Gated Q-learning: Add Off-Policy Bias to Taste

Brett Daley

Keywords: Q-learning, Off-Policy Learning, Bias-Variance Trade-Off, Multistep Returns, Eligibility Traces.

Summary

Multistep credit assignment is critical for sample-efficient reinforcement learning, yet managing off-policy bias in Q-learning remains a fundamental challenge. For 30 years, practitioners have been limited to a binary choice: eliminate the bias at the cost of severely truncated eligibility traces (Watkins' Q(λ)), or ignore the bias to learn faster while injecting detrimental errors into the value estimates (Peng's Q(λ)). Modern off-policy estimators fail to resolve this tension, as importance-sampling ratios collapse under Q-learning's greedy target policy. We introduce Gated Q-learning, a novel algorithmic framework that ends this dilemma by smoothly interpolating between the two historical extremes. Rather than relying on importance sampling, our approach employs a continuous, state-action-dependent gating mechanism to selectively attenuate eligibility traces in an exploration-aware manner. We provide a rigorous theoretical foundation for this mechanism, proving that the expected operator remains a contraction mapping and deriving its exact fixed point. Empirical evaluations verify that intermediate gating safely enables longer credit-assignment horizons, yielding faster initial learning than either extreme. Gated Q-learning offers a simple alternative to importance sampling while enabling customization of the effective multistep horizon and the amount of off-policy bias in Q-learning agents.

Contribution(s)

1. We identify a new, general class of Q(λ) algorithms that utilize state-action-dependent trace-decay values. This offers a novel perspective on partial off-policy bias correction in Q-learning methods, where importance sampling cannot be applied.
   Context: State-action-dependent traces for Expected Sarsa have been previously studied to control off-policy bias in conjunction with importance sampling (e.g., Munos et al., 2016; Sutton & Barto, 2018, Ch. 12.8). To the best of our knowledge, this idea has never been explored in Q-learning beyond Watkins' Q(λ) (Watkins, 1989), which applies exploration-conditional trace cuts to eliminate off-policy bias.
2. We propose Gated Q(λ), which implements soft, exploration-conditional trace cuts to mitigate off-policy bias while preserving multistep credit assignment. We very briefly discuss an n-step version as well.
   Context: Gated Q(λ) interpolates smoothly between the classic methods of Watkins' Q(λ) (Watkins, 1989) and Peng's Q(λ) (Peng & Williams, 1996).
3. We conduct a large-scale hyperparameter sweep in a random-walk environment adapted for off-policy control, generating detailed heatmaps to visualize the influence of step size, trace decay, and gating on Gated Q(λ). Our heatmaps clearly illustrate a performance trade-off between Watkins' Q(λ) and Peng's Q(λ).
   Context: Sutton & Barto (2018, Ex. 7.1) describes the 19-state random walk that we adapt for our experiment.
4. We derive the value-function operator underlying this general class of Q(λ) algorithms with state-action-dependent traces, formally proving how the chosen traces impact its contraction rate and fixed point.
   Context: Kozuno et al. (2021) derived similar results for Peng's Q(λ). Our theorems significantly generalize these to the newly identified class of Q(λ) algorithms.

Gated Q-learning: Add Off-Policy Bias to Taste

Brett Daley[1,†]
brett@brettdaley.ai

[1]**Meta**

[†] All experiments, data collection, and processing activities were conducted in the author's personal capacity. No experiments, data collection, or processing activities were conducted on Meta infrastructure.

Abstract

Multistep credit assignment is critical for sample-efficient reinforcement learning, yet managing off-policy bias in Q-learning remains a fundamental challenge. For 30 years, practitioners have been limited to a binary choice: eliminate the bias at the cost of severely truncated eligibility traces (Watkins' Q(λ)), or ignore the bias to learn faster while injecting detrimental errors into the value estimates (Peng's Q(λ)). Modern off-policy estimators fail to resolve this tension, as importance-sampling ratios collapse under Q-learning's greedy target policy. We introduce Gated Q-learning, a novel algorithmic framework that ends this dilemma by smoothly interpolating between the two historical extremes. Rather than relying on importance sampling, our approach employs a continuous, state-action-dependent gating mechanism to selectively attenuate eligibility traces in an exploration-aware manner. We provide a rigorous theoretical foundation for this mechanism, proving that the expected operator remains a contraction mapping and deriving its exact fixed point. Empirical evaluations verify that intermediate gating safely enables longer credit-assignment horizons, yielding faster initial learning than either extreme. Gated Q-learning offers a simple alternative to importance sampling while enabling customization of the effective multistep horizon and the amount of off-policy bias in Q-learning agents.

1 Introduction

Despite its simplicity, Q-learning (Watkins, 1989) remains a staple of modern reinforcement learning (RL). The appeal of Q-learning lies in its theoretical elegance and its decoupling of the behavior policy from the target policy, allowing agents to continuously refine estimates of optimal values while exploring the environment or learning from historical replay buffers. In deep RL specifically, where neural networks serve as function approximators, Q-learning underpins the success of some of the most sample-efficient methods to date, including Deep Q-Networks (DQN; Mnih et al., 2015), Rainbow (Hessel et al., 2018), and Parallel Q-Networks (PQN; Gallici et al., 2025). It also plays a critical role in offline RL (e.g., Fujimoto et al., 2019; Kumar et al., 2019; 2020; Kostrikov et al., 2022), where its off-policy nature is ideal for learning from static datasets. Consequently, advancing the algorithmic foundations of Q-learning directly translates to broader improvements across a vast array of deep RL architectures.

