maximized at $\gamma = 1$.

Proof. See Appendix C.7. □

The magnitude of this variance reduction also increases monotonically as $\lambda \to 1$, showing that the potential for variance reduction improves as the effective n-step of the return increases. Interestingly, this is reflected in our later random-walk experiments in Section 3.5 (observe the performance gaps in Figure 1 when $\alpha \approx 1$). Future work can derive the weights that maximize the variance reduction in Theorem 3.7, but this will likely require more advanced techniques such as functional analysis and is beyond the scope of this paper.

3.4. Finite-Time Analysis

It might not be obvious that reducing variance leads to faster learning; indeed, in other settings such as direct policy optimization, this is not always the case (see Chung et al., 2021). We conduct a finite-time analysis of compound TD learning to prove that lower variance does lead to faster learning in this setting. We consider linear function approximation, where $v_\theta(s) = \phi(s)^\top \theta$ for features $\phi(s) \in \mathbb{R}^d$ and parameters $\theta \in \mathbb{R}^d$; note that tabular methods can be recovered using one-hot features. The parameters are iteratively updated according to

$$\theta_{t+1} = \theta_t + \alpha g_t^{\boldsymbol{c}}(\theta_t) , \quad (8)$$

$$\text{where } g_t^{\boldsymbol{c}}(\theta) \overset{\text{def}}{=} \left(G_t^{\boldsymbol{c}} - \phi(S_t)^\top \theta_t \right) \phi(S_t) .$$

Our theorem below generalizes recent analysis of 1-step TD learning (Bhandari et al., 2018, Theorem 2).

Theorem 3.9 (Finite-Time Analysis). *Suppose TD learning with linear function approximation is applied under an i.i.d. state model with stationary distribution $\boldsymbol{d} \in \mathbb{R}^{|\mathcal{S}|}$ (see Assumption C.1 in Appendix C.8) using the compound return estimator $G_t^{\boldsymbol{c}}$ as its target. Let $\beta \in [0,1)$ be the contraction modulus of the estimator and let $\sigma^2 \geq 0$ be the variance of the estimator. Assume that the features are normalized such that $\|\phi(s)\|_2^2 \leq 1, \ \forall \, s \in \mathcal{S}$. Define $C \stackrel{\text{def}}{=} (\|r\|_\infty + (1+\gamma)\|\theta^*\|_\infty) / (1-\gamma)$, where θ^* is the minimizer of the projected Bellman error for $G_t^{\boldsymbol{c}}$. For any $T \geq (4 / (1-\beta))^2$ and a constant step size $\alpha = 1 / \sqrt{T}$,*

$$\mathbb{E}\Big[\|v_{\theta^*} - v_{\bar{\theta}_T}\|_{\boldsymbol{D}}^2\Big] \leq \frac{\|\theta^* - \theta_0\|_2^2 + 2(1-\beta)^2 C^2 + 2\sigma^2}{(1-\beta)\sqrt{T}},$$

$$\text{where } \boldsymbol{D} \stackrel{\text{def}}{=} \operatorname{diag}(\boldsymbol{d}) \text{ and } \bar{\theta}_T \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1} \theta_t.$$

Proof. See Appendix C.8. □

With a constant step size, compound TD learning (and hence n-step TD learning as a special case) reduces the value-function error at the same asymptotic rate of $O(1/\sqrt{T})$ for any return estimator. However, both the contraction modulus β and the return variance σ^2 greatly influence the magnitude of the constant that multiplies this rate. Given an n-step return and a compound return with the same contraction modulus, the compound return has lower variance by Theorem 3.7 and therefore converges faster to its respective TD fixed point. Although these two fixed points may generally be different, we can show that the bound on their solution quality is the same by generalizing Lemma 6 of Tsitsiklis & Van Roy (1997) for an arbitrary return.

Proposition 3.10 (TD Solution Quality). *Let θ^* be the minimizer of the projected Bellman error under linear function approximation for any n-step or compound return with contraction modulus β. The following bound always holds:*

$$\|\boldsymbol{\Phi}\theta^* - v_\pi\|_{\boldsymbol{D}} \leq \frac{1}{1-\beta}\|\boldsymbol{\Pi} v_\pi - v_\pi\|_{\boldsymbol{D}}.$$

Proof. See Appendix C.9. □

This shows that the fixed-point error for an arbitrary return is always within $1 / (1-\beta)$ times the optimal error. Since β is equalized for the two returns in Theorem 3.9, the compound return converges to its fixed point faster than the n-step return converges to its fixed point, and the quality bound of these two fixed points is the same.

3.5. Case Study: λ-returns

Although the λ-return is often motivated by its efficient implementation using TD(λ) and eligibility traces, our theory indicates that λ-returns can also promote faster learning via variance reduction. We provide empirical support for this by demonstrating faster learning in the random-walk experiment from Sutton & Barto (2018, Sec. 12.1). In this environment, the agent begins in the center of a linear chain of 19 connected states and can move either left or right. The agent receives a reward only if it reaches one of the far ends of the chain (-1 for the left, $+1$ for the right), in which case the episode terminates. The agent's policy randomly moves in either direction with equal probability. We train the agents for 10 episodes, updating the value functions after each episode with offline backups like Eq. (1). To pair the n-step returns and λ-returns together, we derive the effective λ for an n-step return in the following proposition.

Proposition 3.11 (Effective λ of n-step return). *For any $n \geq 1$, when $\gamma < 1$, the λ-return with $\lambda = (1 - \gamma^{n-1}) / (1 - \gamma^n)$ has the same contraction modulus as the n-step return. When $\gamma = 1$, the λ-return with $\lambda = (n-1) / n$ has the same COM as the n-step return.*

Proof. See Appendix C.10. □

Because this is an undiscounted task, we use the relationship[1] $\lambda = (n-1) / n$ to generate several (n, λ)-pairs with equal COMs in Table 1. For our experiment, we choose four commonly used n-step values, $\{2, 3, 5, 10\}$, which correspond to $\lambda \in \{0.5, 0.67, 0.8, 0.9\}$. In Figure 1, we plot the average root-mean-square (RMS) value error (with respect to v_π) as a function of the step size α over 100 trials. We also indicate 95% confidence intervals by the shaded regions. For all of the tested (n, λ)-pairs, an interesting trend emerges. In the left half of each plot, variance is not an issue because the step size is small and has plenty of time to average out the randomness in the estimates. Learning therefore progresses at a nearly identical rate for both the n-step return and the λ-return since they have the same COM (although there is a small discrepancy due to the truncation of the episodic task). However, as $\alpha \to 1$ in the right half of each plot, variance becomes a significant factor as the step size becomes too large to mitigate the noise in the updates. This causes the n-step return's error to diverge sharply compared to the λ-return's as the λ-return manages variance more effectively. The lowest error attained by each λ-return is also better than that of its corresponding n-step return in all cases. Notably, neither of our variance assumptions hold perfectly in this environment, demonstrating that our variance model's predictions are useful in practical settings.

---

[1]Another way to write this relationship is $n = 1 / (1 - \lambda)$, which makes it clear how the effective n-step is affected by λ.
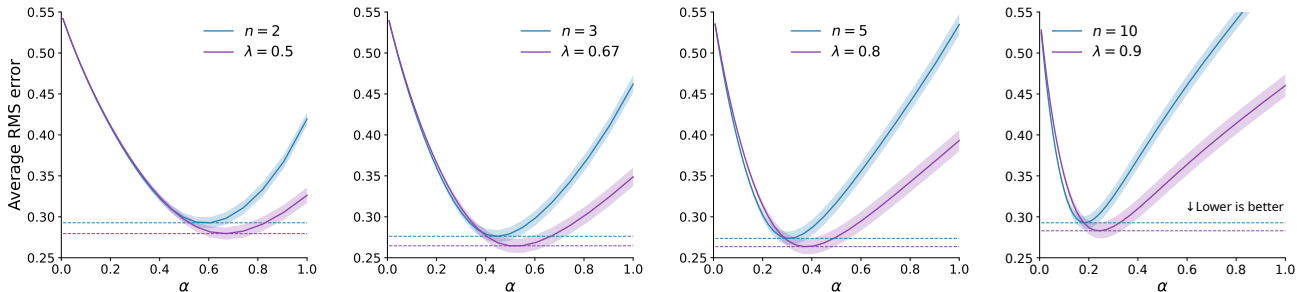
Figure 1. Comparing n-step returns and λ-returns, paired by COM, in a random walk. Dashed lines indicate the lowest errors attained.

Table 1. Common n-step returns and λ-returns with equal COMs.

n	2	3	4	5	10	20	50	100
λ	0.5	0.67	0.75	0.8	0.9	0.95	0.98	0.99

4. Piecewise λ-Returns

Although the previous experiment shows that λ-returns can achieve faster learning, they remain expensive for deep RL. This is because the λ-return at time t theoretically bootstraps on every time step after t (until the end of an episode), with each bootstrap requiring a forward pass through the neural network. Even when truncating the λ-return to a reasonable length, this is still several times more expensive than the single bootstrap of an n-step return. Previous work has amortized the cost of λ-returns over long trajectories by exploiting their recursive structure (e.g., Munos et al., 2016; Harb & Precup, 2016; Daley & Amato, 2019), but the price to pay for this efficiency is the requirement that experiences must be temporally adjacent, which can hurt performance. Our preliminary experiments confirmed this, indicating that the correlations within replayed trajectories counteract the benefits of λ-returns when compared to minibatches of n-step returns (with the batch size chosen to equalize computation).

We instead seek a compound return that approximates the variance-reduction property of the λ-return while being computationally efficient for minibatch replay. There are many ways we could average n-step returns together, and so we constrain our search by considering compound returns that 1) comprise an average of only two n-step returns to minimize cost, 2) preserve the contraction modulus or COM of the λ-return, and 3) place weights on the TD errors that are close to those assigned by the λ-return.

The first property constrains our estimator to have the two-bootstrap form of Eq. (4): $(1-c)G_t^{(n_1)} + cG_t^{(n_2)}$. Let n be our targeted effective n-step; the effective λ can be obtained from Proposition 3.11. Let us also assume $\gamma < 1$, although we include the case where $\gamma = 1$ in Appendix D. To preserve the contraction modulus as in the second property, we must satisfy $(1-c)\gamma^{n_1} + c\gamma^{n_2} = \gamma^n$. Assuming

that we have freedom in the choice of n_1 and n_2, it follows that $c = (\gamma^n - \gamma^{n_1}) / (\gamma^{n_2} - \gamma^{n_1})$.

