Fixed-Horizon Temporal Difference Methods for Stable Reinforcement Learning

Kristopher De Asis,[1] **Alan Chan,**[1,3] **Silviu Pitis,**[2] **Richard S. Sutton,**[1] **Daniel Graves**[3]
[1]University of Alberta, [2]University of Toronto, [3]Huawei Technologies Canada, Ltd.
{kldeasis, achan4, rsutton}@ualberta.ca, spitis@cs.toronto.edu, daniel.graves@huawei.com

Abstract

We explore fixed-horizon temporal difference (TD) methods, reinforcement learning algorithms for a new kind of value function that predicts the sum of rewards over a *fixed* number of future time steps. To learn the value function for horizon h, these algorithms bootstrap from the value function for horizon $h-1$, or some shorter horizon. Because no value function bootstraps from itself, fixed-horizon methods are immune to the stability problems that plague other off-policy TD methods using function approximation (also known as "the deadly triad"). Although fixed-horizon methods require the storage of additional value functions, this gives the agent additional predictive power, while the added complexity can be substantially reduced via parallel updates, shared weights, and n-step bootstrapping. We show how to use fixed-horizon value functions to solve reinforcement learning problems competitively with methods such as Q-learning that learn conventional value functions. We also prove convergence of fixed-horizon temporal difference methods with linear and general function approximation. Taken together, our results establish fixed-horizon TD methods as a viable new way of avoiding the stability problems of the deadly triad.

1 Temporal Difference Learning

Temporal difference (TD) methods (Sutton 1988) are an important approach to reinforcement learning (RL) that combine ideas from Monte Carlo estimation and dynamic programming. A key view of TD learning is that it incrementally learns testable, predictive knowledge of the environment (Sutton et al. 2011). The learned values represent answers to questions about how a signal will accumulate over time, conditioned on a way of behaving. In control tasks, this signal is the reward sequence, and the values represent an arbitrarily long sum of rewards an agent expects to receive when acting greedily with respect to its current predictions.

A TD learning agent's prediction horizon is specified through a *discount factor* (Sutton and Barto 2018). This parameter adjusts how quickly to exponentially decay the weight given to later outcomes in a sequence's sum, and allows computational complexity to be *independent of span* (van Hasselt and Sutton 2015). It's often set to a constant $\gamma \in [0, 1)$, but prior work generalizes the discount rate to be a transition-dependent *termination* function (White 2016). This allows for (variable) finite-length sums dependent on state transitions, like in episodic tasks.

In this paper, we explore a case of *time-dependent* discounting, where the sum considers a fixed number of future steps regardless of where the agent ends up. We derive and investigate properties of fixed-horizon TD algorithms, and identify benefits over infinite-horizon algorithms in both prediction and control. Specifically, by storing and updating a separate value function for each horizon, fixed-horizon methods avoid feedback loops when bootstrapping, so that learning is stable even in presence of function approximation. Fixed-horizon agents can approximate infinite-horizon returns arbitrarily well, expand their set of learned horizons freely (computation permitting), and combine forecasts from multiple horizons to make time-sensitive predictions about rewards. We emphasize our novel focus on predicting a fixed-horizon return from each state as a solution method, regardless of the problem setting. Our algorithms can be applied to both finite-horizon and infinite-horizon MDPs.

2 MDPs and One-step TD Methods

The RL problem is usually modeled as a Markov decision process (MDP), in which an agent interacts with an environment over a sequence of discrete time steps.

At each time step t, the agent receives information about the environment's current *state*, $S_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states in the MDP. The agent uses this state information to select an *action*, $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(s)$ is the set of possible actions in state s. Based on the current state and the selected action, the agent gets information about the environment's next state, $S_{t+1} \in \mathcal{S}$, and receives a *reward*, $R_{t+1} \in \mathbb{R}$, according to the *environment model*, $p(s', r|s, a) = P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$.

Actions are selected according to a *policy*, $\pi(a|s) = P(A_t = a|S_t = s)$, which gives the probability of taking action a given state s. An agent is interested in the return:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{1}$$

where $\gamma \in [0, 1]$ and T is the final time step in an episodic task, and $\gamma \in [0, 1)$ and $T = \infty$ for a continuing task.

Value-based methods approach the RL problem by computing *value functions*. In prediction, or *policy evaluation*, the goal is to accurately estimate a policy's expected return, and a *state-value function*, denoted $v_\pi(s)$, is estimated. In control, the goal is to learn a policy that maximizes the expected return, and an *action-value function*, denoted $q_\pi(s, a)$, is estimated. In each case, the value functions represent a policy's expected return from state s (and action a):

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \tag{2}$$

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{3}$$

TD methods learn to approximate value functions by expressing Equations 2 and 3 in terms of successor values (the *Bellman equations*). The Bellman equation for v_π is:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big(r + \gamma v_\pi(s')\Big) \tag{4}$$

Based on Equation 4, one-step TD prediction estimates the return by taking an action in the environment according to a policy, sampling the immediate reward, and bootstrapping off of the current estimated value of the next state for the remainder of the return. The difference between this *TD target* and the value of the previous state (the *TD error*) is then computed, and the previous state's value is updated by taking a step proportional to the TD error with $\alpha \in (0, 1]$:

$$\hat{G}_t = R_{t+1} + \gamma V(S_{t+1}) \tag{5}$$

$$V(S_t) \leftarrow V(S_t) + \alpha[\hat{G}_t - V(S_t)] \tag{6}$$

Q-learning (Watkins 1989) is arguably the most popular TD method for control. It has a similar update, but because policy improvement is involved, its target is a sample of the Bellman optimality equation for action-values:

$$\hat{G}_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') \tag{7}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[\hat{G}_t - Q(S_t, A_t)] \tag{8}$$

For small finite state spaces where the value function can be stored as a single parameter vector, also known as the *tabular* case, TD methods are known to converge under mild technical conditions (Sutton and Barto 2018). For large or uncountable state spaces, however, one must use function approximation to represent the value function, which does not have the same convergence guarantees.