Standard Q-learning is rooted in 1-step temporal-difference (TD) learning (Sutton, 1988), which struggles to assign credit quickly over long time horizons. Multistep learning is crucial for accelerating this process, but the naive application of forward-view return estimators such as n-step returns or λ-returns is strongly biased in off-policy settings. This bias stems from the distributional mismatch between the agent's exploratory behavior and the targeted greedy behavior. The theoretically correct approach is to eliminate this bias by truncating the multistep return estimates whenever an

exploratory action is taken, as in Watkins' Q(λ) (Watkins, 1989). However, empirical evidence indicates that simply *ignoring* these corrections often yields superior performance (Daley & Amato, 2019; Hernandez-Garcia & Sutton, 2018), which is the exact motivation behind Peng's Q(λ) (Peng & Williams, 1996). Practitioners are thus faced with a limited choice: either strictly eliminate the bias at the cost of severe truncation and slower learning, or accept the bias and ultimately limit the length of multistep returns that can be safely deployed.

Although multistep off-policy estimators that enable finer-grained control over off-policy bias do exist, they rely on importance sampling (Kahn & Marshall, 1953) and are therefore not compatible with Q-learning. Key examples include Tree Backup (Precup et al., 2000), Retrace (Munos et al., 2016), and Recency-Bounded Importance Sampling (Daley et al., 2023)—all of which adapt the degree of reinforcement based on the ratio between the action probabilities assigned by the target and behavior policies. However, in Q-learning, the target policy is strictly greedy, causing the importance-sampling ratio to become binary-valued. Consequently, this whole class of methods degenerates into the same aggressive Watkins-style correction, which, as previously mentioned, fails to preserve the long credit-assignment horizons needed for fast learning.

There is a clear need for a new mechanism to regulate off-policy bias in multistep Q-learning, without the use of importance sampling. We propose a novel approach that utilizes adaptive λ-values to partially "gate" the propagation of the eligibility trace specifically when exploratory (non-greedy) actions are taken. This algorithm, which we call *Gated Q(λ)*, mitigates some but not all of the off-policy bias while preserving the trace along greedy trajectories. This strategy interpolates smoothly between the principled (but slow) Watkins' update and the biased (but fast) Peng's update. We hypothesize that balancing this trade-off leads to superior learning compared to either extreme. Gated Q-learning is conceptually analogous to the gating mechanisms found in Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997), Gated Recurrent Units (GRUs; Cho et al., 2014; Chung et al., 2014), and gated attention (Xu et al., 2015; Dhingra et al., 2017), which inspire its name, although its role is distinct—it modulates credit assignment in RL.

Our paper is dedicated to deeply understanding the properties and implications of this gating mechanism in off-policy credit assignment. We primarily focus on eligibility traces and λ-returns, although we briefly discuss an n-step variant as well (see Section 4.2). We first derive Gated Q(λ) as a special case of a new, more general Q(λ) class which permits state-action-dependent λ-values while targeting a greedy policy—the latter being the key differentiator from Munos et al.'s (2016) per-decision operator. This greatly broadens the scope of our theoretical analysis while helping to contextualize and justify the specific choice of our adaptive gating strategy. We then conduct a focused hyperparameter study in a random walk to illustrate how the gating mechanism impacts credit assignment and learning speed. Building upon these empirical insights, we formally analyze Gated Q(λ) to establish its contraction rate and fixed point, providing clear theoretical proof of the trade-off between trace preservation and off-policy bias. Our results demonstrate that there are still fundamental Q-learning improvements to be discovered, and that faster learning and lower asymptotic error can be simultaneously achieved by adjusting off-policy bias to a desired, intermediate amount.

2 Background

The RL problem considers an agent whose objective is to learn to act in its environment in a way that maximizes its expected cumulative discounted reward. Since the foundational work of Watkins (1989), which introduced Q-learning, the RL problem is most commonly framed as solving a Markov Decision Process (MDP) from sample-based interaction. The MDP is typically described by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$. At each discrete time step $t \geq 0$, the agent observes a state $S_t \in \mathcal{S}$ and selects an action $A_t \in \mathcal{A}$ according to a behavior policy $b(a|s)$, which maps states to probability distributions over actions. The environment then transitions to a new state S_{t+1} according to the transition dynamics $p(s' \mid s, a)$, and the agent receives a scalar reward R_{t+1} governed by the reward function $r(s, a)$. The fundamental objective of the agent is to maximize the expected return, defined as the cumulative sum of discounted future rewards, $G_t \coloneqq \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$, where $\gamma \in [0, 1)$ is

the discount factor. In off-policy learning, we explicitly distinguish between this behavior policy (which explores the environment and generates trajectory data) and the target policy $\pi(a|s)$, the distinct policy that the algorithm is attempting to evaluate or optimize.

