# A Generalized Bootstrap Target for Value-Learning, Efficiently Combining Value and Feature Predictions

**Anthony GX-Chen** [1,2], **Veronica Chelu** [1,2], **Blake A. Richards** [1,2,3], **Joelle Pineau** [1,2,3,4]
[1] Mila - Quebec AI Institute
[2] McGill University
[3]Canada AI Chair program, CIFAR, Learning in Machines & Brains
[4]Meta AI Research
anthony.gx.chen@gmail.com

## Abstract

Estimating value functions is a core component of reinforcement learning algorithms. Temporal difference (TD) learning algorithms use *bootstrapping*, i.e. they update the value function toward a learning target using value estimates at subsequent time-steps. Alternatively, the value function can be updated toward a learning target constructed by separately predicting successor features (SF)—a policy-dependent *model*—and linearly combining them with instantaneous rewards. We focus on bootstrapping targets used when estimating value functions, and propose a new backup target, the *$\eta$-return mixture*, which implicitly *combines* value-predictive knowledge (used by TD methods) with (successor) feature-predictive knowledge—with a parameter $\eta$ capturing how much to rely on each. We illustrate that incorporating predictive knowledge through an $\eta\gamma$-discounted SF model makes more efficient use of sampled experience, compared to either extreme, i.e. bootstrapping entirely on the value function estimate, or bootstrapping on the product of separately estimated successor features and instantaneous reward models. We empirically show this approach leads to faster policy evaluation and better control performance, for tabular and nonlinear function approximations, indicating scalability and generality.

## 1 Introduction

The fundamental goal of reinforcement learning (RL) is to maximize return, i.e. (temporally discounted) cumulative reward. Value functions provide an estimate of the expected return from a specific state (and action), and as such, they are a fundamental component of RL algorithms. Modern deep RL methods require numerous environment interactions to solve complex tasks, which can be expensive or impossible to obtain, particularly for tasks resembling the real-world. This makes it essential to develop data-efficient methods for learning accurate value functions.

The problem we address in this work is that of credit assignment, namely how to associate (distant) rewards to the states and actions that caused them. Value-based RL methods tackle this problem through *temporal difference* (TD) learning algorithms (Sutton 1988). TD algorithms rely on *bootstrapping*: using the *value estimate* at a subsequent timestep, together with the observed data (e.g. rewards), to construct the learning *target*—the *return*—for the current timestep. However, the value estimate in the backup target does not need to come from the current value function being learned. For instance, value can be estimated using *successor features*—the (discounted) cumulative features—linearly combined with an estimate of instantaneous rewards (Barreto et al. 2017). This approach can make use of the same TD methods (Sutton 1988) to estimate the successor features as the former does when learning the value function, requiring similar amounts of sampled experience. Moreover, the *backup target* and the value function can be completely distinct (e.g. if the successor features and learned value function are dis-jointly parameterized); they can share feature representations (e.g. when the value function and the successor features are both linear functions of the features); or partially share representations (e.g. through Polyak averaging). Since the value function is regressed toward the target, the method of computing the target influences the quality of the value function.

In this paper, we aim to improve credit assignment and data efficiency for value-based methods, by proposing a new method of constructing a learning target, which borrows properties from all aforementioned approaches of target construction. This **$\eta$-return mixture** uses a parameter $\eta$ to combine an $\eta\gamma$-discounted successor features model ($\eta\gamma$-SF) *with* the current value function estimate to parameterize the learning target used during bootstrapping—with the $\eta$ parameter controlling the combination of value-predictive and feature-predictive knowledge. We observe an intermediate value of $\eta$ incorporates the benefits of both approaches in a complementary way, using sampled experience more efficiently.

**Contributions** In this paper we make three contributions: (i) We introduce the *$\eta$-return mixture*, a simple yet novel way of constructing a backup target for value learning, using an $\eta\gamma$-discounted SF model to interpolate between a direct value estimate and the fully factorized estimate relying on SF and instantaneous rewards. (ii) We describe a new learning algorithm using the *$\eta$-return mixture* as the bootstrap target for value estimation. (iii) We provide empirical results showing more efficient use of experience with the *$\eta$-return mixture* as the backup target, in both prediction and control, for tabular and nonlinear approximation, when compared to baselines.

## 2 Preliminaries

We denote random variables with uppercase (e.g., $S$) and the obtained values with lowercase letters (e.g., $S = s$). Multi-dimensional functions or vectors are bolded (e.g., $\mathbf{w}$), as are matrices (e.g. $\mathbf{\Phi}$). For all state-dependent functions, we also allow time-dependent shorthands (e.g., $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$).

### 2.1 Reinforcement Learning Problem Setup

A discounted Markov Decision Process (MDP) (Puterman 1994) is defined as the tuple $(\mathcal{S}, \mathcal{A}, P, r)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and transition probability function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{P}(\mathcal{S})$ (with $\mathcal{P}(\mathcal{S})$ the set of probability distributions on $\mathcal{S}$, and $P(s'|s, a)$ the probability of transitioning to state $s'$ by choosing action $a$ at state $s$). A policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ maps states to distributions over actions; $\pi(a|s)$ denotes the probability of choosing action $a$ in state $s$. Let $S_t, A_t, R_t$ denote the random variables of state, action and reward at time $t$, respectively.

Policy evaluation implies estimating the value function $v_\pi$, defined as the expected discounted return:

$$G_t \equiv R_{t+1} + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1} \,, \quad (1)$$

$$v_\pi(s) \equiv \mathbb{E}\left[G_t \mid S_t = s, A_k \sim \pi(S_k), k \geq t\right] \,, \quad (2)$$

where $\gamma \in [0, 1)$ is the discount factor. The learner's goal is to find a policy, $\pi$ which maximizes the *value* $v_\pi$. When the Markov chain induced by $\pi$ is ergodic, we denote with $d_\pi$ the stationary distribution induced by policy $\pi$. We henceforth shorthand the expectation over the environment dynamics and the policy $\pi$ with $\mathbb{E}_\pi[\cdot]$.