We would thus like to find n_1 and n_2 such that the weights given to the TD errors optimize some notion of closeness to the TD(λ) weights, in order to fulfill the third and final property. Although there are many ways we could define the error, we propose to minimize the maximum absolute difference between the weights, since this ensures that no individual weight deviates too far from the TD(λ) weight. Recall that the weight given to TD error δ_{t+i} by

Figure 2. TD-error weights for Pilar and a λ-return ($\lambda = 0.904$). Both returns have the same contraction modulus as a 10-step return when $\gamma = 0.99$.

n-step return $G_t^{(n)}$ is γ^i if $i < n$, and is 0 otherwise. It follows that the two-bootstrap average assigns a weight of $h_i = \gamma^i$ if $i < n_1$; a weight of $h_i = c\gamma^i$ if $n_1 \le i < n_2$, since $(1-c) \cdot 0 + c \cdot \gamma^i = c\gamma^i$; and a weight of $h_i = 0$ otherwise. We then minimize the error $\max_{i \ge 0} \left| h_i - (\gamma\lambda)^i \right|$.

We call our approximation Piecewise λ-Return because each weight h_i is a piecewise function whose value depends on where i lies in relation to the interval $[n_1, n_2)$. Figure 2 illustrates how Pilar roughly approximates the TD(λ) decay using a step-like shape. Although a Pilar's TD-error weights do not form a smooth curve, they retain important properties like contraction modulus, monotonicity, and variance reduction. Crucially, a Pilar is much cheaper to compute than a λ-return, making it more suitable for minibatch experience replay. In Appendix D, we describe a basic search algorithm for finding the lowest-error (n_1, n_2)-pair, along with a reference table of precomputed Pilars for $\gamma = 0.99$.
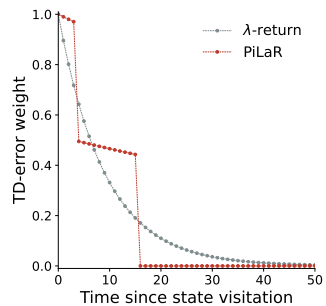
5. Deep RL Experiments

We consider a multistep generalization of Deep Q-Network (DQN; Mnih et al., 2015). The action-value function $q(s, a; \theta)$ is implemented as a neural network to enable generalization over high-dimensional states, where $\theta \in \mathbb{R}^d$ is the learnable parameters. A stale copy θ^- of the parameters is used only for bootstrapping and is infrequently updated from θ in order to stabilize learning. The agent interacts with its environment and stores each transition (s, a, r, s')—where s is the state, a is the action, r is the reward, and s' is the next state—in a replay memory $\mathcal{D}$. The network's loss function is defined as

$$\mathcal{L}(\theta, \theta^-) \overset{\text{def}}{=} \mathbb{E}\left[\frac{1}{2}\left(\hat{G}(r, s', \ldots; \theta^-) - q(s, a; \theta)\right)^2\right],$$

where $\hat{G}(r, s', \ldots; \theta^-)$ is a return estimator for $q_*(s, a)$ and the expectation is taken over a uniform distribution on $\mathcal{D}$. The network is trained by minimizing this loss via stochastic gradient descent on sampled minibatches.

We test our agents in all five MinAtar games (Young & Tian, 2019): Asterix, Breakout, Freeway, Seaquest, and Space Invaders. The

Table 2. Pilars used for MinAtar.

n-step	n_1	n_2	c
3	1	6	0.406
5	2	9	0.437

states are 10×10 multi-channel images depicting object locations and velocity trails. Each agent's network is a two-layer convolutional architecture with rectified linear units (ReLUs). The agents execute an ϵ-greedy policy for 5M time steps (where ϵ is linearly annealed from 1 to 0.1 over the first 100k steps) and conduct a minibatch update of 32 samples on every step. We provide more details in Appendix E. Code is available online.[2]

For $n = 3$ and $n = 5$, we compare the n-step return against the corresponding Pilar of the same contraction modulus with the given discount factor, $\gamma = 0.99$ (see Table 2 for specific values of n_1, n_2, and c). We chose five Adam step sizes over a logarithmic grid search and generated learning curves by plotting the 100-episode moving average of undiscounted return (game score) versus time steps. We then chose the step size for each game-estimator pair that maximized the area under the curve over the last 1M steps, or 20% of training. We showcase the best results for each game in Figure 3 but include all of the learning curves in Appendix E. Out of the five games, Seaquest was the only one in which Pilar did not outperform n-step returns in a statistically significant way. The results are averaged over 32 trials, with shading to indicate 95% confidence intervals.

In several cases, Pilars are able to significantly improve the agent's best performance compared to n-step returns. This

[2] https://github.com/brett-daley/averaging-nstep-returns

is in spite of the fact that each pair of n-step return and Pilar have the same contraction modulus, suggesting that the performance increase is partially due to variance reduction. Overall, Pilars outperform n-step returns for 1 in 5 games (20%) when $n = 3$, and 3 in 5 games (60%) when $n = 5$. In the remaining cases, the performance is not significantly different; thus, Pilars improve average performance in these games. Because the average performance gap between the estimators also widens as n increases, Pilar's benefits appears to become more pronounced for longer—and, hence, higher-variance—n-step returns.

These results corroborate our theory by showing that averaging n-step returns can accelerate learning, even with a nontrivial network architecture. We do not claim that Pilars are necessarily the best average to achieve variance reduction, but our experiments still demonstrate the practical relevance of such compound returns.

PPO To demonstrate that our theory also applies to on-policy methods such as Proximal Policy Optimization (PPO; Schulman et al., 2017), we conduct additional experiments in three MuJoCo environments (Todorov et al., 2012): Half Cheetah, Hopper, and Walker 2D. Because on-policy agents like PPO are trained on relatively short trajectories of recent experiences, it is feasible to compute exact λ-returns for every experience, often referred to as Generalized Advantage Estimation (GAE; Schulman et al., 2015b) in this context. This means that an approximation like Pilar is not necessary in this setting and we can directly compare λ-returns with n-step returns.

We use the CleanRL implementation of PPO (Huang et al., 2022). We compare different returns with the values $(n, \lambda) \in (5, 0.804), (10, 0.905), (20, 0.955)$. These pairs were chosen because the respective λ-returns and n-step returns have the same contraction moduli when $\gamma = 0.99$. We specifically include $n = 20$ because $\lambda = 0.95$ is a common default value for PPO. We showcase the best results in Figure 4 but include all of the learning curves in Appendix E. The results are averaged over 100 trials, with shading to indicate 95% confidence intervals. In seven out of nine of the cases, λ-returns significantly improve sample efficiency over n-step returns with respect to the 100-episode moving average of undiscounted return, again demonstrating the benefits of compound returns.

6. Conclusion

We have shown that compound returns, including λ-returns, have a variance-reduction property. This is the first evidence, to our knowledge, that λ-returns have a theoretical learning advantage over n-step returns in the absence of function approximation; it was previously believed that both were different yet equivalent ways of interpolating between TD and Monte Carlo learning. Our random-walk
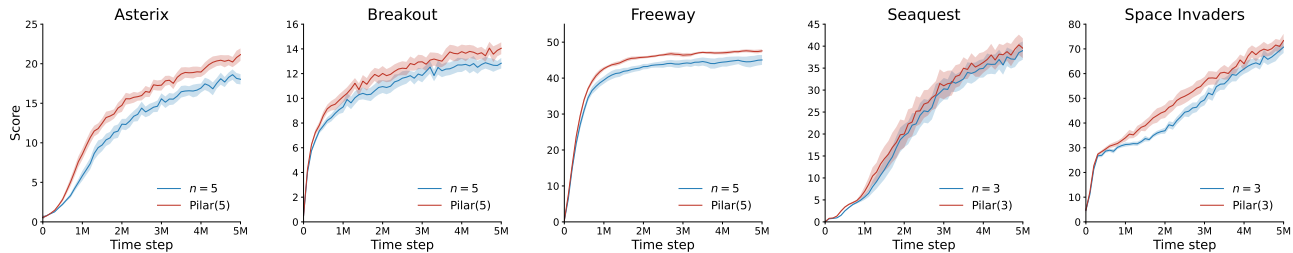
Figure 3. Learning curves for DQN with n-step returns and Pilars in five MinAtar games.
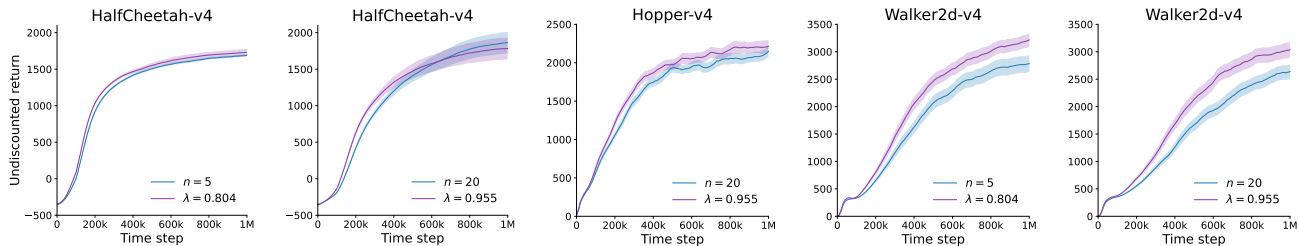

Figure 4. Learning curves for PPO with n-step returns and λ-returns in three MuJoCo environments.

experiments confirm that an appropriately chosen λ-return performs better than a given n-step return across a range of step sizes. In replay-based deep RL methods like DQN, where λ-returns are difficult to implement efficiently, we demonstrated with Pilar that a simpler average is still able to train neural networks with fewer samples. Since the average is formed from only two n-step returns, the additional computational cost is negligible—less expensive than adding a second target network, as is often done in recent methods (e.g., Fujimoto et al., 2018; Haarnoja et al., 2018).

Although we are able to establish strong theoretical guarantees regarding the convergence rate of multistep TD learning with linear function approximation, we note that this result does not automatically extend to nonlinear approximators. This is not a limitation of our analysis, but rather of the class of TD algorithms commonly used in deep RL. Indeed, even 1-step TD does not converge with certain nonlinear function approximators (see, e.g., Tsitsiklis & Van Roy, 1997, Sec. 10). Although our experiments have shown that compound returns often do improve performance for DQN and PPO, it is not guaranteed that these results will generalize to other methods. Additionally, we did not show that the improved performance for these algorithms is due solely to variance reduction; it is possible that other favorable properties of the compound returns also contributed to this improvement, such as convergence to a different fixed point (despite the bound that we proved in Proposition 3.10) or, as we show in concurrent work, the long-tailed credit-assignment characteristic of certain compound returns (Daley et al., 2024). The added complexities of nonconvex optimization likely explain why the performance trends for DQN and PPO are not always as clear as our random-walk results. Nevertheless, we note that the variance-reduction

property *is* a general phenomenon, since the TD-error assumptions are agnostic to the choice of function approximator. This property, coupled with our empirical results, indicates potential for using compound returns in other deep RL methods to improve sample efficiency.