To measure the quality of a policy π, we define the action-value function $q_\pi(s,a) \coloneqq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$, which represents the expected return for taking action a in state s and subsequently following π. Q-learning aims to learn the optimal action-value function q_* by representing these estimates as a tabular matrix or parameterized function $Q \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$. The distinguishing feature of Q-learning is that its target policy is always *greedy* with respect to its current value estimates. The estimated value of a state is therefore given by

$$V(s) \coloneqq \max_{a\in\mathcal{A}} Q(s,a)\,,$$

where the value of a terminal state is always defined to be 0. The classic 1-step Q-learning update rule is then defined as

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\Big(\underbrace{R_{t+1} + \gamma V(S_{t+1}) - Q(S_t,A_t)}_{\delta'_t}\Big)\,,$$

where $\alpha \in (0,1]$ is the step size. We refer to the quantity δ'_t as the *Q-learning (QL) error* to differentiate it from the classic state-value TD error. Rooted in the Bellman optimality equation (Bellman, 1957), this 1-step update is highly robust and guaranteed to converge to q_* in tabular settings (Watkins & Dayan, 1992). However, because information about future rewards is solely conveyed through immediate bootstrapping, its 1-step nature results in painfully slow credit assignment, as rewards must propagate backward through the state-action space one step at a time over repeated episodic interactions.

To overcome the slow credit assignment, we can consider *multistep* versions of Q-learning by substituting the 1-step target with a generalized multistep return estimator $\hat{G}_t$:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\Big(\hat{G}_t - Q(S_t,A_t)\Big)\,.$$

Defining $\hat{G}_t$ to safely accelerate learning in off-policy settings has proven to be highly nontrivial. If one were to simply use eligibility traces to accumulate a fading record of recent 1-step errors, analogous to on-policy TD(λ) (Sutton, 1988), it would produce the following forward-view target:

$$G_t^{\lambda\,\text{(naive)}} = Q(S_t,A_t) + \sum_{i=0}^{\infty}(\gamma\lambda)^i\delta'_{t+i}\,.$$

This update has become known as *naive* Q(λ) (Sutton & Barto, 1998, Sec. 7.6) because it completely fails to address the off-policy distributional mismatch between the exploratory behavior policy and the greedy target policy. As a result, it suffers from severe bias and does not converge in off-policy settings unless λ is kept impractically small (Harutyunyan et al., 2016), which heavily counteracts the original multistep benefits.

Watkins (1989) recognized that off-policy bias can be completely avoided by cutting the eligibility trace whenever a non-greedy action is taken. Define the following greedy indicator function:

$$\text{greedy}(s,a) \coloneqq a \in \arg\max_{a'\in\mathcal{A}} Q(s,a')\,,$$

where ties are broken arbitrarily but consistently. Written as a forward-view return, Watkins' Q(λ), which cuts traces, generates the following target:

$$G_t^{\lambda\,\text{(Watkins)}} = Q(S_t,A_t) + \sum_{i=0}^{\infty}\gamma^i\left(\prod_{j=1}^{i}\lambda_{t+j}\right)\delta'_{t+i}\,, \tag{1}$$

$$\text{where}\quad \lambda_t = \begin{cases}\lambda & \text{if greedy}(S_t,A_t),\\ 0 & \text{otherwise,}\end{cases}$$

and $\prod_{j=1}^{0} \lambda_{t+j} \coloneqq 1$ to correctly initialize the first weight. Watkins' Q(λ) fully eliminates off-policy bias, and in this sense is the theoretically "correct" implementation of Q-learning with eligibility traces. Unfortunately, because exploratory actions are frequent during training, traces are cut early and often. This severe truncation mostly counteracts the speed benefits of multistep learning, making it seem as though Q-learning is simply at odds with the goals of extended credit assignment.

To circumvent the severe trace cutting of Watkins' method, Peng & Williams (1996) introduced a modified version of Q(λ) that prioritizes rapid credit assignment over strict off-policy correctness. To formalize this, we first define the state-value TD error as

$$\delta_t \coloneqq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\,.$$

Then, Peng's Q(λ) target becomes a hybrid of Q-learning and TD(λ):

$$G_t^{\lambda\,(\text{Peng})} \coloneqq Q(S_t, A_t) + \delta'_t + \sum_{i=1}^{\infty} (\gamma\lambda)^i \delta_{t+i}\,.$$

Unlike Watkins' Q(λ), this estimator never cuts the eligibility trace. While this makes the update strongly biased in off-policy settings, it is empirically much faster. Furthermore, this estimator satisfies an elegant recursive equation:

$$G_t^{\lambda\,(\text{Peng})} = R_{t+1} + \gamma\Big((1-\lambda)V(S_{t+1}) + \lambda G_{t+1}^{\lambda\,(\text{Peng})}\Big)\,,$$

where the recursion is initialized by $V(S_T)$ at the end of a truncated trajectory or R_T at the end of an episode. This makes it very efficient to compute over offline trajectories of experience. As a result, Peng's Q(λ) has become a popular choice in trajectory-based deep RL (e.g., Harb & Precup, 2016; Mousavi et al., 2017; Daley & Amato, 2019; Kozuno et al., 2021; Gallici et al., 2025; Elelimy et al., 2025) and is the most common multistep alternative to the widely used n-step return (e.g., Hessel et al., 2018). However, in both cases, relying on uncorrected bias is fundamentally flawed, as it ultimately limits the safe effective horizon of the multistep return.

In a separate line of research, modern methods for off-policy learning have successfully managed this bias-variance trade-off using variations of importance sampling (Kahn & Marshall, 1953). These include algorithms such as Tree Backup (Precup et al., 2000), Retrace (Munos et al., 2016), and Recency-Bounded Importance Sampling (Daley et al., 2023). While highly effective for off-policy learning, these approaches inherently fall under the Sarsa class of algorithms, meaning they evaluate or optimize stochastic (non-greedy) target policies. They are fundamentally inapplicable to Q-learning due to its greedy target policy, which causes importance-sampling ratios to collapse to either zero or nonzero. This structural limitation precludes the use of modern importance-sampling methods, including resampling (e.g., Schlegel et al., 2019), severely limiting the degree to which off-policy corrections can be controlled in Q-learning. As a consequence, there has been surprisingly little progress on multistep *Q-learning* estimators, leaving practitioners caught between the two extremes of Watkins' and Peng's Q(λ).