Some early examples of divergence with function approximation were provided by Boyan and Moore (1995), who proposed the Grow-Support algorithm to combat divergence. Baird (1995) provided perhaps the most famous example of divergence (discussed below) and proposed residual algorithms. Gordon (1995) proved convergence for a specific class of function approximators known as *averagers*. Convergence of TD prediction using linear function approximation, first proved by Tsitsiklis and Van Roy (1997), requires the training distribution to be on-policy. This approach was later extended to Q-learning (Melo and Ribeiro 2007), but under relatively stringent conditions. In particular, Assumption (7) of Theorem 1 in Melo and Ribeiro (2007) amounts to a requirement that the behaviour policy is already somewhat close to the optimal policy.

The on-policy limitations of the latter two results reflect what has come to be known as the *deadly triad* (Sutton and Barto 2018): when using (1) TD methods with (2) function approximation, training on (3) an off-policy data distribution can result in instability and divergence. One response to this problem is to shift the optimization target, as done by Gradient TD (GTD) methods (Sutton et al. 2009; Bhatnagar et al. 2009); while provably convergent, GTD algorithms are empirically slow (Ghiassian et al. 2018; Sutton and Barto 2018). Another approach is to approximate Fitted Value Iteration (FVI) methods (Munos and Szepesvári 2008) using a *target network*, as proposed by Mnih et al. (2015). Though lacking convergence guarantees, this approach has been empirically successful. In the next section, we propose fixed-horizon TD methods, an alternative approach to ensuring stability in presence of the deadly triad.

3 Fixed-horizon TD Methods

A *fixed-horizon return* is a sum of rewards similar to Equation 1 that includes only a fixed number of future steps. For a fixed horizon h, the fixed-horizon return is defined to be:

$$G_t^h = \sum_{k=0}^{\min(h, T-t)-1} \gamma^k R_{t+k+1} \tag{9}$$

which is well-defined for any finite γ. This formulation allows the agent's horizon of interest and sense of urgency to be characterized more flexibly and admits the use of $\gamma = 1$ in the continuing setting. Fixed-horizon value functions are defined as expectations of the above sum when following policy π beginning with state s (and action a):

$$v_\pi^h(s) = \mathbb{E}_\pi[G_t^h | S_t = s] \tag{10}$$

$$q_\pi^h(s, a) = \mathbb{E}_\pi[G_t^h | S_t = s, A_t = a] \tag{11}$$

These fixed-horizon value functions can also be written in terms of successor values. Instead of bootstrapping off of the same value function, $v_\pi^h(s)$ bootstraps off of the successor state's value from an earlier horizon (Bertsekas 2012):

$$v_\pi^h(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a)\Big(r + \gamma v_\pi^{h-1}(s')\Big) \tag{12}$$

where $v_\pi^0(s) = 0$ for all $s \in \mathcal{S}$.

From the perspective of *generalized value functions* (GVFs) (Sutton et al. 2011), fixed-horizon value functions are compositional GVFs. If an agent knows about a signal up to a final horizon of H, it can specify a question under the same policy with a horizon of $H + 1$, and directly make use of existing values. As $H \to \infty$, the fixed-horizon return converges to the infinite-horizon return, so that a fixed-horizon agent with sufficiently large H could solve infinite-horizon tasks arbitrarily well. Indeed, for an infinite-horizon MDP with absolute rewards bounded by R_{max} and $\gamma < 1$, it is easy to see that $|v_\pi^h(s) - v_\pi(s)| \leq \gamma^h \frac{R_{max}}{1-\gamma}$.

For a final horizon $H \ll \infty$, there may be concerns about suboptimal control. We explore this empirically in Section 5. For now, we note that optimality is never guaranteed when values are approximated. This can result from function approximation, or even in tabular settings from the use of a constant step size. Further, recent work shows benefits in considering shorter horizons (van Seijen, Fatemi, and Tavakoli 2019) based on performance metric mismatches.

One-step Fixed-horizon TD

One-step fixed-horizon TD (FHTD) learns approximate values $V^h \approx v^h_\pi$ by computing, for each $h \in \{1, 2, 3, ..., H\}$:

$$\hat{G}^h_t = R_{t+1} + \gamma V^{h-1}(S_{t+1}) \tag{13}$$

$$V^h(S_t) \leftarrow V^h(S_t) + \alpha[\hat{G}^h_t - V^h(S_t)] \tag{14}$$

where $V^0(s) = 0$ for all $s \in \mathcal{S}$. The general procedure of one-step FHTD was previously described by Sutton (1988), but not tested due to the observation that one-step FHTD's computation and storage scale linearly with the final horizon H, as it performs H value updates per step. We argue that because these value updates can be parallelized, reasonably long horizons are feasible, and we'll later show that n-step FHTD can make even longer horizons practical. Forms of temporal abstraction (Sutton, Precup, and Singh 1999) can further substantially reduce the complexity.