### 2.2 Value Learning

Typically, $v_\pi$ is represented directly, using a linear parametrization over some state features $\boldsymbol{\phi}(s) \in \mathbb{R}^d$, where $d$ is the dimension of the representation space:

$$v_{\boldsymbol{\theta}}(s) = \boldsymbol{\phi}(\mathbf{s})^\top \boldsymbol{\theta} \approx v_\pi(s), \quad (3)$$

with $\boldsymbol{\theta} \in \mathbb{R}^d$ learnable parameters, and $\boldsymbol{\phi}(s)$ are features.[1] Learning $v_\pi$ with TD methods involves bootstrapping on a *target*, $U_t$, at each timestep $t$, and updating $\boldsymbol{\theta}$ by regressing it towards the target:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \left[U_t - v_{\boldsymbol{\theta}}(S_t)\right] \nabla_{\boldsymbol{\theta}} v_{\boldsymbol{\theta}}(S_t) \,, \quad (4)$$

with learning rate $\alpha$. The TD(0) algorithm (Sutton 1988) uses the *one-step TD return* as the value target:

$$U_t \equiv G_t^{(0)} = R_{t+1} + \gamma v_{\boldsymbol{\theta}}(S_{t+1}) \,. \quad (5)$$

The forward view of TD($\lambda$) constructs the $\lambda$-*return* target—a geometrically weighted average over all possible multi-step returns (Sutton and Barto (2018), chapter 12.1):

$$U_t \equiv G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)} \,, \text{ with} \quad (6)$$

$$G_t^{(n)} \equiv \left(\sum_{k=1}^{n} \gamma^{k-1} R_{t+k}\right) + \gamma^n v_{\boldsymbol{\theta}}(S_{t+n}) \,, \quad (7)$$

where $\lambda \in [0, 1]$ controls the weight of value estimates from the distant future, interpolating between the one-step return (equation (5)) ($\lambda = 0$) and the Monte Carlo return ($\lambda = 1$). The $\lambda$-return can only be computed offline at the end of an episode, since it requires the entire future trajectory to calculate the multi-step returns.

[1] $\boldsymbol{\phi}(s)$ can be a non-linear function jointly learned with $\boldsymbol{\theta}$, as is the case for many deep reinforcement learning algorithms.

### 2.3 Successor Features (SF)

Previous work (Dayan 1993; Kulkarni et al. 2016; Zhang et al. 2017; Barreto et al. 2017, 2018) has shown it can be useful to decouple the reward and transition information of the value function by factorizing it into immediate rewards and SF. The SF, $\boldsymbol{\psi}_\pi : \mathbb{R}^d \to \mathbb{R}^d$, are defined as the expected cumulative discounted features under a policy $\pi$:

$$\boldsymbol{\psi}_\pi(s) \equiv \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} \gamma^n \boldsymbol{\phi}_{t+n} \,|\, S_t = s\right] \,, \quad (8)$$

and can be learned by TD learning algorithms, similar to the standard value function:

$$\boldsymbol{\psi}_{\boldsymbol{\Xi}}(s) = \boldsymbol{\Xi}^\top \boldsymbol{\phi}(s) \approx \boldsymbol{\psi}_\pi(s) \,, \text{with} \quad (9)$$

$$\boldsymbol{\Xi}' = \boldsymbol{\Xi} + \alpha \, \delta_{\boldsymbol{\Xi}} \, \nabla_{\boldsymbol{\Xi}} \boldsymbol{\psi}_{\boldsymbol{\Xi}}(S_t) \quad (10)$$

$$\delta_{\boldsymbol{\Xi}} \equiv \boldsymbol{\phi}(S_t) + \gamma \boldsymbol{\psi}_{\boldsymbol{\Xi}}(S_{t+1}) - \boldsymbol{\psi}_{\boldsymbol{\Xi}}(S_t) \,, \quad (11)$$

with $\boldsymbol{\Xi} \in \mathbb{R}^{d \times d}$ (learnable) parameters.[2] An alternative approach to the direct representation of value (equation (3)) is to used a factorization of SF and instantaneous reward:

$$v_{\boldsymbol{\psi}}(s) \equiv \boldsymbol{\psi}_{\boldsymbol{\Xi}}(s)^\top \mathbf{w} \approx v_\pi(s) \,, \text{ with} \quad (12)$$

$$r_{\mathbf{w}}(s) \equiv \boldsymbol{\phi}(s)^\top \mathbf{w} \approx \mathbb{E}_\pi[R_{t+1} | S_t = s] \,, \quad (13)$$

the instantaneous reward function with (learnable) parameters $\mathbf{w} \in \mathbb{R}^d$.

## 3 The $\eta$-Return Mixture

We take inspiration from the canonical $\lambda$-return (equation (6)), to write a similar quantity.[3] A full derivation of this section is given in the appendix.

$$G_t^\eta \approx R_{t+1} + \gamma \Big[(1-\eta) \sum_{n=1}^{\infty} (\eta\gamma)^{n-1} v_{\boldsymbol{\theta}}(S_{t+n}) + \eta \sum_{n=1}^{\infty} (\eta\gamma)^{n-1} r_{\mathbf{w}}(S_{t+n})\Big] \,. \quad (14)$$

As both $v_{\boldsymbol{\theta}}$ (equation (3)) and $r_{\mathbf{w}}$ (equation (13)) are linear in features, we can express the geometric sums in equation (14) using $\eta\gamma$-discounted SFs,

$$\boldsymbol{\psi}^\eta(s) \equiv \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} (\eta\gamma)^n \boldsymbol{\phi}_{t+n} \mid S_t = s\right] \,. \quad (15)$$

We can separately estimate this SF-model using equation (10). Further, we can use the SF-model in the bootstrapping process by substituting equation (15) into equation (14). This yields a learning target which uses predictive features ($\boldsymbol{\psi}^\eta$), along with a *mixture* of value ($\boldsymbol{\theta}$) and reward ($\mathbf{w}$) parameters. This is the **$\eta$-return mixture**:

$$U_t \equiv G_t^\eta \equiv R_{t+1} + \gamma \boldsymbol{\psi}^\eta(S_{t+1})^\top \left[(1-\eta)\boldsymbol{\theta} + \eta \mathbf{w}\right] \,. \quad (16)$$

This target can be used to replace e.g. the standard TD(0) backup target from equation (5). Despite its similarity to the standard $\lambda$-return, the $\eta$-return mixture does not assume access to a full episodic trajectory.