3 Variable Q(λ)

Before introducing our Gated Q(λ), we begin by formalizing a more general class of *variable* Q(λ) methods, of which our algorithm is a special case. When applying multistep Q-learning, practitioners have historically been faced with a limited choice: either fully correct the off-policy bias as in Watkins' Q(λ), or ignore the bias as in Peng's Q(λ). As previously established, relying on uncorrected bias is bad because it limits the length of the credit-assignment window that can be safely deployed before value estimates diverge. Furthermore, an ideal multistep Q-learning method must have several properties that make it practically useful: most notably the ability to mediate this bias without sacrificing computational efficiency. To unify these different approaches, we adopt a variable λ framework, which serves as a prime but underexplored candidate for resolving this dilemma.

The concept of state-action-dependent trace parameters has been discussed by Sutton & Barto (2018, Ch. 12.8), but only for *Sarsa* methods and not Q-learning. In our notation, the Q-learning return is

$$\tilde{G}_t^\lambda \coloneqq Q(S_t, A_t) + \delta'_t + \sum_{i=1}^{\infty} \gamma^i \left(\prod_{j=1}^{i} \lambda_{t+j} \right) \delta_{t+i} \,, \tag{2}$$

where the overloaded function $\lambda \colon \mathcal{S} \times \mathcal{A} \to [0, 1]$ now determines $\lambda_t \coloneqq \lambda(S_t, A_t)$. This return admits a recursive form:

$$\tilde{G}_t^\lambda = R_{t+1} + \gamma \Big((1 - \lambda_{t+1}) V(S_{t+1}) + \lambda_{t+1} \tilde{G}_{t+1}^\lambda \Big) \,. \tag{3}$$

The equivalence between Eq. (2) and Eq. (3) follows from a slight generalization of the fixed-λ derivation given by Daley (2025, Sec. 5.2), and it is exactly this recursion that makes the return target efficient to compute, whether with eligibility traces or replayed trajectories. The only instance of this equation that we found is Eq. 12.20 of Sutton & Barto (2018), where it is given for Expected Sarsa but not Q-learning. The key distinction is how the target policies are defined (non-greedy for Expected Sarsa, greedy for Q-learning), which in turn changes the definition of $V(S_{t+1})$ as well as the applicability of importance sampling. To our knowledge, Watkins' Q(λ) is the only existing Q-learning algorithm that leverages this particular variable-trace formulation to manage off-policy data, leaving its broader potential for fine-grained bias control entirely unexamined.[1]

Crucially, this framework reveals that both Watkins' and Peng's Q(λ) are special cases of variable Q(λ) in Eq. (2). Peng's Q(λ) is achieved by simply substituting a constant value of λ. Watkins' Q(λ) is less obvious; it follows because λ is nonzero if and only if an action is greedy, and therefore $\delta'_t = \delta_t$ *only* in Eq. (1). We present pseudocode for the universal eligibility-trace template in Algorithm 1, offering the specific λ definitions for the methods in Table 1. In line 9, we indicate the specific choice of λ to produce our flagship algorithm Gated Q(λ) that is introduced in the next section, but this line can be modified to implement Watkins' Q(λ), Peng's Q(λ), and any other variable Q(λ) method described by Eq. (2).

Table 1: Comparison of different Q(λ) methods that can be expressed by Algorithm 1.

Method	**Decay Strategy**
Watkins' Q(λ)	$\lambda_t = \begin{cases} \lambda & \text{if } \mathrm{greedy}(S_t, A_t) \\ 0 & \text{otherwise} \end{cases}$
Peng's Q(λ)	$\lambda_t = \lambda$
Gated Q(λ)	$\lambda_t = \begin{cases} \lambda & \text{if } \mathrm{greedy}(S_t, A_t) \\ \lambda\chi & \text{otherwise} \end{cases}$

4 Gated Q-learning

Variable Q(λ) offers a large, untapped space of potential multistep Q-learning methods. To address the original problem of balancing trace preservation with off-policy bias mitigation, we focus on a specific subset that we name *Gated Q-learning*.

Gated Q-learning leverages the variable nature of λ_t to target and attenuate credit along *exploratory* trajectories. Unlike Watkins' Q(λ), it does this in a smooth and forgiving way; a trajectory earns only partial credit for each non-greedy action. It turns out that this strategy achieves a pure interpolation between Watkins' and Peng's Q(λ), yet without using any importance sampling. We call this method *Gated Q(λ)* and present it as the main algorithmic contribution of our paper (see Section 4.1), but briefly discuss the special case of an n-step variant as well (see Section 4.2).

[1] Thus, to use this formula with any other choice of $\lambda(s, a)$, we must accept some bias and sacrifice convergence to q_*.