A number of interesting extensions to our work are possible. For instance, we derived Pilar under the assumption that the λ-return is a good estimator to approximate. However, the exponential decay of the λ-return originally arose from the need for an efficient online update rule using eligibility traces, and is not necessarily optimal in terms of the bias-variance trade-off. With experience replay, we are free to average n-step returns in any way we want, even if the average would not be easily implemented online. This opens up exciting possibilities for new families of return estimators: e.g., those that minimize variance for a given contraction modulus or COM. Based on our compound variance model (Proposition 3.5), a promising direction in this regard appears to be weights that initially decay faster than the exponential function, but then slower afterwards. Minimizing variance becomes even more important for off-policy learning, where the inclusion of importance-sampling ratios greatly exacerbates variance. Recent works (Munos et al., 2016; Daley et al., 2023) have expressed arbitrary off-policy corrections in terms of weighted sums of TD errors, and so our theory could be extended to this setting with only minor modifications.

Acknowledgments

9

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. 1996.

Bhandari, J., Russo, D., and Singal, R. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on Learning Theory (COLT)*, 2018.

Chebotar, Y., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., Gopalakrishnan, K., et al. Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-functions. In *Conference on Robot Learning (CoRL)*, 2023.

Chung, W., Thomas, V., Machado, M. C., and Le Roux, N. Beyond variance reduction: Understanding the true impact of baselines on policy optimization. In *International Conference on Machine Learning (ICML)*, 2021.

Daley, B. and Amato, C. Reconciling λ-returns with experience replay. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Daley, B., White, M., Amato, C., and Machado, M. C. Trajectory-aware eligibility traces for off-policy reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Daley, B., Machado, M. C., and White, M. Demystifying the recency heuristic in temporal-difference learning. In *Reinforcement Learning Conference (RLC)*, 2024.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Harb, J. and Precup, D. Investigating recurrence and eligibility traces in deep Q-networks. In *NeurIPS Deep Reinforcement Learning Workshop*, 2016.

Hernandez-Garcia, J. F. and Sutton, R. S. Understanding multi-step deep reinforcement learning: A systematic study of the DQN target. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Kahn, H. and Harris, T. E. Estimation of particle transmission by random sampling. *National Bureau of Standards: Applied Mathematics Series*, 12:27–30, 1951.

Kearns, M. J. and Singh, S. Bias-variance error bounds for temporal difference updates. In *Conference on Learning Theory (COLT)*, 2000.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Representation Learning (ICLR)*, 2015.

Konidaris, G., Niekum, S., and Thomas, P. S. TD$_\gamma$: Re-evaluating complex backups in temporal difference learning. In *Neural Information Processing Systems (NeurIPS)*, 2011.

Kozuno, T., Tang, Y., Rowland, M., Munos, R., Kapturowski, S., Dabney, W., Valko, M., and Abel, D. Revisiting Peng's Q(λ) for modern reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

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

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

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In

International Conference on Machine Learning (ICML), 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Peng, J. and Williams, R. J. Incremental multi-step Q-learning. *Machine Learning*, 22:283–290, 1996.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

Rummery, G. A. and Niranjan, M. On-line Q-Learning using connectionist systems. Technical report, University of Cambridge, 1994.

Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach.* 2009.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2015b.

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

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning (ICML)*, 2023.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

Sutton, R. S. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts Amherst, 1984.

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

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction.* 2018.

Thomas, P. S., Niekum, S., Theocharous, G., and Konidaris, G. Policy evaluation using the Ω-return. *Neural Information Processing Systems (NeurIPS)*, 2015.

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

Tsitsiklis, J. N. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5): 674–690, 1997.

Watkins, C. J. C. H. *Learning from Delayed Rewards.* PhD thesis, University of Cambridge, 1989.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Young, K. and Tian, T. MinAtar: An Atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv*, 1903.03176, 2019.

A. Variance Assumptions

Our variance assumptions are a significant relaxation and generalization of the assumptions made by the n-step variance model of Konidaris et al. (2011, Sec. 3). We show this by starting from their original assumptions and then describing the steps taken to obtain our new assumptions.

Konidaris et al. begin by expanding the variance of an n-step return in the following way:

$$\text{Var}[G_t^{(n)} \mid S_t] = \text{Var}[G_t^{(n-1)} + \gamma^{n-1}\delta_{t+n-1} \mid S_t]$$
$$= \text{Var}[G_t^{(n-1)} \mid S_t] + \gamma^{2(n-1)}\text{Var}[\delta_{t+n-1} \mid S_t] + 2\,\text{Cov}[G_t^{(n-1)}, \gamma^{n-1}\delta_{t+n-1} \mid S_t]. \quad (9)$$

The covariance term is equivalent to

$$\text{Cov}[G_t^{(n-1)}, \gamma^{n-1}\delta_{t+n-1} \mid S_t] = \text{Cov}[G_t^{(n-1)}, G_t^{(n)} - G_t^{(n-1)} \mid S_t]$$
$$= \text{Cov}[G_t^{(n-1)}, G_t^{(n)} \mid S_t] - \text{Cov}[G_t^{(n-1)}, G_t^{(n-1)} \mid S_t]$$
$$= \text{Cov}[G_t^{(n-1)}, G_t^{(n)} \mid S_t] - \text{Var}[G_t^{(n-1)} \mid S_t]. \quad (10)$$

With the rationale that $G_t^{(n-1)}$ and $G_t^{(n)}$ are generated from the same trajectory and therefore highly correlated, the authors assume that
$$\text{Cov}[G_t^{(n-1)}, G_t^{(n)} \mid S_t] \approx \text{Var}[G_t^{(n-1)} \mid S_t], \quad (11)$$

which makes Eq. (10) approximately zero. Consequently, Eq. (9) becomes

$$\text{Var}[G_t^{(n)} \mid S_t] \approx \text{Var}[G_t^{(n-1)} \mid S_t] + \gamma^{2(n-1)}\text{Var}[\delta_{t+n-1} \mid S_t]. \quad (12)$$

The authors additionally assume that the variance of each TD error is the same:

Assumption 3.1. Each TD error has uniform variance: $\text{Var}[\delta_{t+i} \mid S_t] = \kappa, \ \forall\, i \geq 0$.

Hence, Eq. (12) becomes $\text{Var}[G_t^{(n)} \mid S_t] \approx \text{Var}[G_t^{(n-1)} \mid S_t] + \gamma^{2(n-1)}\kappa$. Since $\kappa = \text{Var}[\delta_t \mid S_t] = \text{Var}[G_t^{(1)} \mid S_t]$, unrolling the recursion gives the final n-step variance model:

$$\text{Var}[G_t^{(n)} \mid S_t] \approx \sum_{i=0}^{n-1} \gamma^{2i}\kappa. \quad (13)$$

This completes the derivation from Konidaris et al. (2011, Sec. 3), where the two major assumptions are Eq. (11) and Assumption 3.1. Notice, however, that Eq. (13) could be obtained more simply by assuming that TD errors are uncorrelated—in fact, this is equivalent to assuming Eq. (11) holds. One way to see this is by decomposing the n-step return into a sum of TD errors and applying standard variance rules, assuming no correlations between the random variables:

$$\text{Var}[G_t^{(n)} \mid S_t] = \text{Var}[G_t^{(n)} - v(S_t) \mid S_t] = \text{Var}\left[\sum_{i=0}^{n-1}\gamma^i\delta_{t+i} \,\middle|\, S_t\right] = \sum_{i=0}^{n-1}\gamma^{2i}\text{Var}[\delta_{t+i} \mid S_t] = \sum_{i=0}^{n-1}\gamma^{2i}\kappa,$$

which is identical to Eq. (13). In our work, we directly make this uniform-correlation assumption, while also generalizing it with an arbitrary correlation coefficient $\rho \in [0, 1]$. Note that we must have $\rho \geq 0$ because an infinite number of TD errors cannot be negatively correlated simultaneously.

Assumption 3.2. TD errors are uniformly correlated: $\text{Corr}[\delta_{t+i}, \delta_{t+j} \mid S_t] = \rho, \ \forall\, i \geq 0, \forall\, j \neq i$.

To summarize, our variance model makes Assumptions 3.1 and 3.2, which are equivalent to the assumptions made by Konidaris et al. (2011) when $\rho = 0$. Since $\text{Var}[\delta_{t+i} \mid S_t] = \text{Cov}[\delta_{t+i}, \delta_{t+i} \mid S_t]$, we are able to combine Assumptions 3.1 and 3.2 into a single, concise expression—see Eq. (5) in Section 3.

Although the assumptions here may not always hold in practice, they would be difficult to improve without invoking information about the MDP's transition function or reward function. We show in Appendix B that our derived variances based on these assumptions are nevertheless consistent with empirical data in real environments.

B. How Realistic Is the Proposed Variance Model?

Our assumptions state that all TD errors have uniform variance and are equally correlated to one another. Since these assumptions may be violated in practice, it is informative to test how well our n-step variance model (Proposition 3.3) compares to the true n-step variances in several examples.

We consider three environments: the 19-state random walk (see Section 3), a 4×3 gridworld (Russell & Norvig, 2009, Fig. 17.1), and a 10×8 gridworld (Sutton & Barto, 2018, Fig. 7.4). We choose these environments because they have known dynamics and are small enough to exactly calculate v_π with dynamic programming. The two gridworlds are stochastic because each of the four moves (up, down, left, right) succeeds with only probability 80%; otherwise, the move is rotated by 90 degrees in either direction with probability 10% each. We let the agent execute a uniform-random behavior policy for all environments.

To make the results agnostic to any particular learning algorithm, we use v_π to compute the TD errors. We apply a discount factor of $\gamma = 0.99$ to the 10×8 gridworld (otherwise $v_\pi(s)$ would be constant for all s due to the single nonzero reward) and leave the other two environments undiscounted. We then measure the variance of the n-step returns originating from the initial state of each environment, for $n \in \{1, \ldots, 21\}$. Figure 5 shows these variances plotted as a function of n and averaged over 10k episodes. The best-case (optimistic, $\rho = 0$) and worst-case (pessimistic, $\rho = 1$) variances predicted by the n-step model, assuming that $\kappa = \mathrm{Var}[\delta_0 \mid S_0]$, are also indicated by dashed lines.