[2] Unless stated otherwise, we consider $\boldsymbol{\psi}$ as a linear function of features (equation (9)), though non-linear functions are available (Zhang et al. 2017; Machado, Bellemare, and Bowling 2020).

[3] We replaced $\lambda$ with $\eta$ to denote the different properties of the interpolation parameter $\eta$ compared to $\lambda$.

**Interpretation** Consider learning using single-step transition tuple $(S_t, A_t, R_{t+1}, S_{t+1})$. TD(0) propagates information locally from $S_{t+1}$ to $S_t$ by constructing a *bootstrapping target*. Using the value function in the target (equation (5)) propagates only *value information*; bootstrapping using the product of estimated SF and instantaneous rewards (equation (12)) relies on separately learning the SF, which also uses TD(0), and thus propagates only *feature information*. We can more effectively use the same single-step of experience if we simultaneously use the sampled information to predict both the value *and* the features, and update the value function using a mixture of both in the way specified in equation (16).

**Fixed-point solution** With accurate SF and instantaneous reward models, one-step value-learning with the $\eta$-return mixture as bootstrapping target has the same fixed-point solution as the standard TD(0) target, per the following.

**Proposition 1.** *Assume the SF parameters $\boldsymbol{\Xi}$ have converged to their fixed-point solution, $\boldsymbol{\Xi}_{TD(0)} = \mathbb{E}_{d_\pi}[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t - \eta\gamma\boldsymbol{\phi}_{t+1})^\top]^{-1}\mathbb{E}_{d_\pi}[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top]$, and the instantaneous reward parameters have achieved the optimal solution $\mathbf{w} = \mathbb{E}_{d_\pi}[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top]^{-1}\mathbb{E}_{d_\pi}[\boldsymbol{\phi}_t R_{t+1}]$, where $\mathbb{E}_{d_\pi}[\cdot]$ denotes the expectation over the stationary distribution $d_\pi$ for policy $\pi$, which we assume exists under mild conditions (Tsitsiklis and Van Roy 1997). Then, value learning using the $\eta$-return mixtureas the target has the TD(0) fixed point solution:*

$$\boldsymbol{\theta}^*_\eta = \mathbb{E}_{d_\pi}\left[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})^\top\right]^{-1}\mathbb{E}_{d_\pi}[\boldsymbol{\phi}_t R_{t+1}] = \boldsymbol{\theta}_{TD(0)}. \tag{17}$$

*Proof.* In the appendix. Follows from the linearity of the policy evaluation equations. □

Furthermore, it has been shown that on-policy planning with linear models converges to the same fixed point as direct linear value estimation (Schoknecht 2002; Parr et al. 2008; Sutton et al. 2008). However, despite the fact that the fixed point solution is subject to the same bias as one-step TD methods, our method may still benefit from substantial learning efficiency while moving towards this solution. In fact, our finite sample empirical evaluation shows exactly this.

**Interpolating between value and feature prediction with $\eta$** Similar to how $\lambda$-return interpolates between the one-step TD and Monte-Carlo returns, the $\eta$-return mixture interpolates between bootstrapping on the "value-predictive" parameters of the value function, or on the "feature-predictive" parameters of the SF.

When $\eta = 0$, the $\eta$-return mixture recovers the standard TD(0) learning target (equation (5)):

$$\begin{aligned} G_t^{\eta=0} &= R_{t+1} + \gamma\boldsymbol{\psi}^{\eta=0}(S_{t+1})^\top\left((1-0)\boldsymbol{\theta} + 0\mathbf{w}\right) \\ &= R_{t+1} + \gamma\boldsymbol{\phi}_{t+1}^\top\boldsymbol{\theta}\,. \end{aligned} \tag{18}$$

At the opposite end of the spectrum, when $\eta = 1$, the $\eta$-return mixture relies on the full SF (equation (12)) and the instantaneous reward model, akin to using an implicit infinite model:

$$\begin{aligned} G_t^{\eta=1} &= R_{t+1} + \gamma\boldsymbol{\psi}^{\eta=1}(S_{t+1})^\top\left((1-1)\boldsymbol{\theta} + 1\mathbf{w}\right) \\ &= R_{t+1} + \gamma\boldsymbol{\psi}_{t+1}^\top\mathbf{w}\,. \end{aligned} \tag{19}$$

Consequently, the $\eta$-return mixture is a simple generalization that spans the spectrum of learning target parameterizations using $\eta \in [0, 1]$, with the traditional learning target and the SF factorization as extremes.

Compared to the standard learning target used in TD(0), **the $\eta$-return mixture with an intermediate value of $\eta$ ($0 < \eta < 1$) uses information more effectively** than the extremes $G_t^{\eta=0}$ (equation (5)), and $G_t^{\eta=1}$, approximating the true value faster given the same amount of data (see figure 1 for an intuitive illustration).

### 3.1 Estimating the $\eta$-Return Mixture

There are different choices with respect to how the learning target is estimated, depending on (i) the form or elements used in building the target; (ii) the parametrization of the elements making up the target; (iii) the learning methods used to estimate the elements of the target.

Regarding (i), the form of the $\eta$-return mixture target requires access to SF, instantaneous rewards, and the value parameters themselves. Regarding (ii), we parameterize all these estimators as linear functions of features, and share feature parameters in cases where the feature representation is learned and not given (e.g. in the nonlinear control empirical experiments).

With respect to (iii), we can use any learning method for estimating the SF model $\boldsymbol{\psi}^\eta_{\boldsymbol{\Xi}}$ and the instantaneous reward model $r_{\mathbf{w}}$. In this paper, we make the choice of using TD(0) to learn the SF model, and supervised regression for the reward model, since one-step methods are ubiquitous in contemporary RL, and require the use of only *single-step* transitions (Mnih et al. 2015; van Hasselt, Guez, and Silver 2015; Lillicrap et al. 2015; Wang et al. 2016; Schaul et al. 2015; Haarnoja et al. 2018). Likewise, we use the $\eta$-return mixture as a one-step bootstrap target (equation (16)) for estimating of the value parameters $\boldsymbol{\theta}$ (equation (4)). Although we have chosen to focus here on one-step learning targets for their simplicity and ease of use, these methods can be extended to multi-step targets (e.g. TD($n$) or TD($\lambda$)) analogously as the one-step target.