A key property of FHTD is that bootstrapping is grounded by the 0th horizon, which is *exactly* known from the start. The 1st horizon's values estimate the expected immediate reward from each state, which is a stable target. The 2nd horizon bootstraps off of the 1st, which eventually becomes a stable target, and so forth. This argument intuitively applies even with general function approximation, assuming weights are not shared between horizons. To see the implications of this on TD's deadly triad, consider one-step FHTD with linear function approximation. For each horizon's weight vector, $\mathbf{w}^h$, the following update is performed:

$$\mathbf{w}^h_{t+1} \leftarrow \mathbf{w}^h_t + \alpha\big[R_{t+1} + \gamma \mathbf{w}^{h-1}_t \cdot \boldsymbol{\phi}_{t+1} - \mathbf{w}^h_t \cdot \boldsymbol{\phi}_t\big]\boldsymbol{\phi}_t$$

where $\boldsymbol{\phi}_t$ is the feature vector of the state at time t, $\boldsymbol{\phi}(S_t)$. The expected update can be written as:

$$\mathbf{w}^h_{t+1} \leftarrow \mathbf{w}^h_t + \alpha[\mathbf{b} - \mathbf{A}\mathbf{w}^h_t]$$

where $\mathbf{A} = \mathbb{E}[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^T]$ and $\mathbf{b} = \mathbb{E}[(R_{t+1} + \gamma \mathbf{w}^{h-1}_t \cdot \boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t]$, having expectations over the transition dynamics and the sampling distribution over states. Because the target uses a different set of weights, the $\mathbf{b}$ vector is non-stationary (but convergent), and the $\mathbf{A}$ matrix is an expectation over the *fixed* sampling distribution. Defining $\boldsymbol{\Phi}$ to be the matrix containing each state's feature vector along each row, and $\mathbf{D}$ to be the diagonal matrix containing the probability of sampling each state on its diagonal, we have that $\mathbf{A} = \boldsymbol{\Phi}^T \mathbf{D} \boldsymbol{\Phi}$. $\mathbf{A}$ is always positive definite and so the updates are guaranteed to be stable relative to the current $\mathbf{b}$ vector. In contrast, TD's update gives $\mathbf{A} = \boldsymbol{\Phi}^T \mathbf{D}(I - \gamma \mathbf{P})\boldsymbol{\Phi}$ where $\mathbf{P}$ contains the state-transition probabilities under the target policy. TD's $\mathbf{A}$ matrix can be negative definite if $\mathbf{D}$ and $\mathbf{P}$ do not match (i.e., off-policy updates), and the weights may diverge. We explore the convergence of FHTD formally in Section 4.

A horizon's reliance on earlier horizons' estimates being accurate may be expected to have worse sample complexity. If each horizon's parameters are identically initialized, distant horizons will match infinite-horizon TD's updates for the first $H - 1$ steps. If $H \to \infty$, this suggests that FHTD's sample complexity might be upper bounded by that of TD. See Appendix D for a preliminary result in this vein.

An FHTD agent has strictly more predictive capabilities than a standard TD agent. FHTD can be viewed as computing an inner product between the rewards and a step function. This gives an agent an exact notion of *when* rewards occur, as one can subtract the fixed-horizon returns of subsequent horizons to get an individual expected reward at a specific future time step.

Multi-step Fixed-horizon TD Prediction

Another way to estimate fixed-horizon values is through *Monte Carlo* (MC) methods. From a state, one can generate reward sequences of fixed lengths, and average their sums into the state's expected fixed-horizon return. Fixed-horizon MC is an opposite extreme from one-step FHTD in terms of a bias-variance trade-off, similar to the infinite-horizon setting (Jaakkola, Jordan, and Singh 1994). This trade-off motivates considering multi-step FHTD methods.

Fixed-horizon MC is appealing when one only needs the expected return for the final horizon H, and has no explicit use for the horizons leading up to H. This is because it can learn the final horizon directly by storing and summing the last H rewards, and performing one value-function update for each visited state. If we use a fixed-horizon analogue to n-step TD methods (Sutton and Barto 2018), denoted n-step FHTD, the algorithm stores and sums the last n rewards, and only has to learn $\lceil \frac{H}{n} \rceil$ value functions. For each $h \in \{n, 2n, 3n, ..., H\}$, n-step FHTD computes:

$$\hat{G}^h_{t:t+n} = \gamma^n V^{h-n}(S_{t+n}) + \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} \tag{15}$$

$$V^h(S_t) \leftarrow V^h(S_t) + \alpha[\hat{G}^h_{t:t+n} - V^h(S_t)] \tag{16}$$

assuming that H is divisible by n. If H is not divisible by n, the final horizon's update will only sum the first $H \pmod n$ of the last n rewards. Counting the number of rewards to sum, and the number of value function updates, n-step FHTD performs $n - 1 + H/n$ operations per time step. This has a worst case of H operations at $n = 1$ and $n = H$, and a best case of $2\sqrt{H} - 1$ operations at $n = \sqrt{H}$. This suggests that in addition to trading off reliance on sampled information versus current estimates, n-step FHTD's computation can scale sub-linearly in H.

Fixed-horizon TD Control

The above FHTD methods for state-values can be trivially extended to learn action-values under fixed policies, but it is less trivial with policy improvement involved.

If we consider the best an agent can do from a state in terms of the next H steps, it consists of an immediate reward, and the best it can do in the next $H - 1$ steps from the next state. That is, each horizon has a separate target policy

that's greedy with respect to its own horizon. Because the greedy action may differ between one horizon and another, FHTD control is inherently off-policy.