For all of the environments, the measured n-step variances always remain within the lower and upper bounds predicted by Proposition 3.3. These results show that our n-step variance model can still make useful variance predictions even when our assumptions do not hold. The variances also grow roughly linearly as a function of n, corresponding more closely to the linear behavior of the optimistic, uncorrelated case than the quadratic behavior of the pessimistic, maximally correlated case. This further suggests that the majority of TD-error pairs are weakly correlated in practice, which makes sense because temporally distant pairs are unlikely to be strongly related.
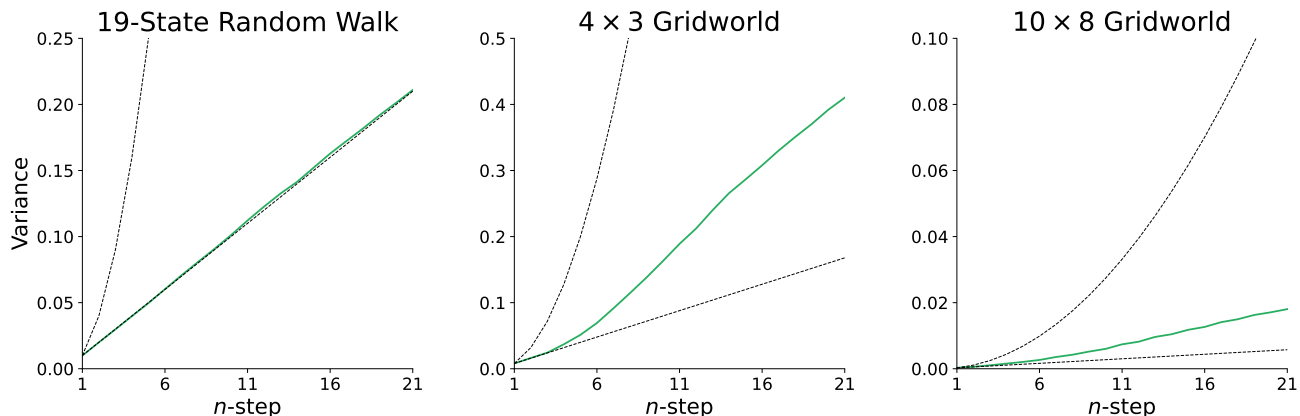

Figure 5. Variances of the n-step returns originating from the initial state in three environments. The solid green line indicates the true variance while the dashed black lines indicate the lower and upper bounds predicted by our n-step variance model (Proposition 3.3).

C. Proofs

In this section, we include all proofs that were omitted from the main paper due to space constraints.

C.1. Proof of Proposition 3.3

Proposition 3.3. *The variance of an n-step return is*

$$\mathrm{Var}[G_t^{(n)} \mid S_t] = (1-\rho)\,\Gamma_2(n)\,\kappa + \rho\,\Gamma_1(n)^2\,\kappa\,.$$

Proof. The n-step error can be expressed as a finite summation of TD errors:

$$G_t^{(n)} - v(S_t) = \sum_{i=0}^{n-1} \gamma^i \delta_{t+i}\,. \tag{14}$$

Using this, we calculate the covariance between two n-step returns with lengths n_1, n_2:

$$
\begin{aligned}
\mathrm{Cov}[G_t^{(n_1)}, G_t^{(n_2)} \mid S_t] &= \mathrm{Cov}\left[\sum_{i=0}^{n_1-1} \gamma^i \delta_{t+i}, \sum_{j=0}^{n_2-1} \gamma^j \delta_{t+j} \,\middle|\, S_t\right] \\
&= \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \mathrm{Cov}[\gamma^i \delta_{t+i}, \gamma^j \delta_{t+j} \mid S_t] \\
&= \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \gamma^{i+j} \mathrm{Cov}[\delta_{t+i}, \delta_{t+j} \mid S_t] \\
&= \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \gamma^{i+j}\big((1-\rho)\mathbf{1}_{i=j} + \rho\big)\kappa \\
&= (1-\rho)\sum_{i=0}^{\min(n_1,n_2)-1} \gamma^{2i}\kappa + \rho\sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} \gamma^{i+j}\kappa \\
&= (1-\rho)\sum_{i=0}^{\min(n_1,n_2)-1} \gamma^{2i}\kappa + \rho\sum_{i=0}^{n_1-1} \gamma^i \sum_{j=0}^{n_2-1} \gamma^j\kappa \\
&= (1-\rho)\,\Gamma_2(\min(n_1,n_2))\,\kappa + \rho\,\Gamma_1(n_1)\,\Gamma_1(n_2)\,\kappa\,.
\end{aligned}
\tag{15}
$$

Because $\mathrm{Var}[G_t^{(n)} \mid S_t] = \mathrm{Cov}[G_t^{(n)}, G_t^{(n)} \mid S_t]$, then by letting $n_1 = n_2 = n$, we obtain the n-step variance formula and the proof is complete. $\qquad\square$

C.2. Proof of Lemma 3.4

Lemma 3.4. *A compound error can be written as a weighted summation of TD errors:*

$$G_t^{\mathbf{c}} - v(S_t) = \sum_{i=0}^{\infty} \gamma^i h_i \delta_{t+i}\,, \text{ where } h_i \stackrel{\text{def}}{=} \sum_{n=i+1}^{\infty} c_n\,.$$

Proof. We decompose the compound error into a weighted average of n-step errors, and then decompose those n-step errors into weighted sums of TD errors using Eq. (14):

$$
\begin{aligned}
G_t^{\mathbf{c}} - v(S_t) &= \left(\sum_{n=1}^{\infty} c_n G_t^{(n)}\right) - v(S_t) \\
&= \sum_{n=1}^{\infty} c_n \left(G_t^{(n)} - v(S_t)\right)
\end{aligned}
$$

$$= \sum_{n=1}^{\infty} c_n \sum_{i=0}^{n-1} \gamma^i \delta_{t+i}$$

$$= c_1 \delta_t + c_2 (\delta_t + \gamma \delta_{t+1}) + c_3 (\delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2}) + \dots$$

$$= (c_1 + c_2 + \dots) \delta_t + \gamma (c_2 + c_3 + \dots) \delta_{t+1} + \gamma^2 (c_3 + c_4 + \dots) \delta_{t+2} + \dots$$

$$= h_0 \delta_t + \gamma h_1 \delta_{t+1} + \gamma^2 h_2 \delta_{t+2} + \dots$$

$$= \sum_{i=0}^{\infty} \gamma^i h_i \delta_{t+i},$$

which completes the lemma. $\qquad\square$

C.3. Proof of Proposition 3.5

Proposition 3.5. *The variance of a compound return is*

$$\text{Var}[G_t^c \mid S_t] = (1-\rho) \sum_{i=0}^{\infty} \gamma^{2i} h_i^2 \kappa + \rho \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} h_i h_j \kappa.$$

Proof. From Lemma 3.4, the variance of the compound return is

$$\text{Var}\left[G_t^c \mid S_t\right] = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \text{Cov}[\gamma^i h_i \delta_{t+i}, \gamma^j h_j \delta_{t+j} \mid S_t]$$

$$= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} h_i h_j \text{Cov}[\delta_{t+i}, \delta_{t+j} \mid S_t]$$

$$= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} h_i h_j ((1-\rho) \mathbf{1}_{i=j} + \rho) \kappa$$

$$= (1-\rho) \sum_{i=0}^{\infty} \gamma^{2i} h_i^2 \kappa + \rho \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} h_i h_j \kappa,$$

which establishes Proposition 3.5. $\qquad\square$

C.4. λ-return Variance

We calculate the variance of the λ-return under our assumptions. In the main text, we showed that the λ-return assigns a cumulative weight of $h_i = \lambda^i$ to the TD error at time $t + i$, which is also known from the TD(λ) algorithm. We can therefore apply Proposition 3.5 to obtain the following variance expression:

$$\text{Var}[G_t^\lambda \mid S_t] = (1-\rho) \sum_{i=0}^{\infty} (\gamma\lambda)^{2i} \kappa + \rho \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} (\gamma\lambda)^{i+j} \kappa$$

$$= (1-\rho) \sum_{i=0}^{\infty} (\gamma\lambda)^{2i} \kappa + \rho \sum_{i=0}^{\infty} (\gamma\lambda)^i \sum_{j=0}^{\infty} (\gamma\lambda)^j \kappa$$

$$= \frac{(1-\rho)\kappa}{1 - (\gamma\lambda)^2} + \frac{\rho\kappa}{(1 - \gamma\lambda)^2}. \tag{16}$$

C.5. Proof of Proposition 3.6

Proposition 3.6 (Effective n-step of compound return). *Let G_t^c be any compound return and let*

$$n = \begin{cases} \log_\gamma \left(\sum_{k=1}^{\infty} c_k \gamma^k \right), & \text{if } 0 < \gamma < 1, \\ \sum_{k=1}^{\infty} c_k k, & \text{if } \gamma = 1. \end{cases} \tag{7}$$

When n is an integer, G_t^c shares the same bound in Eq. (6) as the n-step return $G_t^{(n)}$.

Proof. When $\gamma < 1$, we can take the logarithm of both sides of $\gamma^n = \sum_{k=1}^{\infty} c_k \gamma^k$ to get

$$n = \log_\gamma \left(\sum_{k=1}^{\infty} c_k \gamma^k \right) = \frac{\log \left(\sum_{n=1}^{\infty} c_n \gamma^n \right)}{\log \gamma}.$$

For the undiscounted case, we would like to evaluate this expression at $\gamma = 1$; however, since $\sum_{n=1}^{\infty} c_n = 1$ from the definition of a compound return, we arrive at an indeterminate form, $\frac{0}{0}$. Instead, we can apply L'Hôpital's rule to evaluate the limit as $\gamma \to 1$:

$$
\begin{aligned}
\lim_{\gamma \to 1} \frac{\log \left(\sum_{n=1}^{\infty} c_n \gamma^n \right)}{\log \gamma} &= \lim_{\gamma \to 1} \frac{\frac{d}{d\gamma} \log \left(\sum_{n=1}^{\infty} c_n \gamma^n \right)}{\frac{d}{d\gamma} \log \gamma} \\
&= \lim_{\gamma \to 1} \frac{\left(\sum_{n=1}^{\infty} c_n \gamma^{n-1} n \right) / \left(\sum_{n=1}^{\infty} c_n \gamma^n \right)}{1 / \gamma} \\
&= \lim_{\gamma \to 1} \frac{\sum_{n=1}^{\infty} c_n \gamma^n n}{\sum_{n=1}^{\infty} c_n \gamma^n} \\
&= \frac{\sum_{n=1}^{\infty} c_n n}{\sum_{n=1}^{\infty} c_n} \\
&= \sum_{n=1}^{\infty} c_n n,
\end{aligned}
$$

where the last step follows again from the fact that $\sum_{n=1}^{\infty} c_n = 1$. This establishes the case where $\gamma = 1$ and completes the proof. $\square$

C.6. Proof of Theorem 3.7

Theorem 3.7 (Variance-reduction property of compound returns). *Let $G_t^{(n)}$ be any n-step return and let $G_t^{\boldsymbol{c}}$ be any compound return with the same effective n-step: i.e., $\boldsymbol{c}$ satisfies Proposition 3.6. The inequality $\mathrm{Var}[G_t^{\boldsymbol{c}} \mid S_t] \leq \mathrm{Var}[G_t^{(n)} \mid S_t]$ always holds, and is strict whenever TD errors are not perfectly correlated ($\rho < 1$).*

Proof. Eq. (15) gives us an expression for the covariance between two n-step returns. We use this to derive an alternative formula for the variance of a compound return:

$$
\begin{aligned}
\mathrm{Var}[G_t^{\boldsymbol{c}} \mid S_t] &= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \mathrm{Cov}[c_i G_t^{(i)}, c_j G_t^{(j)} \mid S_t] \\
&= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \mathrm{Cov}[G_t^{(i)}, G_t^{(j)} \mid S_t] \\
&= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \left((1-\rho) \, \Gamma_2(\min(i,j)) + \rho \, \Gamma_1(i) \, \Gamma_1(j) \right) \kappa \\
&= (1-\rho) \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_2(\min(i,j)) \, \kappa + \rho \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_1(i) \, \Gamma_1(j) \, \kappa.
\end{aligned}
$$