All components of the $\eta$-return mixture are now learnable with one-step transitions tuples of the form $(S_t, A_t, R_{t+1}, S_{t+1})$, which make these methods amenable to both the online setting and the i.i.d. setting. In the former, the algorithm is presented with an infinite sequence of state, actions, rewards $\{S_0, A_0, R_1, S_1, A_1, R_2, \dots\}$, where $A_t \sim \pi(S_t), R_{t+1} \equiv r(S_t, A_t), S_{t+1} \sim P(S_t, A_t)$. In the i.i.d. setting, the learner is presented with a set of transition tuples $\{(S_t, A_t, R_{t+1}, S_{t+1})\}_{t\geq 0}$.

From an algorithmic perspective, we now describe a computationally congenial way for learning the value function online, from a single stream of experience, using our method. As mentioned, in the online setting, the agent has access to experience in the form of tuples

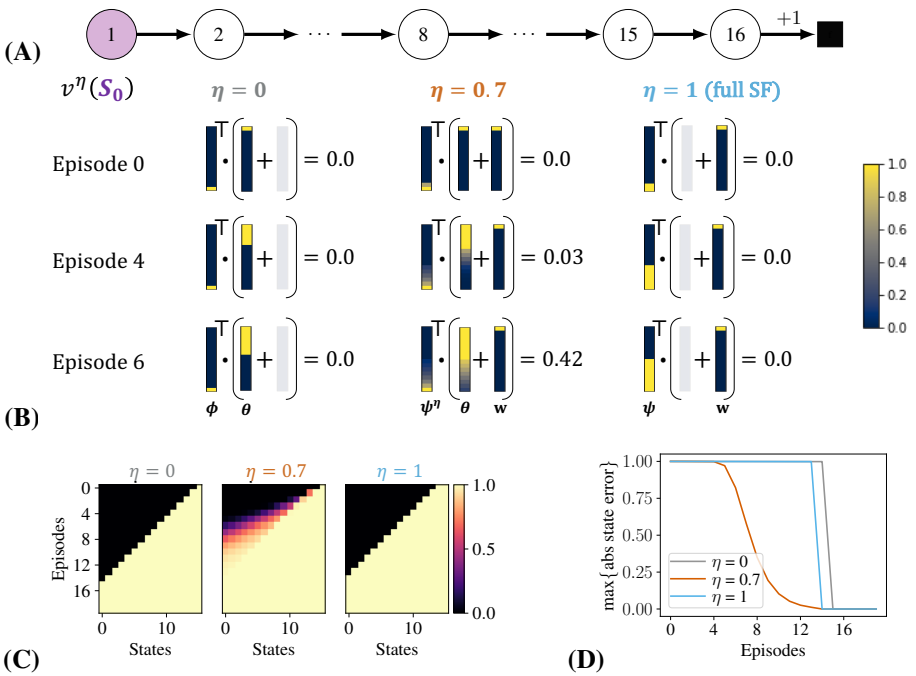


Figure 1: Online value prediction in a deterministic MRP for different $\eta$'s. (A) The agent starts in the left-most state ($s_0$) and deterministically transitions right until reaching the terminal state. All rewards are 0, except for the transition into the terminal state when it is $+1$. (B) Parameter dynamics: The table shows how the $\eta$-return mixture value estimate for the first state, $v^\eta(S_0)$, is computed using $\boldsymbol{\psi}^\eta(s_0)^\top$, $\boldsymbol{\theta}$ and $\mathbf{w}$ over the course of training, for different values of $\eta = \{0.0, 0.7, 1.0\}$. For $\eta = 0.7$ (center) the estimation for the $\eta$-return mixture combines the parameters of the value function ($\boldsymbol{\theta}$) and the SF predictions ($\boldsymbol{\psi}^\eta$) to more quickly propagate value information than either extremes. (C) The estimated value function for all states (columns) across learning episodes (rows). For $\eta = 0.7$ value information propagates faster than $\eta = 0$ and 1. (D) Absolute value error: for different $\eta$ values over episodes. For $\eta = 0.7$ error reduction is faster.

$(S_t, A_t, R_{t+1}, S_{t+1})$ at each timestep $t$. The pseudo-code in algorithm 1 describes the online value estimation process, for the linear case, with given representations.

Algorithm 1: Value prediction using a linear $\eta$-return mixture

**Input**: Given $v_{\boldsymbol{\theta}}(s) = \boldsymbol{\phi}(s)^\top \boldsymbol{\theta}$ (value function), $r_{\mathbf{w}}(s) = \boldsymbol{\phi}(s)^\top \mathbf{w}$ (instantaneous reward function), $\boldsymbol{\psi}^\eta_{\boldsymbol{\Xi}}(s) = \boldsymbol{\Xi}^\top \boldsymbol{\phi}(s)$ (SF model), $\gamma \in [0, 1)$, $\eta \in [0, 1]$, $\alpha^{\boldsymbol{\theta}}$, $\alpha^{\mathbf{w}}$, $\alpha^{\boldsymbol{\Xi}}$ (learning rates).

**Output**: Value function $v_{\boldsymbol{\theta}} \approx v_\pi$ for a policy $\pi$.

**while** sample one-step experience tuple using $\pi$, $(S_t, A_t, R_{t+1}, S_{t+1})$, **do**

SF learning update: $\boldsymbol{\Xi}_{t+1} \leftarrow \boldsymbol{\Xi}_t + \alpha^{\boldsymbol{\Xi}}_t(\boldsymbol{\phi}(S_t) + \gamma\boldsymbol{\psi}^\eta_{\boldsymbol{\Xi}_t}(S_{t+1}) - \boldsymbol{\Xi}_t{}^\top \boldsymbol{\phi}(S_t))\, \boldsymbol{\phi}(S_t)^\top$

Instantaneous reward learning update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha^{\mathbf{w}}_t(R_{t+1} - \boldsymbol{\phi}(S_t)^\top \mathbf{w}_t)\, \boldsymbol{\phi}(s)$

Next step value estimate: $v^\eta_{t+1} = \boldsymbol{\phi}(S_{t+1})^\top \boldsymbol{\Xi}_{t+1}((1-\eta)\boldsymbol{\theta}_t + \eta \mathbf{w}_{t+1})$

Value learning update: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}}_t(R_{t+1} + \gamma v^\eta_{t+1} - \boldsymbol{\phi}(S_t)^\top \boldsymbol{\theta}_t)\boldsymbol{\phi}(S_t)$

**end while**

# 4 Empirical Studies

We start with two simple prediction examples to provide intuition about our approach, after which, we verify that our method scales by extending it to a more complex non-linear control setting.