Q-learning provides a natural way to handle this off-policyness, where the TD target bootstraps off of an estimate under a greedy policy. Based on this, fixed-horizon Q-learning (FHQ-learning) performs the following updates:

$$\hat{G}_t^h = R_{t+1} + \gamma \max_{a'} Q^{h-1}(S_{t+1}, a') \tag{17}$$

$$Q^h(S_t, A_t) \leftarrow Q^h(S_t, A_t) + \alpha[\hat{G}_t^h - Q^h(S_t, A_t)] \tag{18}$$

A saddening observation from FHTD control being inherently off-policy is that the computational savings of n-step FHTD methods may not be possible without approximations. With policy improvement, an agent needs to know the current greedy action for each horizon. This information isn't available if n-step FHTD methods avoid learning intermediate horizons. On the other hand, a benefit of each horizon having its own greedy policy is that the optimal policy for a horizon is unaffected by the policy improvement steps of later horizons. As a compositional GVF, the unidirectional decoupling of greedy policies suggests that in addition to a newly specified final horizon leveraging previously learned values for prediction, it can leverage *previously learned policies* for control.

4 Convergence of Fixed-horizon TD

This section sets forth our convergence results, first for linear function approximation (which includes the tabular case), and second for general function approximation.

Linear Function Approximation

For linear function approximation, we prove the convergence of FHQ-Learning under some additional assumptions, outlined in detail in Appendix A. Analogous proofs may be obtained easily for policy evaluation. We provide a sketch of the proof below, and generally follow the outline of Melo and Ribeiro (2007). Full proofs are in the Appendix.

We denote $\phi(s, a) \in \mathbb{R}^d$ for the feature vector corresponding to the state s and action a. We will sometimes write ϕ_s for $\phi(s, a)$. Assume furthermore that the features are linearly independent and bounded.

We assume that we are learning H horizons, and approximate the fixed-horizon h-th action-value function linearly.

$$Q^h_{\mathbf{w}}(s, a) := \mathbf{w}^{h,:}\phi(s, a),$$

where $\mathbf{w} \in \mathbb{R}^{H \times d}$. Colon indices denote array slicing, with the same conventions as in NumPy. For convenience, we also define $\mathbf{w}^{0,:} := 0$.

We define the feature corresponding to the max action of a given horizon:

$$\phi^*_{\mathbf{w}}(s, h) := \phi(s, \arg\max_a Q^h_{\mathbf{w}}(s, a)).$$

The update at time $t + 1$ for each horizon h is given by

$$\mathbf{w}_{t+1}^{h+1,:} := \mathbf{w}_t^{h+1,:} + \alpha_t(r(s, a, s') + \gamma \mathbf{w}_t^{h,:}\phi^*_{\mathbf{w}_t}(s', h) - \mathbf{w}_t^{h+1,:}\phi_s)\phi_s^T.$$

where the step-sizes α_t are assumed to satisfy:

$$\sum_t \alpha_t = \infty \quad \text{and} \quad \sum_t \alpha_t^2 < \infty.$$

Proposition 1. *For $h = 1, ..., H$, the following ODE system has an equilibrium.*

$$\dot{\mathbf{w}}^{h+1,:} = \mathbb{E}\left[(r(s, a, s') + \gamma \mathbf{w}^{h,:}\phi^*_{\mathbf{w}}(s', h) - \mathbf{w}^{h+1,:}\phi_s)\phi_s^T\right] \tag{19}$$

Denote one such equilibrium by $\mathbf{w}_e$, and define $\tilde{\mathbf{w}} := \mathbf{w} - \mathbf{w}_e$. If, furthermore, we have that for $h = 1, ..., H$,

$$\gamma^2 \mathbb{E}[(\mathbf{w}^{h,:}\phi^*_{\mathbf{w}}(s, h) - \mathbf{w}_e^{h,:}\phi^*_{\mathbf{w}_e}(s, h))^2] < \mathbb{E}\left[(\mathbf{w}^{h+1,:}\phi_s - \mathbf{w}_e^{h+1,:}\phi_s)^2\right] \tag{20}$$

then $\mathbf{w}_e$ is a globally asymptotically stable equilibrium of Equation (19)*.*

Proof. See Appendix A. The main idea is to explicitly construct an equilibrium of Equation (19) and to use a Lyapunov function along with Equation (20) to show global asymptotic stability. □

Equation (20) means that the h-th fixed-horizon action-value function must be closer, when taking respective max actions, to its equilibrium point than the $(h + 1)$-th fixed-horizon action-value function is to its equilibrium, where distance is measured in terms of squared error of the action-values. Intuitively, the functions for the previous horizons must have converged somewhat for the next horizon to converge, formalizing the cascading effect described earlier. This assumption is reasonable given that value functions for smaller horizons have a bootstrap target that is closer to a Monte Carlo target than the value functions for larger horizons. As a result, eq. (20) is somewhat less stringent than the corresponding assumption (7) in Theorem 1 of Melo and Ribeiro (2007), which requires the behaviour policy already be somewhat close to optimal.

Theorem 1. *Viewing the right-hand side of the ODE system eq.* (19) *as a single function $g(\mathbf{w})$, assume that g is locally Lipschitz in $\mathbf{w}$. Assuming also Equation* (20)*, a fixed behaviour policy π, and the assumptions in the Appendix, the iterates of FHQ-learning converge with probability 1 to the equilibrium of the ODE system eq.* (19)*.*

Proof. See Appendix A. The main idea is to use Theorem 17 of Benveniste, Métivier, and Priouret (1990) and Proposition 1. □

A limitation of Theorem 1 is the assumption of a fixed behaviour policy. It is possible to make a similar claim for a changing policy, assuming that it satisfies a form of Lipschitz continuity with respect to the parameters. See Melo and Ribeiro (2007) for discussion on this point.