Algorithm 1: Gated Q(λ)

Initialize $Q(s, a)$ arbitrarily and $Z(s, a) \leftarrow 0$ for all (s, a)

for $t = 0, 1, \ldots$ **do**

  Compute greedy action $A_t^* \coloneqq \arg\max_{a \in \mathcal{A}} Q(S_t, a)$

  Take action A_t according to policy in state S_t `// Reuse` A_t^* `if needed`

  Observe reward R_{t+1} and next state S_{t+1}

  $G_t^{(1)} \coloneqq \begin{cases} R_{t+1} & \text{if terminal}(S_{t+1}) \\ R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a') & \text{otherwise} \end{cases}$ `// 1-step return`

  $\delta'_t \coloneqq G_t^{(1)} - Q(S_t, A_t)$ `// QL error`

  $\delta_t \coloneqq G_t^{(1)} - Q(S_t, A_t^*)$ `// TD error`

  $\lambda_t \coloneqq \begin{cases} \lambda & \text{if } A_t = A_t^* \\ \lambda\chi & \text{otherwise} \end{cases}$ `// Change to implement other algorithms`

  foreach (s, a) **do**

    $Z(s, a) \leftarrow \gamma\lambda_t Z(s, a)$ `// Decay trace`

    $Q(s, a) \leftarrow Q(s, a) + \alpha\delta_t Z(s, a)$ `// Apply TD error to past`

  end

  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta'_t$ `// Apply QL error to present`

  if terminal(S_{t+1}) **then**

    $Z(s, a) \leftarrow 0$ for all (s, a) `// Reset traces`

    Reset environment state S_{t+1} `// Next episode`

  else

    $Z(S_t, A_t) \leftarrow Z(S_t, A_t) + 1$ `// Increment trace`

  end

end

4.1 Gated Q(λ)

We can conceptualize credit assignment in Q(λ) methods as productive errors flowing backward in time to update previous value estimates, like water through a series of pipe segments. In any segment, the value of λ determines the instantaneous flow rate: what proportion of the water passes through and affects downstream values. A maximum value of $\lambda = 1$ allows unconstrained flow, flooding the value estimates with noise (high variance). A minimum value of $\lambda = 0$ shuts off the flow entirely, starving the pipeline of precious water (high bias). Ideally, these two concerns would be balanced.

Continuing with this metaphor, non-greedy actions "pollute" the water from a Q-learning perspective; exploratory behavior taints the value estimation with bias because such actions do not match the greedy target policy. However, contaminated water is preferable to no water—the intuitive reason why Peng's Q(λ) outperforms Watkins' Q(λ), which discards the entire stream at the first sign of pollution. It is better to selectively regulate the intake of contaminated water to maintain a reasonable flow rate while balancing cleanliness. This is the core motivation behind Gated Q(λ).

Formally, we introduce a hyperparameter $\chi \in [0, 1]$, which we refer to as the "gate." When an exploratory action is taken, we attenuate the eligibility of the TD error by an additional factor of χ:

$$\lambda_t = \begin{cases} \lambda & \text{if greedy}(S_t, A_t), \\ \lambda\chi & \text{otherwise.} \end{cases}$$

This definition makes it clear that Gated Q(λ) achieves a pure mathematical interpolation between the strict, safe updates of Watkins' Q(λ) ($\chi = 0$) and the fast, uncorrected updates of Peng's Q(λ)

($\chi = 1$). This targeting mechanism is conceptually analogous to the gating architectures found in LSTM networks (Hochreiter & Schmidhuber, 1997), GRUs (Cho et al., 2014; Chung et al., 2014), and gated attention (Xu et al., 2015; Dhingra et al., 2017). Just as these architectures smoothly regulate information flow to protect a network's internal memory state, our mechanism smoothly regulates error flow to protect the action-value estimates. However, *unlike* these architectures, the gate here is not a learnable parameter and remains independent of the agent's function approximator.

To understand the theoretical implication of this mechanism, let $k_{t:t+i} \in \{0, \dots, i\}$ denote the total number of non-greedy actions taken from time step $t+1$ through $t+i$. The forward-view return of Gated Q(λ) can be expressed as:

$$G_t^{\lambda\,(\text{Gated})} = Q(S_t, A_t) + \delta'_t + \sum_{i=1}^{\infty} (\gamma\lambda)^i \chi^{k_{t:t+i}} \delta_{t+i}\,. \tag{4}$$

This perspective clearly illustrates that the multistep error decays unconditionally by $(\gamma\lambda)^i$ as in Peng's method, but is strictly attenuated by an additional factor of χ for every non-greedy choice made along the trajectory. While this forward view provides theoretical clarity regarding the geometric accumulation of the gate, we note that practical implementations will rely on either the equivalent recursive formula in Eq. (3) or standard backward-view eligibility traces as detailed in Algorithm 1.

4.2 n-step Gated Q-learning

As an aside, we note that we can derive an n-step return target that applies the same gating mechanism, and thus our idea is not specific to λ-returns. We achieve this by truncating the Gated Q(λ) target in Eq. (4) to just n steps and then setting $\lambda = 1$. This yields the following return estimate:

$$G_t^{(n)} \coloneqq Q(S_t, A_t) + \delta'_t + \sum_{i=1}^{n-1} \gamma^i \chi^{k_{t:t+i}} \delta_{t+i}\,.$$

Although this n-step estimator is very interesting, we do not pursue it further in this work. We imagine, though, that it could be beneficial to large-buffer deep RL algorithms in the DQN family (Mnih et al., 2015; Hessel et al., 2018), which commonly use biased, uncorrected n-step returns to improve sample efficiency. A slight drawback to our n-step estimate is that its computational cost scales linearly with n due to the individual TD errors. However, with massive hardware parallelization, the cost may not be noticeable for typical values of n.