## 4.1 Value Prediction in a Deterministic Chain

**Experiment setup:** Consider the 16-state deterministic Markov reward process (MRP) with tabular features illustrated in figure 1-A. The agent starts in the left-most state ($s_0$), deterministically transitions right to the right-most absorbing state. The reward is 0 everywhere except for the final transition into the absorbing state, where it is $+1$. We apply algorithm 1 to estimate the value function in an online incremental setting. We use a discount factor $\gamma = 0.9999$ and learning rate $\alpha = 1.0$.

**Results:** Figure 1-B illustrates the result of combining the successor features model $\boldsymbol{\psi}_{\boldsymbol{\Xi}}$, with the value parameters $\boldsymbol{\theta}$, and reward parameters $\mathbf{w}$ into a prediction of the $\eta$-return mixture for the starting state $s_0$, $v^\eta(s_0)$, for different values of $\eta$. When completely relying on the canonical value bootstrap target ($\eta = 0$, recovering TD(0)), we have $v^{\eta=0} = v_{\boldsymbol{\theta}}$, which corresponds to an unchanging feature representation. In this setting, the value information (in $\boldsymbol{\theta}$) moves *backward* one state per episode. For the opposite end, when bootstrap-

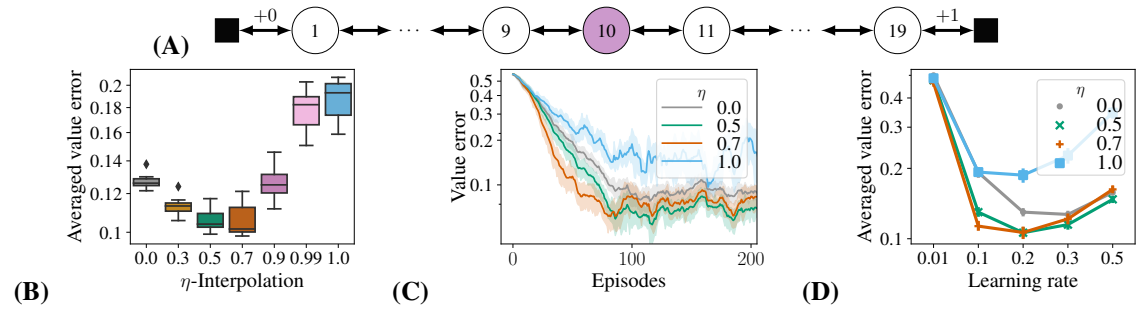


Figure 2: Policy evaluation in 19-state tabular random chain. (A) The agent starts in the center and transitions left/right randomly until either end is reached. Reward is 0 on all transitions, except the on the right-side termination, which yields a reward of $+1$. (B) Parameter study for $\eta$: The y-axis shows the root mean squared error (RMSE) (minimized over learning rates for each $\eta$) averaged over first 400 episodes. (C) Learning dynamics: The y-axis shows the RMSE for four illustrative $\eta$ values. (D) Parameter study for the learning rate The y-axis shows the RMSE for four illustrative $\eta$ values, across different learning rates. Results averages over first 400 episodes. Error bars and shaded areas denote 95 confidence intervals (some too small to see), with 10 independent seeds.

ping on the full successor features ($\eta = 1$), the instantaneous reward is learned immediately (parameter $\mathbf{w}$) for the final state, while the successor features (parameter $\boldsymbol{\psi}$) learns about one additional future state per episode. For both cases, we require $\sim 16$ episodes for the information to propagate across the entire chain and for the value estimate of $s_0$ to improve (Figure 1-D). However, with an intermediate value of $0 < \eta < 1$ (Figure 1-B, middle, $\eta = 0.7$ here), we are able to both propagate value information backward by bootstrapping on $\boldsymbol{\theta}$, as well as improve the predictive features (using $\boldsymbol{\psi}^\eta$) to predict further in the forward direction. This results in an improved value estimate much earlier, as we can observe in figure 1-B middle, C middle, and D.

**Interpretation:** In an online prediction setting, using the $\eta$-return mixture (with an intermediate $\eta$: $0 < \eta < 1$), in place of the standard TD(0) learning target, effectively combines both *backward* credit assignment by bootstrapping the value estimates, as well as *forward* feature prediction, to more quickly estimate the correct values.

### 4.2 Value Prediction in a Random Chain

**Experiment setup:** We now switch to a slightly harder setting, a stochastic 19-state chain prediction task, still with tabular features (Sutton and Barto 2018, Example 6.2). The agent starts in the centre (state 10) and randomly transitions left or right until reaching the absorbing states at either end (figure 2-A). The reward is 0 everywhere except upon transitioning into the right-most terminal state, when it is $+1$. Hyperparameters were chosen by sweeping over learning rates $\alpha \in \{0.01, 0.1, 0.2, 0.3, 0.5\}$, and mixing parameter $\eta = \{0.0, 0.3, 0.5, 0.7, 0.9, 0.99, 1.0\}$. Figure 2-B,D illustrate value error averaged over the first 400 episodes.

**Results:** In figure 2-B, we observe that mixing with $\eta \in [0, 1]$ results in a U-shape error curve, illustrating that an intermediate value of $\eta$ is optimal. For each value of $\eta$, we plot the optimal learning rate $\alpha$. Figure 2-C further confirms our hypothesis that an intermediate value (here for $\eta = 0.5$ or 0.7) is most efficient. We also observe that intermediate $\eta$'s show a degree of parameter robustness, having low value error over a range of different learning rates (figure 2-D).

**Interpretation:** Using the $\eta$-return mixture as one-step learning target is robust to environment stochasticity and learns most efficiently for intermediate values of $\eta$.