General Function Approximation

We now address the case where Q^h is represented by a general function approximator (e.g., a neural network). As before, the analysis extends easily to prediction (V^h). General non-linear function approximators have non-convex loss surfaces and may have many saddle points and local minima. As a result, typical convergence results (e.g., for gradient descent) are not useful without some additional assumption about the approximation error (cf., the *inherent Bellman error* in the analysis of Fitted Value Iteration (Munos and Szepesvári 2008)). We therefore state our result for general function approximators in terms of δ-*strongness* for $\delta > 0$:

Definition 1. *A function approximator, consisting of function class $\mathcal{H}$ and iterative learning algorithm $\mathcal{A}$, is **δ-strong** with respect to target function class $\mathcal{G}$ and loss function $J : \mathcal{H} \times \mathcal{G} \to \mathbb{R}^+$ if, for all target functions $g \in \mathcal{G}$, the learning algorithm $\mathcal{A}$ is guaranteed to produce (within a finite t number of steps) an $h_t \in \mathcal{H}$ such that $J(h_t, g) \leq \delta$.*

We consider learning algorithms $\mathcal{A}$ that converge once some minimum progress c can no longer be made:

Assumption 1. *There exists stopping constant c such that algorithm $\mathcal{A}$ is considered "converged" with respect to target function g if less than c progress would be made by an additional step; i.e., if $J(h_t, g) - J(h_{t+1}, g) < c$.*

Note that δ-strongness may depend on stopping constant c: a larger c naturally corresponds to earlier stopping and looser δ. Note also that, so long as the distance between the function classes $\mathcal{H}$ and $\mathcal{G}$ is upper bounded, say by $\boldsymbol{d}$, any convergent $\mathcal{A}$ is "$\boldsymbol{d}$-strong". Thus, a δ-strongness result is only meaningful to the extent that δ is sufficiently small.

We consider functions $Q_t^h = Q(\mathbf{w}_t^h)$ parameterized by $\mathbf{w}_t^h$. Letting T denote the Bellman operator $[TQ](s,a) = \mathbb{E}_{s',r\sim p(\cdot|s,a)}[r + \gamma \max_{a'} Q(s',a')]$, we assume:

Assumption 2. *The target function $TQ(\mathbf{w})$ is Lipschitz continuous in the parameters: there exists constant L such that $\|TQ(\mathbf{w}_1) - TQ(\mathbf{w}_2)\|_{\mathcal{F}} \leq L \|\mathbf{w}_1 - \mathbf{w}_2\|$ for all $\mathbf{w}_1, \mathbf{w}_2$, where $\|\cdot\|_{\mathcal{F}}$ is a norm on value function space $\mathcal{F}$ (typically a weighted L_2 norm, weighted by the data distribution), which we take to be a Banach space containing both $\mathcal{H}$ and $\mathcal{G}$.*

It follows that if $(\mathbf{w}_t^{h-1}) \to \mathbf{w}_*^{h-1}$, the sequence of target functions TQ_t^{h-1} converges to $TQ_*^{h-1} = TQ(\mathbf{w}_*^{h-1})$ in $\mathcal{F}$ under norm $\|\cdot\|$. We can therefore define the "true" loss:

$$J^*(\mathbf{w}_t^h) = \|TQ_*^{h-1} - Q_t^h\| \tag{21}$$

where we drop the square from the usual mean square Bellman error (Sutton and Barto 2018) for ease of exposition (the analysis is unaffected after an appropriate adjustment). Since we cannot access J^*, we optimize the surrogate loss:

$$J(\mathbf{w}_t^{h-1}, \mathbf{w}_t^h) = \|TQ_t^{h-1} - Q_t^h\|. \tag{22}$$

Lemma 1. *If $\|TQ_t^{h-1} - TQ_*^{h-1}\| < \epsilon$, and learning has not yet converged with respect to the surrogate loss J, then $J^*(\mathbf{w}_t^h) - J^*(\mathbf{w}_{t+1}^h) > c - 2\epsilon$.*

Proof. Intuitively, enough progress toward a similar enough surrogate loss guarantees progress toward the true loss. Applying the triangle inequality (twice) gives:

$$\begin{aligned}
&J^*(\mathbf{w}_t^h) - J^*(\mathbf{w}_{t+1}^h) = \|TQ_*^{h-1} - Q_t^h\| - \|TQ_*^{h-1} - Q_{t+1}^h\| \\
&\quad = \|TQ_*^{h-1} - Q_t^h\| \\
&\qquad + (\|TQ_t^{h-1} - TQ_*^{h-1}\| - \|TQ_t^{h-1} - TQ_*^{h-1}\|) \\
&\qquad - \|TQ_*^{h-1} + (TQ_t^{h-1} - TQ_t^{h-1}) - Q_{t+1}^h\| \\
&\quad \geq (\|TQ_t^{h-1} - Q_t^h\| - \|TQ_t^{h-1} - TQ_*^{h-1}\|) \\
&\qquad - (\|TQ_t^{h-1} - Q_{t+1}^h\| + \|TQ_*^{h-1} - TQ_t^{h-1}\|) \\
&\quad = (\|TQ_t^{h-1} - Q_t^h\| - \|TQ_t^{h-1} - Q_{t+1}^h\| \\
&\qquad - 2\|TQ_*^{h-1} - TQ_t^{h-1}\| > c - 2\epsilon,
\end{aligned}$$

where the final inequality uses $\|TQ_t^{h-1} - Q_t^h\| - \|TQ_t^{h-1} - Q_{t+1}^h\| \geq c$ from Assumption 1. □