We analyze both sums separately, starting with the first term. If we assume $\gamma < 1$ for now, then $\gamma^n = \sum_{i=1}^{\infty} c_i \gamma^i$ by Proposition 3.6. Further note that $\min(i,j) \leq (i+j)/2$, with the inequality strict if $i \neq j$. Because Γ_2 is monotonically increasing, it follows that

$$
\begin{aligned}
\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_2(\min(i,j)) &< \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_2 \left(\frac{i+j}{2} \right) \\
&= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \left(\frac{1 - \gamma^{i+j}}{1 - \gamma^2} \right)
\end{aligned}
$$

16

$$= \frac{1 - \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \gamma^{i+j}}{1 - \gamma^2}$$

$$= \frac{1 - \sum_{i=1}^{\infty} c_i \gamma^i \sum_{j=1}^{\infty} c_j \gamma^j}{1 - \gamma^2}$$

$$= \frac{1 - \gamma^{2n}}{1 - \gamma^2}$$

$$= \Gamma_2(n) \ .$$

The inequality is strict because at least two weights in c are nonzero by definition of a compound return, guaranteeing at least one element in the sum has $i \neq j$. If instead $\gamma = 1$, then $n = \sum_{i=1}^{\infty} c_i i$ by Proposition 3.6 and also $\Gamma_2(\min(i,j)) = \min(i,j)$. Therefore, by Jensen's inequality, we have

$$\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_2(\min(i,j)) = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \min(i,j)$$

$$< \min \left(\sum_{i=1}^{\infty} c_i i, \ \sum_{j=1}^{\infty} c_j j \right)$$

$$= \min(n, \ n)$$

$$= \Gamma_2(n) \ .$$

Again, the inequality is strict by definition of a compound return, so we conclude that

$$\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_2(\min(i,j)) < \Gamma_2(n), \quad \text{for } 0 < \gamma \leq 1 \, .$$

We now address the second term. We show that Γ_1 is invariant under a weighted average under our assumption that Eq. (7) holds. If $\gamma < 1$, then

$$\sum_{i=1}^{\infty} c_i \, \Gamma_1(i) = \sum_{i=1}^{\infty} c_i \left(\frac{1 - \gamma^i}{1 - \gamma} \right) = \frac{1 - \sum_{i=1}^{\infty} c_i \gamma^i}{1 - \gamma} = \frac{1 - \gamma^n}{1 - \gamma} = \Gamma_1(n) \ . \tag{17}$$

If $\gamma = 1$, then

$$\sum_{i=1}^{\infty} c_i \, \Gamma_1(i) = \sum_{i=1}^{\infty} c_i i = n = \Gamma_1(n) \ .$$

Thus, regardless of γ, the second term becomes

$$\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} c_i c_j \, \Gamma_1(i) \, \Gamma_1(j) = \sum_{i=1}^{\infty} c_i \, \Gamma_1(i) \sum_{j=1}^{\infty} c_j \, \Gamma_1(j) = \Gamma_1(n)^2 \ .$$

Putting everything together, we have so far shown that

$$\mathrm{Var}[G_t^c \mid S_t] \leq (1 - \rho) \, \Gamma_2(n) \, \kappa + \rho \, \Gamma_1(n)^2 \, \kappa \, ,$$

where the right-hand side is the n-step return variance given by Proposition 3.3. As we showed above, this inequality is strict whenever the first term is active, i.e., $\rho < 1$, which completes the proof. $\qquad \square$

C.7. Proof of Corollary 3.8

Corollary 3.8 (Variance reduction of λ-return). *The magnitude of variance reduction for a λ-return is bounded by*

$$0 \leq \mathrm{Var}[G_t^n \mid S_t] - \mathrm{Var}[G_t^\lambda \mid S_t] \leq (1 - \rho) \frac{\lambda}{1 - \lambda^2} \kappa \, .$$

This magnitude is monotonic in γ and maximized at $\gamma = 1$.

Proof. First, the left-hand side of the inequality follows immediately from Theorem 3.7.

For the right-hand side, we next make the observation that, for any compound return,

$$\sum_{i=0}^{\infty} \gamma^i h_i = \sum_{i=0}^{\infty} \sum_{n=i+1}^{\infty} \gamma^i c_n = \sum_{n=1}^{\infty} \sum_{i=0}^{n-1} \gamma^i c_n = \sum_{n=1}^{\infty} c_n \frac{1-\gamma^n}{1-\gamma} = \frac{1 - \sum_{n=1}^{\infty} c_n \gamma^n}{1-\gamma} = \frac{1-\beta}{1-\gamma}.$$

This allows us to rewrite the variance model in Proposition 3.5 as

$$\mathrm{Var}[G_t^{\boldsymbol{c}} \mid S_t] = (1-\rho) \sum_{i=0}^{\infty} \gamma^{2i} h_i^2 \kappa + \rho \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} h_i h_j \kappa = (1-\rho) \sum_{i=0}^{\infty} \gamma^{2i} h_i^2 \kappa + \rho \left(\frac{1-\beta}{1-\gamma} \right)^2 \kappa.$$

The second term is identical for all returns with a fixed contraction modulus β, as is assumed in Theorem 3.7. Thus, by subtracting the variance of both returns, we get the following expression for variance reduction (positive means more variance reduction):

$$\mathrm{Var}[G_t^n \mid S_t] - \mathrm{Var}[G_t^{\boldsymbol{c}} \mid S_t] = (1-\rho) \left(\Gamma_2(n) - \sum_{i=0}^{\infty} \gamma^{2i} h_i^2 \right) \kappa.$$

This reveals that the variance reduction of a compound return is proportional to $1 - \rho$, affirming the result of Theorem 3.7 that variance reduction occurs whenever $\rho < 1$. Additionally, this variance reduction is proportional to the critical quantity $\Gamma_2(n) - \sum_{i=0}^{\infty} \gamma^{2i} h_i^2$, a function of the weights of the compound return. Substituting the weights for a λ-return, this critical quantity becomes

$$\Gamma_2(n) - \frac{1}{1-(\gamma\lambda)^2}.$$

This gap is monotonically increasing with γ and attains its maximum value as $\gamma \to 1$, i.e., the maximum-variance case of undiscounted rewards.[3] Taking the limit as $\gamma \to 1$ makes the possible variance reduction proportional to

$$n - \frac{1}{1-\lambda^2} = \frac{1}{1-\lambda} - \frac{1}{1-\lambda^2} = \frac{\lambda}{1-\lambda^2},$$

where the identity $n = 1 / (1 - \lambda)$ comes from Proposition 3.11, the effective n-step of the λ-return. This gives us the final upper bound:

$$\mathrm{Var}[G_t^n \mid S_t] - \mathrm{Var}[G_t^{\lambda} \mid S_t] \leq (1-\rho) \frac{\lambda}{1-\lambda^2} \kappa,$$

which completes the inequality. $\square$

C.8. Finite-Time Analysis

In this section, we prove Theorem 3.9 to establish a finite-time bound on the performance of multistep TD learning. We derive the bound in terms of the return's contraction modulus and variance, allowing us to invoke Theorem 3.7 and show an improved convergence rate.

At each iteration, TD learning updates the current parameters $\theta_t \in \mathbb{R}^d$ according to Eq. (8). A value-function estimate for any state s is obtained by evaluating the dot product of the parameters and the state's corresponding feature vector: $v_\theta(s) \overset{\text{def}}{=} \theta^\top \phi(s)$. Following Bhandari et al. (2018), we assume that $\|\phi(s)\|_2^2 \leq 1, \forall\, s \in \mathcal{S}$. This can be guaranteed in practice by normalizing the features and is therefore not a strong assumption.

In the prediction setting, the agent's behavior policy is fixed such that the MDP can be cast as a Markov reward process (MRP), where $r(s, s')$ denotes the expected reward earned when transitioning from state s to state s'. We adopt the i.i.d. state model from Bhandari et al. (2018, Sec. 3) and generalize it for multistep TD updates.

Assumption C.1 (i.i.d. state model). Assume the MRP under the policy π is ergodic. Let $\boldsymbol{d} \in \mathbb{R}^{|\mathcal{S}|}$ represent the MRP's unique stationary distribution. Each iteration of Eq. (8) is calculated by first sampling a random initial state $S_{t,0} \sim \boldsymbol{d}$ and then sampling a trajectory of subsequent states $S_{t,i+1} \sim \mathrm{Pr}(\cdot \mid S_{t,i}), \forall\, i \geq 0$.

---

[3] To see this, make the substitution $\lambda = (1 - \gamma^{n-1}) / (1 - \gamma^n)$ from Proposition 3.11 because the effective n-step is the same, and analyze the behavior as $\gamma \to 1$.

That is, a state $S_{t,0}$ sampled from the steady-state MRP forms the root node for the following trajectory $(S_{t,1}, S_{t,2}, \dots)$ that is generated according to the MRP transition function. Notably, this setting parallels the experience-replay setting utilized by many deep RL agents.

To facilitate our analysis, we decompose the compound TD updates into weighted averages of n-step TD updates, where each n-step update has the form

$$g_t^n(\theta) \stackrel{\text{def}}{=} \left(G_t^{(n)} - \phi(S_t)^\top \theta \right) \phi(S_t) \, .$$

This allows us to conveniently express a compound TD update as

$$g_t^{\boldsymbol{c}}(\theta) \stackrel{\text{def}}{=} \sum_{n=1}^{\infty} c_n g_t^n(\theta) \, .$$

Our proofs also make use of the *expected* n-step TD update:

$$\bar{g}^n(\theta) \stackrel{\text{def}}{=} \sum_{s_0 \in \mathcal{S}} \sum_{\tau \in \mathcal{S}^{n-1}} d(s_0) \Pr(\tau \mid s_0) \left(G^{(n)}(s_0, \tau, \theta) - \phi(s_0)^\top \theta \right) \phi(s_0)$$

where $(s_1, s_2, \dots) = \tau$ and $G^{(n)}(s_0, \tau, \theta) \stackrel{\text{def}}{=} r(s_0, s_1) + \cdots + \gamma^{n-1} r(s_{n-1}, s_n) + \gamma^n \phi(s_n)^\top \theta$ is the n-step return generated from (s_0, τ).

For brevity, let $R_i \stackrel{\text{def}}{=} r(S_{t,i}, S_{t,i+1})$ and $\phi_i \stackrel{\text{def}}{=} \phi(S_{t,i})$ be random variables sampled according to Assumption C.1. We more conveniently write the expected n-step TD update as

$$\bar{g}^n(\theta) = \mathbb{E}\left[\phi_0(R_0 + \gamma R_1 + \cdots + \gamma^{n-1} R_{n-1})\right] + \mathbb{E}\left[\phi_0(\gamma^n \phi_n - \phi_0)^\top\right] \theta \, . \tag{18}$$

The expected compound TD update easily follows as the weighted average

$$\bar{g}^{\boldsymbol{c}}(\theta) \stackrel{\text{def}}{=} \sum_{n=1}^{\infty} c_n \bar{g}^n(\theta) \, .$$

Finally, let θ^* be the fixed point of the compound TD update: i.e., $\bar{g}^{\boldsymbol{c}}(\theta^*) = 0$. This fixed point always exists and is unique because the projected Bellman operator is a contraction mapping (Tsitsiklis & Van Roy, 1997), and therefore so is any weighted average of the n-iterated operators.