### 4.3 Value-Based Control in Mini-Atari

We hypothesize that efficient value prediction using the $\eta$-return can help in value-based control, so we extend our proposed algorithm to the control setting, simply by estimating the action-value function $q_{\boldsymbol{\theta}}$ using the $\eta$-return mixture. We build on top of the deep Q network (DQN) architecture (Mnih et al. 2015), and simply replace the bootstrap target with an estimate of the $\eta$-return mixture starting from a *state and action*.

Given a sampled transition $(S_t, A_t, R_{t+1}, S_{t+1})$, DQN encodes features $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$, then estimates the action-values $q_{\boldsymbol{\theta}}(\boldsymbol{\phi}_t, A_t) \approx q(S_t, A_t)$ using the canonical bootstrap target in which it relies on the next value estimate, $\max_{a'} q_{\boldsymbol{\theta}}(\boldsymbol{\phi}_{t+1}, a')$, with $\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}(S_{t+1})$. We use the same feature encoding $\boldsymbol{\phi}(\cdot)$ to track the successor features of the current policy $\boldsymbol{\psi}^\eta_t = \boldsymbol{\psi}^\eta_{\boldsymbol{\Xi}}(\boldsymbol{\phi}_t) \approx \boldsymbol{\psi}(\boldsymbol{\phi}_t)$, and estimate the instantaneous rewards $r_{\mathbf{w}}(\boldsymbol{\phi}_t)$. This allows us to construct the $\eta$-return mixture and use it in the learning target of Q-learning when updating the parameters $\theta$:

$$q^\eta_{t+1} \equiv (1-\eta)\, q_{\boldsymbol{\theta}}\big(\boldsymbol{\psi}^\eta_{\boldsymbol{\Xi}}(S_{t+1}), a'\big) + \eta\, r_{\mathbf{w}}\big(\boldsymbol{\psi}^\eta_{\boldsymbol{\Xi}}(S_{t+1})\big)\,, \tag{20}$$

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha\big(R_{t+1} + \gamma \max_{a'} q^\eta_{t+1} - q_{\boldsymbol{\theta}}(\boldsymbol{\phi}_t, A_t)\big)\, \nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}\,, \tag{21}$$

where $q^\eta$ is the value estimate of the $\eta$-return mixture used in the learning target, and $\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\boldsymbol{\phi}_t, A_t)$. We simultaneously estimate the feature representation and the action-values in an end-to-end fashion. See appendix algorithm for a complete description.

**Experiment Set-up:** We test our algorithm in the Mini-Atari (MinAtar, Young and Tian (2019), GNU General Public License v3.0) environment, which is a smaller version of

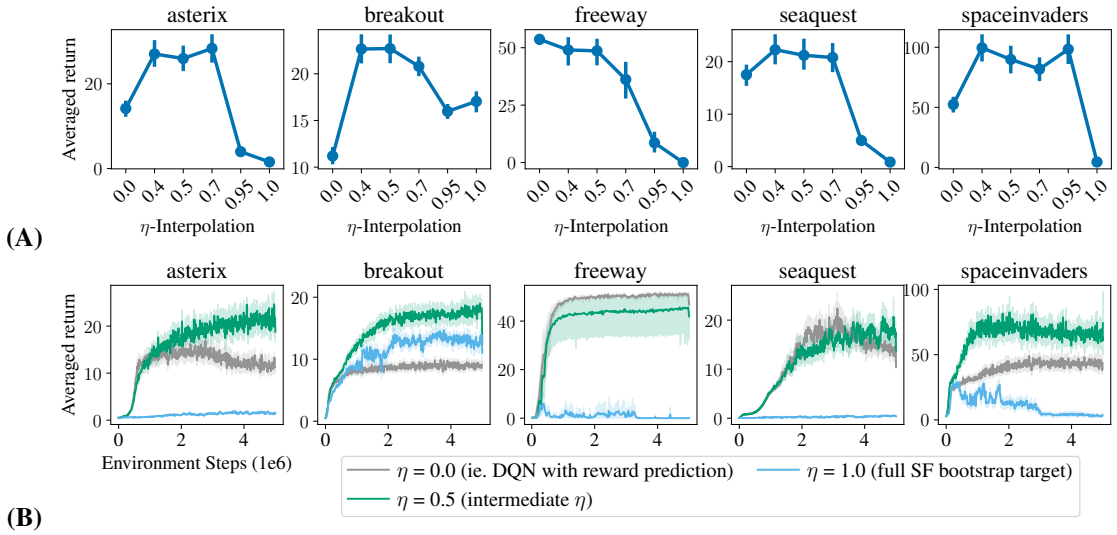


Figure 3: Performance for value-based control in Mini-Atari. (A) Parameter study for different values of $\eta$. The y-axis shows the average performance over 10k timesteps and 10 seeds using an $\epsilon$-greedy policy with $\epsilon = 0.05$, after stopping training after $5e6$ learning steps. (B) Learning curves for 3 illustrative $\eta$ values over the course of training. The y-axis displays the average return over 10 independent seed. Shaded area and error bars depicts 95 confidence interval.

the Arcade Learning Environment (Bellemare et al. 2013) with 5 games (`asterix`, `breakout`, `freeway`, `seaquest`, `space invaders`) played in the same way as their larger counterparts. Other than the architectural update to the bootstrap target, we make no other changes (e.g. to policy, relay buffer, etc.). Unless otherwise stated, we use the same hyperparameters as DQN version from Young and Tian (2019). Details on environment, algorithms and hyperparameters can be found in the appendix.

**Intermediate $\eta$ improves nonlinear control.** Figure 3-A illustrates a parameter study on the mixing parameter $\eta$ after training for 5 million environmental steps. We again observe the U-shaped performance curve as we interpolate across $\eta$, confirming the advantage of using an intermediate $\eta$ value. Figure 3-B shows the learning curves of our proposed model that uses an intermediate value of $\eta$ in comparison to the two baseline algorithms: bootstrapping entirely on the value parameters ($\eta = 0$, equivalent to vanilla DQN with a reward prediction auxiliary loss), and bootstrapping entirely on the full SF value ($\eta = 1$). The latter baseline is remarkably unstable, while the $\eta$-return mixture, with an intermediate $\eta = 0.5$, outperforms both in $4/5$ games, and is competitive with $\eta = 0$ in `freeway`. The poor performance for higher $\eta$ values in `freeway` is likely due to *sparse reward*, as the reward gradient used to shape the representation $\phi(\cdot)$ is uninformative most of the time, leading to a collapse in representation (this is explicitly measured in the appendix). This highlights a weakness of learning the feature encoding and SF simultaneously, since poor features result in poor SF, and thus poor value estimates. The use of auxiliary losses can help ameliorate this issue (Machado, Bellemare, and Bowling 2020; Kumar et al. 2020), although it is not explored here as we found the issue to only be significant for high values of $\eta$.