It follows from Lemma 1 that when ϵ is small enough—$\epsilon < \frac{c}{2} - k$ for some constant k—either the true loss J^* falls by at least k, or learning has converged with respect to the current target TQ_t^{h-1}. Since J^* is non-negative (so cannot go to $-\infty$), it follows that the *loss* converges to a δ-strong solution: $J^*(\mathbf{w}_t^h) \to d$ with $d \leq \delta$. Since there are only a finite number of k-sized steps between the current loss at time t and 0 (i.e., only a finite number of opportunities for the learning algorithm to have "not converged" with respect to the surrogate J), the parameters $\mathbf{w}_t^h$ must also converge.

Since $Q_t^0 = 0$ is stationary, it follows by induction that:

Theorem 2. *Under Assumptions 1 and 2, each horizon of FHQ-learning converges to a δ-strong solution when using a δ-strong function approximator.*

In contrast to Theorem 1, which applies quite generally to linear function approximators, δ-strongness and Assumption 1 limit the applicability of Theorem 2 in two important ways.

First, since gradient-based learning may stall in a bad saddle point or local minimum, neural networks are not, in general, δ-strong for small δ. Nevertheless, repeat empirical experience shows that neural networks consistently find good solutions (Zhang et al. 2016), and a growing number of theoretical results suggest that almost all local optima are "good" (Pascanu et al. 2014; Choromanska et al. 2015; Pennington and Bahri 2017). For this reason, we argue that δ-strongness is reasonable, at least approximately.

Second, Assumption 1 is critical to the result: only if progress is "large enough" relative to the error in the surrogate target is learning guaranteed to make progress on J^*. Without a lower bound on progress—e.g., if the progress at each step is allowed to be less than 2ϵ regardless of ϵ—training might accumulate an error on the order of ϵ at every step. In pathological cases, the sum of such errors may diverge even if $\epsilon \to 0$. As stated, Assumption 1 does not reflect common practice: rather than progress being measured at every step, it is typically measured over several, say k, steps. This is because training targets are noisy estimates of the expected Bellman operator and several steps are needed to accurately assess progress. Our analysis can be adapted to this more practical scenario by making use of a target network (Mnih et al. 2015) to freeze the targets for k steps at a time. Then, considering each k step window as a single step in the above discussion, Assumption 1 is fair. This said, intuition suggests that pathological divergence when Assumption 1 is not satisfied is rare, and our experiments with Deep FHTD Control show that training can be stable even with shared weights and no target networks.

5 Empirical Evaluation

This section outlines several hypotheses concerning fixed-horizon TD methods, experiments aimed at testing them, and the results from each experiment. Pseudo-code, diagrams, more experimental details, and additional experiments can be found in the supplementary material.

Stability in Baird's Counterexample

We hypothesize that FHTD methods provide a stable way of bootstrapping, such that divergence will not occur under off-policy updating with function approximation. To test this, we used Baird's counterexample (Baird 1995), a 7-state MDP where every state has two actions. One action results in a uniform random transition to one of the first 6 states, and the other action results in a deterministic transition to the 7th state. Rewards are always zero, and each state has a specific feature vector for use with linear function approximation. It was presented with a discount rate of $\gamma = 0.99$, and a target policy which always chooses to go to the 7th state.

In our experiment, we used one-step FHTD with importance sampling corrections (Rubinstein 1981) to predict up to a horizon of $H = \frac{1}{1-\gamma} = 100$. Each horizon's weights were initialized to be $\mathbf{w}^h = [1, 1, 1, 1, 1, 1, 10, 1]^T$, based on Sutton and Barto (2018), and we used a step size of $\alpha = \frac{0.2}{|\mathcal{S}|}$. We performed 1000 independent runs of 10,000 steps, and the results can be found in Figure 1.

We see that one-step FHTD eventually and consistently converges. The initial apparent instability is due to each horizon being initialized to the same weight vector, making early updates resemble the infinite-horizon setting where weight updates bootstrap off of the same weight vector. The results emphasize what TD would have done, and how FHTD can recover from it. Of note, the final weights do give optimal state-values of 0 for each state. In results not shown, FHTD still converges, sometimes quicker, when each horizon's weights are initialized randomly (and not identically).

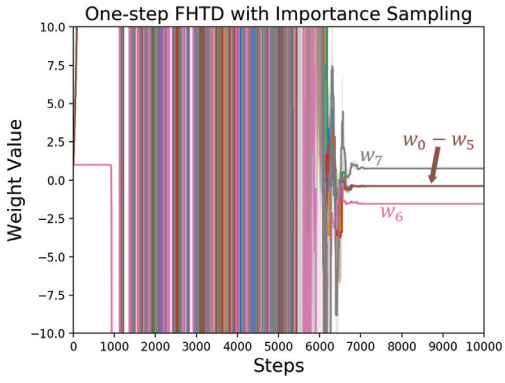

Figure 1: Weight trajectories of one-step FHTD's 100th horizon value function on Baird's counterexample, plotted after each time step. Shaded regions represent one standard error.

Tabular FHTD Control

In this section, we evaluate one-step FHQ-learning in a control problem. We hypothesize that when transitions are highly stochastic, predicting too far into the future results in unnecessarily large variance. Using fixed step sizes, we expect this to be an issue even in tabular settings. Both truncating the horizon and constant-valued discounting can address the variability of long term information, so we compare undiscounted FHQ-learning to discounted Q-learning.