Before we prove Theorem 3.9, we must introduce two lemmas. The first establishes a lower bound on the angle between the expected TD update and the true direction toward the fixed point.

Lemma C.2. *Define the diagonal matrix $\boldsymbol{D} \stackrel{\text{def}}{=} \text{diag}(\boldsymbol{d})$. For any $\theta \in \mathbb{R}^d$,*

$$(\theta^* - \theta)^\top \bar{g}^{\boldsymbol{c}}(\theta) \geq (1 - \beta) \|v_{\theta^*} - v_\theta\|_{\boldsymbol{D}}^2 \, . \tag{19}$$

Proof. Let $\xi_i \stackrel{\text{def}}{=} v_{\theta^*}(S_{t,i}) - v_\theta(S_{t,i}) = (\theta^* - \theta)^\top \phi_i$ for $i \geq 0$. By stationarity, each ξ_i is a correlated random variable with the same marginal distribution. Because $S_{t,0}$ is drawn from the stationary distribution, we have $\mathbb{E}[\xi_i^2] = \|v_{\theta^*} - v_\theta\|_{\boldsymbol{D}}^2$.

From Eq. (18), we show

$$\bar{g}^{\boldsymbol{c}}(\theta) = \bar{g}^{\boldsymbol{c}}(\theta) - \bar{g}^{\boldsymbol{c}}(\theta^*) = \sum_{n=1}^{\infty} c_n \mathbb{E}\left[\phi_0(\gamma^n \phi_n - \phi_0)^\top(\theta - \theta^*)\right]$$

$$= \sum_{n=1}^{\infty} c_n \mathbb{E}[\phi_0(\xi_0 - \gamma^n \xi_n)] \, .$$

It follows that

$$(\theta^* - \theta)^\top \bar{g}^{\boldsymbol{c}}(\theta) = \sum_{n=1}^{\infty} c_n \mathbb{E}[\xi_0(\xi_0 - \gamma^n \xi_n)]$$

$$= \mathbb{E}\big[\xi_0^2\big] - \sum_{n=1}^{\infty} c_n \gamma^n \mathbb{E}[\xi_0 \xi_n]$$

$$\geq \left(1 - \sum_{n=1}^{\infty} c_n \gamma^n\right) \mathbb{E}\big[\xi_0^2\big]$$

$$= (1 - \beta)\|v_{\theta^*} - v_\theta\|_{\boldsymbol{D}}^2 \,.$$

The inequality uses the Cauchy-Schwarz inequality along with the fact that every ξ_i has the same marginal distribution: thus, $\mathbb{E}[\xi_0 \xi_i] \leq \sqrt{\mathbb{E}[\xi_0^2]}\sqrt{\mathbb{E}[\xi_i^2]} = \mathbb{E}\big[\xi_0^2\big]$. □

The next lemma establishes a bound on the second moment of the squared norm of the TD update in terms of the contraction modulus β and the variance σ^2 of the compound return.

Lemma C.3. *Define* $\Delta^* \overset{\text{def}}{=} \|r\|_\infty + (1+\gamma)\|\theta^*\|_\infty$ *and* $C \overset{\text{def}}{=} \Delta^* / (1 - \gamma)$. *For any* $\theta \in \mathbb{R}^d$,

$$\mathbb{E}[\|g_t(\theta)\|_2^2] \leq 2(1-\beta)^2 C^2 + 2\sigma^2 + 4(1+\beta)\|v_{\theta^*} - v_\theta\|_{\boldsymbol{D}}^2 \,.$$

Proof. Let $\delta_{t,i}^* \overset{\text{def}}{=} R_i + \gamma \phi_{i+1}^\top \theta^* - \phi_i^\top \theta^*$ and note that $\big|\delta_{t,i}^*\big| \leq \Delta^*$ for all $i \geq 0$ by the triangle inequality and the bounded-feature assumption. Denote the n-step and compound errors constructed from θ^* by $\delta_t^{(n)} \overset{\text{def}}{=} \sum_{i=0}^{n-1} \gamma^i \delta_{t,i}^*$ and $\delta_t^c \overset{\text{def}}{=} \sum_{n=1}^{\infty} c_n \delta_t^{(n)}$, respectively. We have

$$\mathbb{E}\Big[\|g_t(\theta^*)\|_2^2\Big] = \mathbb{E}\Big[\|\delta_t^c \phi_0\|_2^2\Big] \leq \mathbb{E}\big[(\delta_t^c)^2\big] = \mathbb{E}[\delta_t^c]^2 + \sigma^2 \,, \tag{20}$$

where the inequality follows from the assumption that $\|\phi_0\|_2^2 \leq 1$. The absolute value of the expectation can be bounded using the triangle inequality:

$$\Big|\mathbb{E}[\delta_t^c]\Big| = \left|\mathbb{E}\left[\sum_{n=1}^{\infty} c_n \delta_t^{(n)}\right]\right| \leq \sum_{n=1}^{\infty} c_n \, \Gamma_1(n) \, \Delta^* = \frac{1-\beta}{1-\gamma} \Delta^* = (1-\beta)C \,. \tag{21}$$

The identity $\sum_{n=1}^{\infty} c_n \Gamma_1(n) = (1 - \beta) / (1 - \gamma)$ comes from Eq. (17). Eqs. (20) and (21) imply

$$\mathbb{E}\Big[\|g_t(\theta^*)\|_2^2\Big] \leq (1-\beta)^2 C^2 + \sigma^2 \,. \tag{22}$$

Recall that $\mathbb{E}\big[\xi_i^2\big] = \|v_{\theta^*} - v_\theta\|_{\boldsymbol{D}}^2$ for all $i \geq 0$. Next, we show

$$\mathbb{E}\Big[\|g_t(\theta) - g_t(\theta^*)\|_2^2\Big] = \mathbb{E}\left[\left\|\sum_{n=1}^{\infty} c_n \phi_0 (\gamma^n \phi_n - \phi_0)^\top (\theta - \theta^*)\right\|_2^2\right]$$

$$= \mathbb{E}\left[\left\|\sum_{n=1}^{\infty} c_n \phi_0 (\xi_0 - \gamma^n \xi_n)\right\|_2^2\right]$$

$$\leq \sum_{n=1}^{\infty} c_n \mathbb{E}\Big[\|\phi_0 (\xi_0 - \gamma^n \xi_n)\|_2^2\Big]$$

$$\leq \sum_{n=1}^{\infty} c_n \mathbb{E}\big[(\xi_0 - \gamma^n \xi_n)^2\big]$$

$$\leq 2 \sum_{n=1}^{\infty} c_n \left(\mathbb{E}\big[\xi_0^2\big] + \gamma^{2n} \mathbb{E}\big[\xi_n^2\big]\right)$$

$$= 2 \sum_{n=1}^{\infty} c_n (1 + \gamma^{2n}) \|v_{\theta^*} - v_\theta\|_{\boldsymbol{D}}^2$$

$$\leq 2 \sum_{n=1}^{\infty} c_n (1 + \gamma^n) \|v_{\theta^*} - v_\theta\|_D^2$$

$$= 2(1 + \beta) \|v_{\theta^*} - v_\theta\|_D^2. \tag{23}$$

The four inequalities respectively follow from Jensen's inequality, the bounded-feature assumption $\|\phi\|_2^2 \leq 1$, the triangle inequality, and the fact that $\gamma^{2n} \leq \gamma^n$. The final equality comes from the definition of the contraction modulus, the coefficient of the left-hand side of Eq. (6). Combining Eqs. (22) and (23) gives the final result:

$$\mathbb{E}\left[\|g_t(\theta)\|_2^2\right] \leq \mathbb{E}\left[(\|g_t(\theta^*)\|_2 + \|g_t(\theta) - g_t(\theta^*)\|_2)^2\right]$$

$$\leq 2\mathbb{E}\left[\|g_t(\theta^*)\|_2^2\right] + 2\mathbb{E}\left[\|g_t(\theta) - g_t(\theta^*)\|_2^2\right]$$

$$\leq 2(1-\beta)^2 C^2 + 2\sigma^2 + 4(1+\beta)\|v_{\theta^*} - v_\theta\|_D^2,$$

where the second inequality uses the algebraic identity $(x+y)^2 \leq 2x^2 + 2y^2$. $\qquad\square$

We are now ready to derive the finite-time bound. We restate Theorem 3.9 and then provide the proof.

Theorem 3.9 (Finite-Time Analysis)**.** *Suppose TD learning with linear function approximation is applied under an i.i.d. state model with stationary distribution $d \in \mathbb{R}^{|\mathcal{S}|}$ (see Assumption C.1 in Appendix C.8) using the compound return estimator G_t^c as its target. Let $\beta \in [0,1)$ be the contraction modulus of the estimator and let $\sigma^2 \geq 0$ be the variance of the estimator. Assume that the features are normalized such that $\|\phi(s)\|_2^2 \leq 1$, $\forall s \in \mathcal{S}$. Define $C \stackrel{\text{def}}{=} (\|r\|_\infty + (1+\gamma)\|\theta^*\|_\infty) / (1-\gamma)$, where θ^* is the minimizer of the projected Bellman error for G_t^c. For any $T \geq (4 / (1-\beta))^2$ and a constant step size $\alpha = 1 / \sqrt{T}$,*

$$\mathbb{E}\left[\|v_{\theta^*} - v_{\bar{\theta}_T}\|_D^2\right] \leq \frac{\|\theta^* - \theta_0\|_2^2 + 2(1-\beta)^2 C^2 + 2\sigma^2}{(1-\beta)\sqrt{T}},$$

where $D \stackrel{\text{def}}{=} \text{diag}(d)$ and $\bar{\theta}_T \stackrel{\text{def}}{=} \frac{1}{T} \sum_{t=0}^{T-1} \theta_t$.