**Parameter study: robustness to the learning rates of the SF and instantaneous reward models.** Figure 4 shows parameter studies for an intermediate $\eta$ that illustrate the sensitivity to the learning rates of the successor features and reward heads used in learning the value function. We vary the learning rates for these estimators while keeping the learning rates of the representation torso and the value function head fixed (at the same values used by Young and Tian (2019): $\alpha_{\boldsymbol{\theta}} = 2.5\text{e-}4$). We observe that performance is not highly dependent on the SF and reward learning rates (figure 4, green), but a higher learning rate for the SF than the one used by the representation torso facilitates tracking the changes in the feature representations ($\phi$) by the SF. This choice is important in `freeway`. For comparison, we also sweep over the value and encoder learning rates of a vanilla DQN (figure 4, blue), and see that it is sensitive to the learning rate, i.e. performance drops as learning rate settings deviate from the recommendation of Young and Tian (2019) (most prominently observed in `asterix`, `seaquest` and `space_invaders`, and for high learning rates in `breakout`). Additionally, we also sweep over the learning rates of *all* parameters making up the $\eta$-return mixture used as target for the q-function: either keeping all learning rates the same (figure 4, brown) or setting the successor feature and reward learning rates to be $10\times$ the encoder learning rates (figure 4, pink). Overall, we again observe that the agent is most sensitive to learning rates in the value head and encoder torso: performance decreases in all games other than `breakout`.

## 5 Related Work

**Successor features** (SF, equation (8)) are an extension to state-based successor representations (Dayan 1993), allowing feature-based value functions to be factorized using a

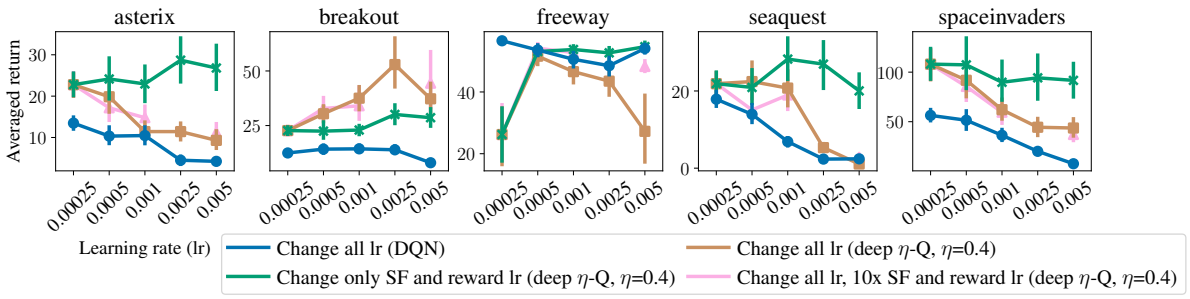


Figure 4: Parameter study on the learning rates of the SF and instantaneous reward model: The y-axis shows the average return over 10k evaluation steps using an $\epsilon$-greedy policy with $\epsilon = 0.05$, after stopping training after $5e6$ steps. For our algorithm, shown here as the Deep $\eta$-Q algorithm (green), we sweep over the SF and reward learning rates while keeping the learning rates for the representation torso and the value function head fixed at $0.00025$. For the vanilla DQN (blue), we vary the learning rates of the representation torso and the value function head. We also show the cumulative sensitivity to the parameters as we vary all the learning rates in our algorithm (brown). Error bar denote 95 confidence intervals and each setting is ran using 3 independent seeds.

separately parameterized policy-dependent transition model and an instantaneous reward model (Kulkarni et al. 2016; Lehnert and Littman 2020). A wide variety of uses have been proposed for the SF: aiding in exploration (Janz et al. 2019; Machado, Bellemare, and Bowling 2020), option discovery (Machado, Bellemare, and Bowling 2017; Machado et al. 2018), and transferring across multiple goals (Lehnert, Tellex, and Littman 2017; Zhang et al. 2017; Ma et al. 2020; Brantley, Mehri, and Gordon 2021), in particular through the generalized policy improvement framework (Barreto et al. 2017, 2018; Borsa et al. 2018; Hansen et al. 2019; Grimm et al. 2019). Our method adds to this repertoire, by using the SF inside the learning target in bootstrapping methods.

**Forward model-based planning** can facilitate efficient credit assignment. Among the algorithms that address this topic, are Dyna-style methods, which use explicit models to generate fictitious experience, that they then leverage to improve the value function (Schoknecht 2002; Parr et al. 2008; Sutton et al. 2008; Yao et al. 2009b). Closest to our method is the work by Yao et al. (2009a,b) which learns an explicit $\lambda$-model and uses it to generate fictitious experience for $k$-step updates to the value function. Our work is different in that the model we use is an implicit model, used to construct a learning target. Particularly, the SF here are akin to models for implicit planning, aiding in speeding up the value learning process within a *single-task* setup. Furthermore, we extend our method to learned non-linear feature representations and combine it with batch learning algorithms (DQN) in MinAtar.

**Building state representation** is fundamental for deep RL. The SF model is a type of *general value function* (Sutton et al. 2011), hypothesized to be a core component in building internal representations of autonomous agents (Sutton et al. 2011; White et al. 2015; Schlegel et al. 2021). Our work ties to this topic, since we can view the partial SF model as a new *learned* representation of the value function used as learning target in the $\eta$-return mixture.

## 6 Discussion

In this work we propose a new, generalized learning target that combines the previous approaches, making more efficient use of the same experience. The approach we proposed uses an implicit model represented by the SF model, and can thus also be viewed as implicit planning with a multi-step policy-dependent expectation model. The $\eta$-return mixture we proposed for the learning target can easily be used in place of the bootstrap target used in any value-based algorithm (e.g. TD($n$), TD($\lambda$)), as we have illustrated in this work for one-step returns used by TD(0). Empirically, we showed that this method, while using the same amount of sampled experience, is more effective, resulting in more efficient value function estimation and higher control performance.