We designed the *slippery maze* environment, a maze-like grid world with 4-directional movement. The agent starts in the center, and hitting walls keep the agent in place. The "slipperiness" involves a 75% chance that the agent's action is overridden by a random action. A reward of -1 is given at each step. The optimal deterministic path is 14 steps, but due to stochasticity, an optimal policy averages 61.77 steps.

Each agent behaved ϵ-greedily with $\epsilon = 0.1$. We swept linearly spaced step-sizes, final horizons $H \in \{8, 16, 32, 48\}$ for FHQ-learning, and discount rates $\gamma \in \{0.875, 0.938, 0.969, 0.979\}$ for Q-learning. The discount rates were selected such that if $1 - \gamma$ represented a per-step termination probability of a stochastic process, the expected number of steps before termination matches the tested values of H. We performed 100 independent runs, and Figure 2 shows the mean episode length over 100 episodes.

For FHQ-learning, it can be seen that if the final horizon is unreasonably short ($H = 8$), the agent performs poorly. However, $H = 16$ does considerably better than if it were to predict further into the future. With Q-learning, each discount rate performed relatively similar to one another, despite discount rates chosen to have expected sequence lengths comparable to the fixed horizons. This may be because they still include a portion of the highly variable information about further steps. For both algorithms, a shorter horizon was preferable over the full episodic return.

Deep FHTD Control

We further expect FHTD methods to perform well in control with non-linear function approximation. FHTD's derivation assumes weights are separated by horizon. To see the effect of horizons sharing weights, we treated each horizon's values as linear neural network outputs over shared hidden layers. Use of this architecture along with parallelization emphasizes that the increased computation can be minimal. Due to bootstrapped targets being decoupled by horizon, we also expect deep FHTD methods to not need target networks.

In OpenAI Gym's *LunarLander-v2* environment (Brockman et al. 2016), we compared Deep FHQ-learning (DFHQ) with a final horizon $H = 64$ and DQN (Mnih et al. 2015). We restricted the neural network to have two hidden layers, and swept over hidden layer widths for each algorithm. We used $\gamma \in \{0.99, 1.0\}$, and behaviour was ϵ-greedy with ϵ annealing linearly from 1.0 to 0.1 over 50,000 frames. RMSprop (Tieleman and Hinton 2012) was used on sampled mini-batches from an experience replay buffer (Mnih et al. 2015), and ignoring that the target depends on the weights,

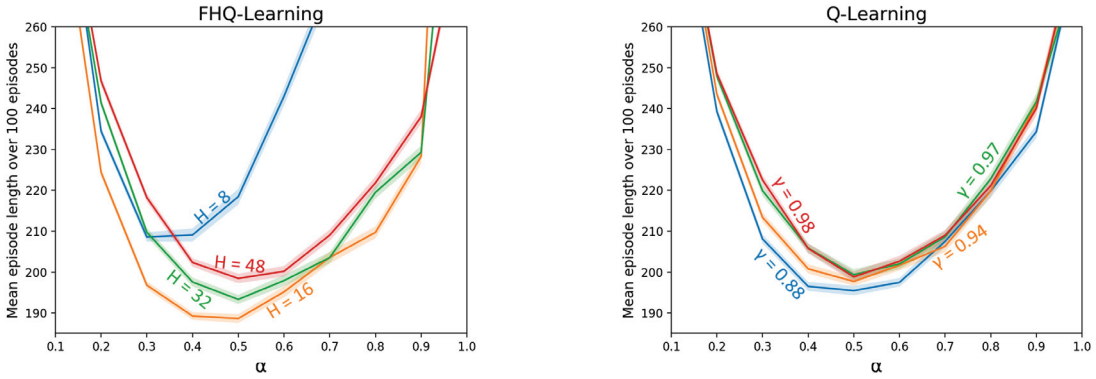

Figure 2: Mean episode lengths over 100 episodes of FHQ-learning and Q-learning with various step-sizes and horizons of interest. Results are averaged over 100 runs, and shaded regions represent one standard error.

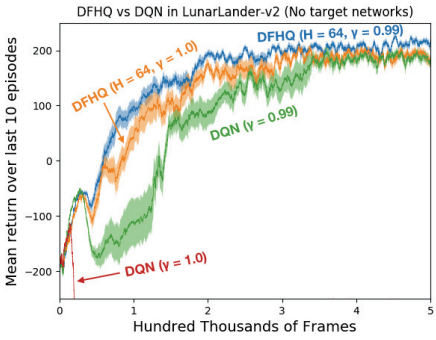

Figure 3: Mean return over last 10 episodes at each frame of DFHQ and DQN, without target networks, averaged over 30 runs. Shaded regions represent one standard error.

DFHQ minimized the mean-squared-error across horizons:

$$\hat{G}_t^h = R_{t+1} + \gamma \max_{a'} Q^{h-1}(S_{t+1}, a'; \mathbf{w})$$

$$J(\mathbf{w}) = \frac{1}{H} \sum_{h=1}^{H} \left(\hat{G}_t^h - Q^h(S_t, A_t; \mathbf{w}) \right)^2 \qquad (23)$$

We performed 30 independent runs of 500,000 frames each (approximately 1000 episodes for each run). Figure 3 shows for each frame, the mean return over the last 10 episodes of each algorithm's best parameters (among those tested) in terms of area under the curve. Note that the results show DFHQ and DQN *without* target networks. From additional experiments, we found that target networks slowed DFHQ's learning more than it could help over a run's duration. They marginally improve DQN's performance, but the area under the curve remained well below that of DFHQ.