Proof. TD learning updates the parameters according to Eq. (8). Therefore,

$$\|\theta^* - \theta_{t+1}\|_2^2 = \|\theta^* - \theta_t - \alpha \, g_t(\theta_t)\|_2^2$$

$$= \|\theta^* - \theta_t\|_2^2 - 2\alpha \, g_t(\theta_t)^\top (\theta^* - \theta_t) + \alpha^2 \|g_t(\theta)\|_2^2.$$

Taking the expectation and then applying Lemmas C.2 and C.3 gives

$$\mathbb{E}\left[\|\theta^* - \theta_{t+1}\|_2^2\right] = \mathbb{E}\left[\|\theta^* - \theta_t\|_2^2\right] - 2\alpha\mathbb{E}\left[g_t(\theta_t)^\top(\theta^* - \theta_t)\right] + \alpha^2\mathbb{E}\left[\|g_t(\theta)\|_2^2\right]$$

$$= \mathbb{E}\left[\|\theta^* - \theta_t\|_2^2\right] - 2\alpha\mathbb{E}\left[\mathbb{E}\left[g_t(\theta_t)^\top(\theta^* - \theta_t)\right] \mid \theta_t\right] + \alpha^2\mathbb{E}\left[\mathbb{E}\left[\|g_t(\theta)\|_2^2\right] \mid \theta_t\right]$$

$$\leq \mathbb{E}\left[\|\theta^* - \theta_t\|_2^2\right] - \left(2\alpha(1-\beta) - 4\alpha^2(1+\beta)\right)\|v_{\theta^*} - v_\theta\|_D^2 + 2\alpha^2\left((1-\beta)^2 C^2 + \sigma^2\right)$$

$$\leq \mathbb{E}\left[\|\theta^* - \theta_t\|_2^2\right] - \alpha(1-\beta)\|v_{\theta^*} - v_\theta\|_D^2 + 2\alpha^2\left((1-\beta)^2 C^2 + \sigma^2\right).$$

The first inequality is due to Lemmas C.2 and C.3, which are applicable due to the i.i.d. setting (because the trajectory influencing g_t is independent of θ_t). The second inequality follows from the assumption that $\alpha \leq (1-\beta) / 4$. Rearranging the above inequality gives us

$$\mathbb{E}\left[\|v_{\theta^*} - v_{\theta_t}\|_D^2\right] \leq \frac{\|\theta^* - \theta_t\|_2^2 - \|\theta^* - \theta_{t+1}\|_2^2 + 2\alpha^2\left((1-\beta)^2 C^2 + \sigma^2\right)}{\alpha(1-\beta)}.$$

Summing over T iterations and then invoking the assumption that $\alpha = 1 / \sqrt{T}$:

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\|v_{\theta^*} - v_{\theta_t}\|_D^2\right] \leq \frac{\|\theta^* - \theta_0\|_2^2 - \|\theta^* - \theta_T\|_2^2 + 2\alpha^2\left((1-\beta)^2 C^2 + \sigma^2\right)T}{\alpha(1-\beta)}$$

$$\leq \frac{\|\theta^* - \theta_0\|_2^2 + 2\alpha^2 \left((1-\beta)^2 C^2 + \sigma^2\right) T}{\alpha(1-\beta)}$$

$$= \frac{\|\theta^* - \theta_0\|_2^2 \sqrt{T} + 2 \left((1-\beta)^2 C^2 + \sigma^2\right) \sqrt{T}}{1-\beta} .$$

We therefore conclude that

$$\mathbb{E}\left[\|v_{\theta^*} - v_{\bar{\theta}_T}\|_D^2\right] \leq \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\|v_{\theta^*} - v_{\theta_t}\|_D^2\right] \leq \frac{\|\theta^* - \theta_0\|_2^2 + 2(1-\beta)^2 C^2 + 2\sigma^2}{(1-\beta)\sqrt{T}},$$

which completes the bound. $\qquad\square$

C.9. Proof of Proposition 3.10

Proposition 3.10 (TD Solution Quality). *Let θ^* be the minimizer of the projected Bellman error under linear function approximation for any n-step or compound return with contraction modulus β. The following bound always holds:*

$$\|\mathbf{\Phi}\theta^* - v_\pi\|_D \leq \frac{1}{1-\beta} \|\mathbf{\Pi}v_\pi - v_\pi\|_D .$$

Proof. Our proof generalizes the bound for the λ-return operator given by Tsitsiklis & Van Roy (1997, Lemma 6). Let $T_\pi : v \mapsto r + \gamma \boldsymbol{P}_\pi v$ be the Bellman operator, where $\boldsymbol{P}_\pi$ is the stochastic transition matrix of the MDP under policy π. The n-step Bellman operator is denoted by T_π^n, where $T_\pi^n v \stackrel{\text{def}}{=} T_\pi(T_\pi^{n-1}v)$ and $T_\pi^0 v \stackrel{\text{def}}{=} v$.

A compound return corresponds to the operator $T_\pi^{(\boldsymbol{c})} : v \mapsto \sum_{n=1}^\infty c_n T_\pi^n v$. This operator is a contraction mapping since it is a convex combination of contraction mappings. Let β be the contraction modulus of $T_\pi^{(\boldsymbol{c})}$ for the given weights, $\boldsymbol{c}$. The compound TD procedure in Eq. (1) converges to a fixed point θ^* which is the unique solution of the following projected Bellman equation:

$$\mathbf{\Pi}T_\pi^{(\boldsymbol{c})}(\mathbf{\Phi}\theta^*) = \mathbf{\Phi}\theta^* ,$$

where $\mathbf{\Pi} = \mathbf{\Phi}(\mathbf{\Phi}^\top D \mathbf{\Phi})^{-1}\mathbf{\Phi}^\top D$ is the linear projection operator. In their proof of Lemma 6, Tsitsiklis & Van Roy show that $\mathbf{\Pi}$ is nonexpansive with respect to the norm $\|\cdot\|_D$. Therefore, for any compound return, it follows that

$$\begin{aligned}
\|\mathbf{\Phi}\theta^* - v_\pi\|_D &\leq \|\mathbf{\Phi}\theta^* - \mathbf{\Pi}v_\pi\|_D + \|\mathbf{\Pi}v_\pi - v_\pi\|_D \\
&= \left\|\mathbf{\Pi}T_\pi^{(\boldsymbol{c})}(\mathbf{\Phi}\theta^*) - \mathbf{\Pi}v_\pi\right\|_D + \|\mathbf{\Pi}v_\pi - v_\pi\|_D \\
&\leq \left\|T_\pi^{(\boldsymbol{c})}(\mathbf{\Phi}\theta^*) - v_\pi\right\|_D + \|\mathbf{\Pi}v_\pi - v_\pi\|_D \\
&\leq \beta\|\mathbf{\Phi}\theta^* - v_\pi\|_D + \|\mathbf{\Pi}v_\pi - v_\pi\|_D
\end{aligned}$$

Solving the inequality for $\|\mathbf{\Phi}\theta^* - v_\pi\|_D$ gives the final bound. $\qquad\square$

C.10. Proof of Proposition 3.11

Proposition 3.11 (Effective λ of n-step return). *For any $n \geq 1$, when $\gamma < 1$, the λ-return with $\lambda = (1 - \gamma^{n-1})/(1 - \gamma^n)$ has the same contraction modulus as the n-step return. When $\gamma = 1$, the λ-return with $\lambda = (n-1)/n$ has the same COM as the n-step return.*

Proof. **Case $\gamma < 1$:** We substitute $c_n = (1-\lambda)\lambda^{n-1}$ into the weighted average in Eq. (6) to compute the contraction modulus of the λ-return:

$$\sum_{n=1}^\infty c_n \gamma^n = \sum_{n=1}^\infty (1-\lambda)\lambda^{n-1}\gamma^n = \gamma(1-\lambda) \sum_{n=1}^\infty (\gamma\lambda)^{n-1} = \frac{\gamma(1-\lambda)}{1-\gamma\lambda} .$$

We therefore seek λ such that

$$\frac{\gamma(1-\lambda)}{1-\gamma\lambda} = \gamma^n$$

in order to equate the λ-return's contraction modulus to that of the given n-step return. We multiply both sides of the equation by $1 - \gamma\lambda$ and isolate λ to complete the case:

$$\gamma(1 - \lambda) = \gamma^n(1 - \gamma\lambda)$$
$$1 - \lambda = \gamma^{n-1}(1 - \gamma\lambda)$$
$$1 - \lambda = \gamma^{n-1} - \gamma^n\lambda$$
$$\gamma^n\lambda - \lambda = \gamma^{n-1} - 1$$
$$\lambda(\gamma^n - 1) = \gamma^{n-1} - 1$$
$$\lambda = (1 - \gamma^{n-1}) / (1 - \gamma^n).$$

Case $\gamma = 1$: We use Proposition 3.6 with $c_n = (1 - \lambda)\lambda^{n-1}$ to compute the effective n-step of the λ-return:

$$n = \sum_{k=1}^{\infty}(1 - \lambda)\lambda^{k-1}k = (1 - \lambda)\sum_{k=1}^{\infty}\lambda^{k-1}k = (1 - \lambda)\frac{1}{(1 - \lambda)^2} = \frac{1}{1 - \lambda}.$$

Rearranging the equation $n = 1 / (1 - \lambda)$ for λ gives the final result of $\lambda = (n - 1) / n$. $\qquad\square$

D. Pilar: Piecewise λ-Return

We present a basic search algorithm for finding the corresponding Pilar for a given n-step return (see Algorithm 1). The algorithm accepts the desired effective n-step (which does not need to be an integer necessarily) as its only argument and returns the values (n_1, n_2, c) such that the two-bootstrap return $(1 - c)G_t^{(n_1)} + cG_t^{(n_2)}$ minimizes the maximum absolute difference between its cumulative weights and those of the λ-return with the same effective n-step. The algorithm proceeds as follows. For each $n_1 \in \{1, \ldots, \lfloor n \rfloor\}$, scan through $n_2 \in \{n_1 + 1, n_1 + 2, \ldots, \}$ until the error stops decreasing. Every time a better (n_1, n_2)-pair is found, record the values, and return the last recorded values upon termination. The resulting Pilar has the same contraction modulus as the targeted n-step return; thus, their error-reduction properties are the same, but the Pilar's variance is lower by Theorem 3.7. To modify the search algorithm for undiscounted rewards, we just need to change λ and c such that they equate the COMs—rather than the contraction moduli—of the two returns. We also include this case in Algorithm 1.

We populate Table 3 with corresponding Pilar values for several common n-step returns when $\gamma = 0.99$. A discount factor of $\gamma = 0.99$ is extremely common in deep RL, and so it is hoped that this table serves as a convenient reference that helps practitioners avoid redundant searches with Algorithm 1.