5 Hyperparameter Study

Before we formally analyze and evaluate Gated Q(λ) in Section 6, we conduct a focused hyperparameter sweep to gain an intuition for how the gating mechanism impacts credit assignment. Code to reproduce this experiment is available online.[2]

For our test environment, we adapt the 19-state random walk from Sutton & Barto (2018, Sec. 12.1). This environment has 19 linearly connected states and two actions to move left or right. The agent starts each episode in the central state. The agent's behavior policy chooses either action with equal probability. Reaching the extreme ends of the walk yields a -1 or $+1$ reward (left and right, respectively) and terminates the episode. The simple, linear topology of this environment is ideal for isolating and measuring credit assignment.

Because the traditional random-walk experiment is set up for on-policy prediction, we must modify it for off-policy control. We first apply a slight discount factor of $\gamma = 0.99$, to incentivize the agent to earn rewards expediently. We then calculate the optimal action-value function, q_*. The key difference in our setup is that we train the agents for a fixed number of steps (500) instead of

[2] https://github.com/brett-daley/gated-q-learning

episodes, to capture the initial learning speed of the agent. Each agent's value function is initialized with negligible Gaussian noise ($\sigma = 10^{-9}$) to break ties at the start. We record the root-mean-square (RMS) error between Q and q_* on every time step. Note that if we trained for *too* long, the results would be biased in favor of Watkins' Q(λ) because it has no asymptotic bias. This would fail to capture the phenomenon we really want to study: the ability for trace preservation across off-policy trajectories to accelerate learning in spite of the bias. We thus want to examine the early phase of training where trace preservation can be a key performance differentiator.

We evaluate Gated Q(λ) with eligibility traces, described in Algorithm 1, by sweeping over α, λ, and χ. We divide each axis into 21 uniform values from 0 to 1. This results in 9,261 hyperparameter combinations, each averaged across 300 random seeds, for a total of almost 3 million independent trials. Note that the extremal values of $\chi = 0$ and $\chi = 1$ correspond to Watkins' and Peng's, respectively, so the vast majority of evaluated configurations represent new and previously untested variations of Q(λ).

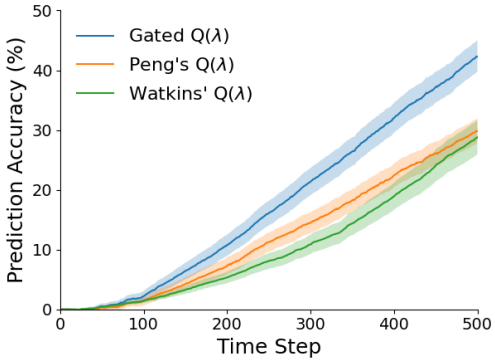

Figure 1: Learning curves for Gated, Peng's, and Watkins' Q(λ) methods with their respective best hyperparameters. Prediction accuracy is defined as the overall RMS error reduction, where 0% represents no progress and 100% represents perfect convergence to q_*. Results are averaged over 300 random seeds each. The shading represents 95% confidence intervals.

Let us consider how the results are theoretically affected by preserving traces. Each time the agent moves right, it receives an optimal experience. Clearly, this experience should be reinforced, and all of the agents weight its influence on past state-action pairs by the same value: λ. The sole difference in the methods lies in the case where the agent moves *left*—a suboptimal exploratory move. If the agent were to move right again afterwards, the methods that do not cut traces will ultimately reinforce the correct action in spite of this exploration. This is the principal mechanism that makes Peng's Q(λ) learn faster in practice. However, if the agent *does* end up receiving the negative reward on the left, it will reinforce a bad experience that would never be taken by the optimal policy, incurring bias.

We generate learning curves for each hyperparameter configuration. We invert and normalize the RMS errors to instead measure *prediction accuracy*, where 0% represents the initial error (no learning) and 100% represents perfect convergence to q_* (optimal performance). We take the area under the curve (AUC) of each learning curve to produce a single summarizing scalar. The curves for the best-performing hyperparameter selection for each method (with χ fixed at 0 or 1 for Watkins' and Peng's, respectively; see Table 2) are plotted in Figure 1.

Table 2: Hyperparameters used in Figure 1.

Method	α	λ	χ
Watkins' Q(λ)	1.00	0.95	0.00
Gated Q(λ)	0.95	1.00	0.45
Peng's Q(λ)	1.00	0.70	1.00

Collectively, the AUC scalars form a 3-D cube of $21 \times 21 \times 21$ voxels. We identify the globally optimal point at $\alpha = 0.95$, $\lambda = 1.00$, and $\chi = 0.45$. We then cross section the cube through this point along orthogonal planes to generate three 2-D heatmap visualizations in Figure 2.

This experiment reveals three major insights: (i) Gated Q(λ) learns significantly faster than both baselines; (ii) the gating mechanism enables the safe use of higher λ-values; and (iii) performance is remarkably robust across a wide range of intermediate gate values ($\chi \in [0.2, 0.6]$). Clearly, an intermediate χ improves the best performance in this task. This gives merit to the idea that balancing the bias-variance trade-off through a gating mechanism is advantageous.