Many potential directions of investigation have been left for future work. (i) The $\eta$-return mixture contains a successor feature estimate, which could also be further leveraged for exploration and transfer. (ii) Chelu, Precup, and Hasselt (2020) investigates the complementary properties of explicit forward and backward models and argues for the potential of optimally combining both "forward" and "backward" facing credit assignment schemes. Further, van Hasselt et al. (2020) introduces expected eligibility traces as implicit backward models, a kind of "predecessor features" (time-reversed successor features). Future work can explore the differences and commonalities between implicit models in the forward and backward direction using our proposed SF model and expected eligibility traces. The right balance between using *backward* credit assignment through the use of eligibility traces, and *forward* prediction through predictive representations remains an open question with fundamental implications for learning efficiency. (iii) How to best use predictive representations to build an internal agent state is central to generalization and efficient credit assignment. Our work opens up many exciting new questions for investigation in this direction.

## Acknowledgments

AGXC was supported by the NSERC CGS-M, FRQNT, and UNIQUE excellence scholarships. This work was supported by NSERC (Discovery Grant: RGPIN-2020-05105; Discovery Accelerator Supplement: RGPAS-2020-00031) and CIFAR (Canada AI Chair; Learning in Machine and Brains Fellowship) grants to BAR. VC was partially supported by a Borelis AI fellowship. This research was enabled in part by computational resources provided by Calcul Québec (www.calculquebec.ca) and Compute Canada (www.computecanada.ca). We thank the anonymous reviewers for their valuable feedback. We thank our colleagues at Mila for the insightful discussions that have made this project better.

## References

Barreto, A.; Borsa, D.; Quan, J.; Schaul, T.; Silver, D.; Hessel, M.; Mankowitz, D.; Zidek, A.; and Munos, R. 2018. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, 501–510. PMLR.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, 4055–4065.

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

Borsa, D.; Barreto, A.; Quan, J.; Mankowitz, D.; Munos, R.; van Hasselt, H.; Silver, D.; and Schaul, T. 2018. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*.

Brantley, K.; Mehri, S.; and Gordon, G. J. 2021. Successor Feature Sets: Generalizing Successor Representations Across Policies. *arXiv preprint arXiv:2103.02650*.

Chelu, V.; Precup, D.; and Hasselt, H. V. 2020. Forethought and Hindsight in Credit Assignment. *ArXiv*, abs/2010.13685.

Dayan, P. 1993. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4): 613–624.

Grimm, C.; Higgins, I.; Barreto, A.; Teplyashin, D.; Wulfmeier, M.; Hertweck, T.; Hadsell, R.; and Singh, S. 2019. Disentangled cumulants help successor representations transfer to new tasks. *arXiv preprint arXiv:1911.10866*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 1861–1870. PMLR.

Hansen, S.; Dabney, W.; Barreto, A.; Van de Wiele, T.; Warde-Farley, D.; and Mnih, V. 2019. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*.

Janz, D.; Hron, J.; Mazur, P.; Hofmann, K.; Hernández-Lobato, J. M.; and Tschiatschek, S. 2019. Successor uncertainties: exploration and uncertainty in temporal difference learning. *Advances in Neural Information Processing Systems*, 32: 4507–4516.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Kumar, A.; Agarwal, R.; Ghosh, D.; and Levine, S. 2020. Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning. *arXiv preprint arXiv:2010.14498*.

Lehnert, L.; and Littman, M. L. 2020. Successor features combine elements of model-free and model-based reinforcement learning. *Journal of Machine Learning Research*, 21(196): 1–53.

Lehnert, L.; Tellex, S.; and Littman, M. L. 2017. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Ma, C.; Ashley, D. R.; Wen, J.; and Bengio, Y. 2020. Universal successor features for transfer reinforcement learning. *arXiv preprint arXiv:2001.04025*.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. 2017. A Laplacian Framework for Option Discovery in Reinforcement Learning. *ArXiv*, abs/1703.00956.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. 2020. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5125–5133.

Machado, M. C.; Rosenbaum, C.; Guo, X.; Liu, M.; Tesauro, G.; and Campbell, M. 2018. Eigenoption Discovery through the Deep Successor Representation. *ArXiv*, abs/1710.11089.

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. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Parr, R. E.; Li, L.; Taylor, G.; Painter-Wakefield, C.; and Littman, M. 2008. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *ICML '08*.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA: John Wiley & Sons, Inc., 1st edition. ISBN 0471619779.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schlegel, M.; Jacobsen, A.; Abbas, Z.; Patterson, A.; White, A.; and White, M. 2021. General value function networks. *Journal of Artificial Intelligence Research*, 70: 497–543.

Schoknecht, R. 2002. Optimality of Reinforcement Learning Algorithms with Linear Function Approximation. In *NIPS*.

Sutton, R. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3: 9–44.

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

Sutton, R.; Szepesvari, C.; Geramifard, A.; and Bowling, M. 2008. Dyna-Style Planning with Linear Function Approximation and Prioritized Sweeping. In *UAI*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tsitsiklis, J. N.; 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.; Guez, A.; and Silver, D. 2015. Deep reinforcement learning with double Q-learning. CoRR abs/1509.06461 (2015). *arXiv preprint arXiv:1509.06461*.

van Hasselt, H.; Madjiheurem, S.; Hessel, M.; Silver, D.; Barreto, A.; and Borsa, D. 2020. Expected Eligibility Traces. *arXiv preprint arXiv:2007.01839*.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

White, A.; et al. 2015. *Developing a predictive approach to knowledge*. Ph.D. thesis, University of Alberta.

Yao, H.; Sutton, R.; Bhatnagar, S.; Diao, D.; and Szepesvári, C. 2009a. Dyna (k): A Multi-Step Dyna Planning. *Abstraction in Reinforcement Learning*, 54.

Yao, H.; Sutton, R. S.; Bhatnagar, S.; Dongcui, D.; and Szepesvári, C. 2009b. Multi-step dyna planning for policy evaluation and control. In *NIPS*.

Young, K.; and Tian, T. 2019. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*.

Zhang, J.; Springenberg, J. T.; Boedecker, J.; and Burgard, W. 2017. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2371–2378. IEEE.