Table 3. n-step returns and Pilars with equal contraction moduli when $\gamma = 0.99$.

effective n-step	n_1	n_2	c
2	1	4	0.337
3	1	6	0.406
4	2	7	0.406
5	2	9	0.437
10	4	16	0.515
20	6	35	0.519
25	8	43	0.530
50	13	79	0.640
100	22	147	0.760

Algorithm 1 Pilar(n)

1: **require** $n \geq 1$, $\gamma \in (0, 1)$

2: $\lambda = \begin{cases} (1 - \gamma^{n-1}) / (1 - \gamma^n) & \text{if } \gamma < 1 \\ (n - 1) / n & \text{if } \gamma = 1 \end{cases}$

3: best_error $\leftarrow \infty$

4: **for** $n_1 = 1, \ldots, \lfloor n \rfloor$ **do**

5:     $n_2 \leftarrow \lfloor n \rfloor$

6:     error $\leftarrow \infty$

7:     **repeat**

8:         $n_2 \leftarrow n_2 + 1$

9:         $c = \begin{cases} (\gamma^n - \gamma^{n_1}) / (\gamma^{n_2} - \gamma^{n_1}) & \text{if } \gamma < 1 \\ (n - n_1) / (n_2 - n_1) & \text{if } \gamma = 1 \end{cases}$

10:         prev_error $\leftarrow$ error

11:         error $\leftarrow$ ERROR(λ, n_1, n_2, c)

12:         **if** error $<$ best_error **then**

13:             values $\leftarrow (n_1, n_2, c)$

14:             best_error $\leftarrow$ error

15:         **end if**

16:     **until** error $\geq$ prev_error

17: **end for**

18: **return** values

19: **function** ERROR(λ, n_1, n_2, c)

20:     Let $h_i = \begin{cases} \gamma^i & \text{if } i < n_1 \\ c\gamma^i & \text{else if } i < n_2 \\ 0 & \text{else} \end{cases}$

21:     **return** $\max_{i \geq 0} \left| h_i - (\gamma\lambda)^i \right|$

22: **end function**

E. Experiment Setup and Additional Results

Our DQN experiment procedure closely matches that of Young & Tian (2019). The only differences in our methodology are the mean-squared loss (rather than Huber loss), the Adam optimizer (Kingma & Ba, 2015), and multistep returns. MinAtar represents states as $10 \times 10 \times 7$ binary images. The agents process these with a convolutional network; the first layer is a 16-filter 3×3 convolutional layer, the output of which is flattened and then followed by a dense layer with 128 units. Both layers use ReLU activations.

The agents were trained for 5 million time steps each. They executed a random policy for the first 5k time steps to prepopulate the replay buffer (capacity: 100k transitions), and then switched to an ϵ-greedy policy for the remainder of training, with ϵ annealed linearly from 1 to 0.1 over the next 100k steps. Every step, the main network was updated using a minibatch of 32 return estimates to minimize the loss in Section 5. The target network's parameters were copied from the main network every 1k time steps.

To obtain the n-step returns, the replay buffer is modified to return a minibatch of sequences of $n + 1$ experiences for each return estimate (instead of the usual two experiences for DQN). The return is computed by summing the first n rewards and then adding the value-function bootstrap from the final experience, with discounting if $\gamma < 1$. If the episode terminates at any point within this trajectory, then the return is truncated and no bootstrapping is necessary, since the value of a terminal state is defined to be zero. For Pilars, the idea is the same, but the trajectories must have length $n_2 + 1$ to accommodate the lengths of both n-step returns. The two returns are computed as above, and then combined by averaging them: $(1 - c)G^{(n_1)} + cG^{(n_2)}$.

We show the learning curves for each return estimator using its best step size in Figure 6, where the step size was chosen from $\{10^{-5}, 3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$. We observe that Pilars perform no worse than n-step returns in the MinAtar domain, and sometimes significantly better. The relative improvement between Pilars and n-step returns tends to widen for $n = 5$, suggesting that Pilars are better at tolerating the higher variance in this setting.

In Figure 7, we also show similar learning curves for the PPO experiments. Other than the return estimators themselves, we use the default hyperparameters from the CleanRL implementation, which are similar to those of Schulman et al. (2017). The network architectures for the actor and critic are 2-layer, 64-unit dense networks with tanh activations.
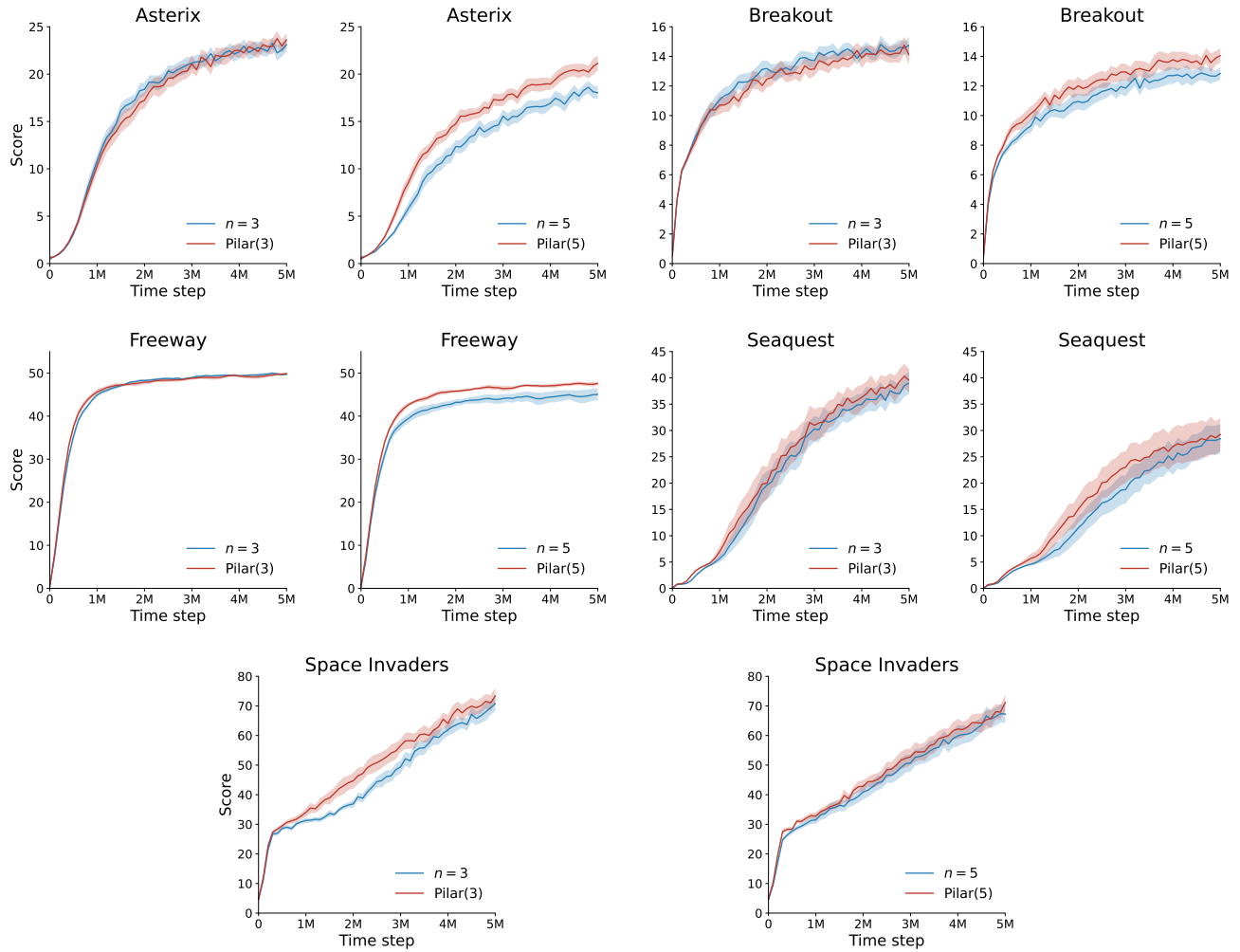
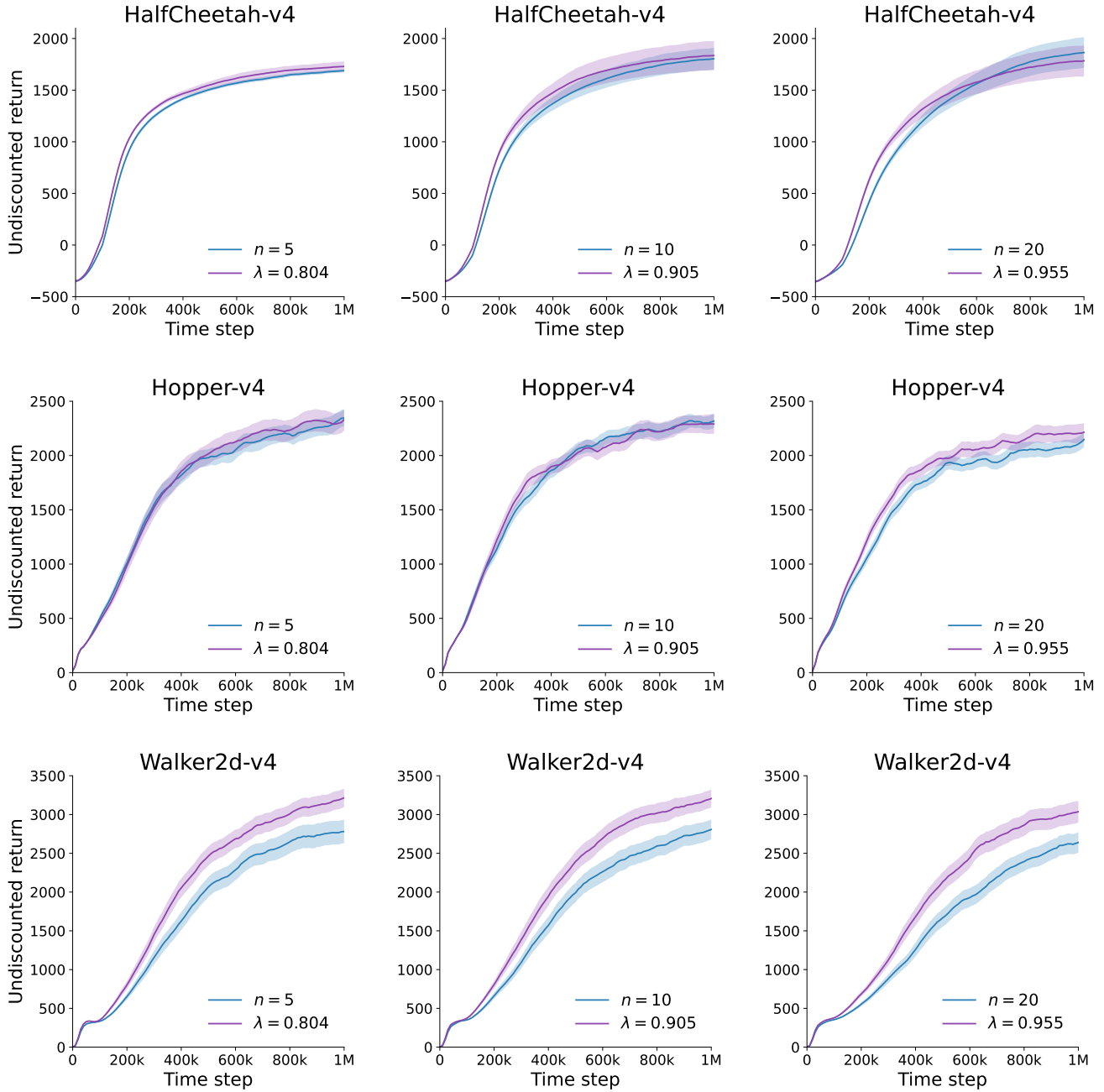*Figure 6.* Learning curves for DQN with n-step returns and Pilars in five MinAtar games.

Figure 7. Learning curves for PPO with n-step returns and λ-returns in three MuJoCo environments.