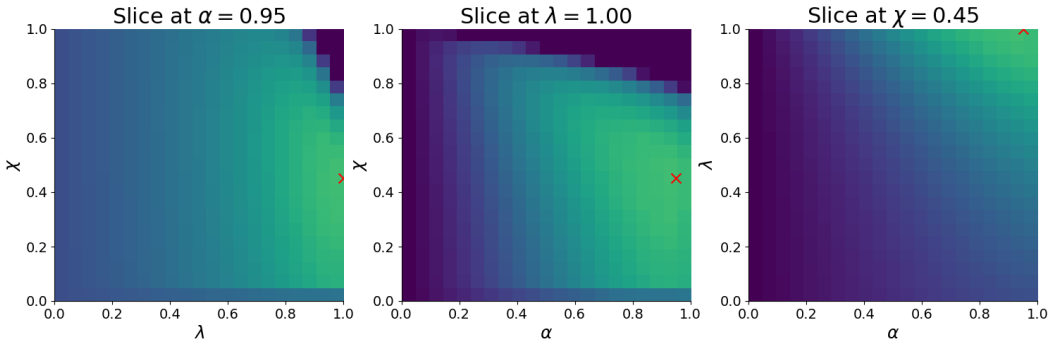

Figure 2: Cross-sectional slices of the 3-D heatmap generated for the random walk. Each slice passes through the coordinate of the best hyperparameters found by the search, indicated by a red X (see Table 2). The colors indicate the area under the curve achieved by each hyperparameter configuration, with warmer colors indicating larger values. Each voxel is averaged over 300 trials.

6 Analysis

In this section, we identify and analyze a general value-function operator underpinning the variable Q(λ) method introduced in Section 3. Our analysis here is not intended to capture the stochastic behavior of the online eligibility-trace method presented in Algorithm 1. Instead, we primarily wish to understand the properties of Gated Q(λ)'s forward-view return target.[3] This is especially relevant for deep RL, where value estimates are typically produced by a frozen target network and backward-view eligibility traces are generally not used.[4] Operator theory provides exactly the right tool to understand how the gating mechanism addresses the trade-off between expected convergence speed and off-policy bias.

6.1 Variable Q(λ) Operator

We start by deriving $T^\Lambda \colon \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, a general operator capable of expressing the variable Q(λ) methods introduced in Section 3, including Watkins' Q(λ), Peng's Q(λ), and our Gated Q(λ). Here, vectors represent value functions (e.g., $\boldsymbol{q} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ represents Q), where each element corresponds to the value estimate for a particular state-action pair. The operator acts on the state-action value estimates in the update $\boldsymbol{q} \leftarrow T^\Lambda \boldsymbol{q}$, thus producing the expected value of the targets $\tilde{G}_t^\lambda$ from Eq. (2) for each state-action pair. Similarly, if we were to set α to be very small and hold Q fixed while executing Algorithm 1 and averaging updates on the side, then the net update would be approximately equal to $T^\Lambda \boldsymbol{q} - \boldsymbol{q}$. This is the classic forward-backward equivalence of eligibility traces (Sutton & Barto, 1998, Sec. 7.4), allowing us to analyze the expected behavior of the return target and the eligibility traces purely through the lens of T^Λ.

To construct T^Λ, we first define a few fundamental operators based on the bipartite MDP decomposition from Daley (2025, Ch. 2). Let $P \colon \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ be the transition dynamics operator. We also define a policy operator $E_\pi \colon \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}|}$ which computes the expected state values under any policy π. Let E_* denote the special case of a *greedy* policy π_{greedy} with respect to Q. Finally, to simplify the analysis, we introduce an expansion operator $J \colon \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, which broadcasts a state's value to each of its action values. Note that the reward function is itself treated as a value function, $\boldsymbol{r} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, to which dimensionally compatible operators can be applied.

[3] Standard results in stochastic approximation (e.g., Bertsekas & Tsitsiklis, 1996, Ch. 4–5) suggest that a stochastic algorithm with a corresponding contractive operator converges to the same fixed point, given appropriate step-size schedules and sufficient exploration. Formally establishing this here is left for future work.

[4] We refer the reader to Daley (2025, Ch. 1, 7) for a detailed account. Forward-view returns are more compatible with mini-batch sampling, which draws transitions out of temporal order. Although some recent algorithms have revisited backward-view eligibility traces for deep RL, they currently tend to be less stable and sample-efficient than replay methods.

With these definitions, the Bellman operator for action values is expressed as

$$T\boldsymbol{q} \coloneqq \boldsymbol{r} + \gamma \boldsymbol{P} E_* \boldsymbol{q} \,. \tag{5}$$

Note that we bold both value functions and *linear* operators whenever they appear as such algebraic quantities. Next, we define the trace-decay matrix $\boldsymbol{\Lambda}$. Because the trace-decay parameter $\lambda(s,a)$ can vary by state-action pair, $\boldsymbol{\Lambda}$ is a diagonal matrix of size $|\mathcal{S}\times\mathcal{A}| \times |\mathcal{S}\times\mathcal{A}|$. The elements are defined by

$$\Lambda_{i,j} \coloneqq \begin{cases} \lambda(s_i,a_i) & \text{if } i = j \,, \\ 0 & \text{if } i \neq j \,, \end{cases} \quad \forall\, i,j \in \{1,\ldots,|\mathcal{S}\times\mathcal{A}|\} \,.$$

It is crucial to note that because methods like Gated Q(λ) define $\lambda(s,a)$ as a function of the current value estimates, $\boldsymbol{\Lambda}$ is an implicit function of Q in general. This theoretically makes it a nonlinear operator. However, because we analyze only the pure policy-evaluation setting in our work, we can safely assume that Q is a fixed quantity and therefore treat $\boldsymbol{\Lambda}$ as a linear operator. This is a standard and necessary assumption for policy evaluation, which makes the analysis tractable while perfectly mirroring the stale-target setting common in modern RL architectures.