It can be seen that DFHQ had considerably lower variance, and relatively steady improvement. Further, DFHQ was significantly less sensitive to γ, as DQN with $\gamma = 1$ immediately diverged. From the remainder of our sweep, DFHQ appeared relatively insensitive to large hidden layer widths beyond the setting shown, whereas DQN's performance considerably dropped if the width further increased.

DFHQ's good performance may be attributed to the representation learning benefit of predicting many outputs (Jaderberg et al. 2016; Fedus et al. 2019); in contrast with auxiliary tasks, however, these outputs are *necessary tasks* for predicting the final horizon. An early assessment of the learned values can be found in Appendix D.

6 Discussion and future work

In this work, we investigated using *fixed-horizon* returns in place of the conventional infinite-horizon return. We derived FHTD methods and compared them to their infinite-horizon counterparts in terms of prediction capability, complexity, and performance. We argued that FHTD agents are stable under function approximation and have additional predictive power. We showed that the added complexity can be substantially reduced via parallel updates, shared weights, and n-step bootstrapping. Theoretically, we proved convergence of FHTD methods with linear and general function approximation. Empirically, we showed that off-policy linear FHTD converges on a classic counterexample for off-policy linear TD. Further, in a tabular control problem, we showed that greedifying with respect to estimates of a short, fixed horizon could outperform doing so with respect to longer horizons. Lastly, we demonstrated that FHTD methods can scale well to and perform competitively on a deep reinforcement learning control problem.

There are many avenues for future work. Given that using shorter horizons may be preferable (Figure 2), it would be interesting if optimal weightings of horizons could be learned, rather than relying on the furthest horizon to act. Developing ways to handle the off-policyness of n-step FHTD control (See Appendix D), incorporating temporal abstraction, and experiments in more complex environments would improve our understanding of the scalability of our methods to extremely long horizon tasks. Finally, the applications to complex and hyperbolic discounting (Appendix B), exploring the benefits of iteratively deepening the final horizon, and the use of fixed-horizon critics in actor-critic methods might be promising.

Acknowledgments The authors thank the Reinforcement Learning and Artificial Intelligence research group, Amii, and the Vector Institute for providing the environment to nurture and support this research. We gratefully acknowledge funding from Alberta Innovates – Technology Futures, Google Deepmind, and from the Natural Sciences and Engineering Research Council of Canada.

References

Baird, L. 1995. Residual algorithms: Reinforcement learning with function approximation. In Prieditis, A., and Russell, S., eds., *Machine Learning Proceedings 1995*. Morgan Kaufmann. 30 – 37.

Benveniste, A.; Métivier, M.; and Priouret, P. 1990. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag.

Bertsekas, D. 2012. *Dynamic Programming & Optimal Control, Vol II: Approximate Dynamic Programming*. Athena Scientific, 4 edition.

Bhatnagar, S.; Precup, D.; Silver, D.; Sutton, R. S.; Maei, H. R.; and Szepesvári, C. 2009. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, 1204–1212.

Boyan, J. A., and Moore, A. W. 1995. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*, 369–376.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI gym. *CoRR* abs/1606.01540.

Choromanska, A.; Henaff, M.; Mathieu, M.; Arous, G. B.; and LeCun, Y. 2015. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, 192–204.

Fedus, W.; Gelada, C.; Bengio, Y.; Bellemare, M. G.; and Larochelle, H. 2019. Hyperbolic discounting and learning over multiple horizons.

Ghiassian, S.; Patterson, A.; White, M.; Sutton, R. S.; and White, A. 2018. Online off-policy prediction. *arXiv preprint arXiv:1811.02597*.

Gordon, G. J. 1995. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*. Elsevier. 261–268.

Jaakkola, T. S.; Jordan, M. I.; and Singh, S. P. 1994. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation* 6(6):1185–1201.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *CoRR* abs/1611.05397.

Melo, F. S., and Ribeiro, M. I. 2007. Q-learning with linear function approximation. In *International Conference on Computational Learning Theory*, 308–322. Springer.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Munos, R., and Szepesvári, C. 2008. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research* 9(May):815–857.

Pascanu, R.; Dauphin, Y. N.; Ganguli, S.; and Bengio, Y. 2014. On the saddle point problem for non-convex optimization. *arXiv preprint arXiv:1405.4604*.

Pennington, J., and Bahri, Y. 2017. Geometry of neural network loss surfaces via random matrix theory. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2798–2806. JMLR. org.

Rubinstein, R. Y. 1981. *Simulation and the Monte Carlo Method*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition.

Sutton, R. S.; Maei, H. R.; Precup, D.; Bhatnagar, S.; Silver, D.; Szepesvári, C.; and Wiewiora, E. 2009. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 993–1000. ACM.

Sutton, R. S.; Modayil, J.; Delp, M.; Degris, T.; Pilarski, P. M.; White, A.; and Precup, D. 2011. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS*, 761–768. IFAAMAS.

Sutton, R. S.; Precup, D.; and Singh, S. P. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181–211.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3(1):9–44.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

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

van Hasselt, H., and Sutton, R. S. 2015. Learning to predict independent of span. *CoRR* abs/1508.04582.

van Seijen, H.; Fatemi, H.; and Tavakoli, A. 2019. Using a logarithmic mapping to enable lower discount factors in reinforcement learning.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, King's College, Cambridge, UK.

White, M. 2016. Unifying task specification in reinforcement learning. *CoRR* abs/1609.01995.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.