Proposition 1. *The operator T^Λ for variable Q(λ) has the closed-form definition*

$$T^\Lambda \boldsymbol{q} \coloneqq (\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda})^{-1}(T\boldsymbol{q} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*\boldsymbol{q}) \,. \tag{6}$$

Proof. See Section A.1. The result follows by converting the expectation of the recursive target in Eq. (3) to operator form, and then manipulating it algebraically to isolate T^Λ. □

Eq. (6) reveals that the T^Λ operator takes the generic form $\boldsymbol{Z}^{-1}(\boldsymbol{y} + \boldsymbol{X}\boldsymbol{q})$. Here, the matrix $\boldsymbol{Z}^{-1} = (\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda})^{-1}$ represents the expected eligibility trace for every state-action pair under the behavior policy b and the chosen decay scheme $\boldsymbol{\Lambda}$. The inner vector term $(\boldsymbol{y} + \boldsymbol{X}\boldsymbol{q}) = \boldsymbol{r} + \gamma \boldsymbol{P}E_*\boldsymbol{q} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*\boldsymbol{q}$ fills in the complementary gaps of the decayed trace with bootstrapped state-value estimates derived from $\max_{a\in\mathcal{A}} Q(s,a)$.

6.2 Contraction Rate

To quantify the expected error reduction achieved by a single application of T^Λ, our next step is to prove that it is a contraction mapping. Let $\|\cdot\|_\infty$ denote the maximum norm of a vector or matrix (i.e., the maximum row sum of the absolute values). We formally define a contraction mapping with respect to this norm below.

Definition 1. *A value-function operator $H\colon \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ is a contraction mapping if and only if there exists a constant $\beta \in [0,1)$ such that, $\forall\, \boldsymbol{q}_1, \boldsymbol{q}_2 \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$,*

$$\|H\boldsymbol{q}_1 - H\boldsymbol{q}_2\|_\infty \leq \beta \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|_\infty \,.$$

We seek to find a *contraction modulus* for T^Λ: a constant β that satisfies Definition 1. To facilitate this, let $\mathbf{1}_\mathcal{S} \in \mathbb{R}^{|\mathcal{S}|}$ and $\mathbf{1}_{\mathcal{S}\times\mathcal{A}} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ denote the all-ones vectors.

Theorem 1. *Let $\boldsymbol{c} \coloneqq \boldsymbol{E}_b\boldsymbol{\Lambda}\mathbf{1}_{\mathcal{S}\times\mathcal{A}}$ and $c^- \coloneqq \min_{s\in\mathcal{S}} c(s)$. The operator T^Λ is a contraction mapping with modulus*

$$\beta = \frac{\gamma(1-c^-)}{1-\gamma c^-} \,.$$

Proof. See Section A.2. □

The value c^- represents the smallest state-conditional expected trace-decay value across all states. Because β is monotonically decreasing with c^-, this formalizes the intuition that the trace preservation inherent to methods like Peng's Q(λ) is indeed beneficial to convergence speed in expectation.

When we apply a constant value of λ to all state-action pairs, the formula gracefully collapses back to $\beta = \gamma(1-\lambda) \,/\, (1-\gamma\lambda)$, which exactly matches the known contraction modulus for standard TD(λ) and Peng's Q(λ) (see, e.g., Daley et al., 2024, Proof of Prop. 3.11).

For Gated Q(λ), the algorithm's trace decay depends on whether a greedy action is taken: λ if so and $\lambda\chi$ if not. If the behavior policy has probability $p_g(s)$ of acting greedily in state s, then the expected trace value is $c(s) = p_g(s) \cdot \lambda + (1 - p_g(s)) \cdot \lambda\chi$. This means c^- is determined by the *least-greedy* state: the state where the agent currently explores the most.[5] This implies that a gate of $\chi < 1$ actually *reduces* expected convergence speed, making the method more like Watkins' Q(λ). However, this can be compensated for by increasing λ beyond what would be considered safe for Peng's Q(λ), as we show with the fixed-point analysis in the next subsection.

6.3 Fixed Point and Off-Policy Bias

We have now established that T^Λ is a contraction mapping. By the Banach fixed-point theorem (Banach, 1922), T^Λ admits a *unique* fixed point, to which repeated application of the operator converges: i.e., $\lim_{i\to\infty}(T^\Lambda)^i \boldsymbol{q}$ exists and is the same for every $\boldsymbol{q} \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$. We must now identify the nature of this fixed point.

We once again leverage the recursive formula to unpack the components of the operator. By doing so, we find that the algorithm implicitly evaluates a composite policy that interpolates between the behavior and (greedy) target policies.

Theorem 2. *Let π_{mix} be the mixture policy defined by the following policy operator:*

$$\boldsymbol{E}_{\pi_{mix}} \coloneqq \boldsymbol{E}_b\boldsymbol{\Lambda} + (\boldsymbol{I}_\mathcal{S} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J})E_* \,.$$

The unique fixed point of operator T^Λ is exactly the action-value function of π_{mix}:

$$\boldsymbol{q}_{\pi_{mix}} \coloneqq (\boldsymbol{I} - \gamma\boldsymbol{P}\boldsymbol{E}_{\pi_{mix}})^{-1}\boldsymbol{r} \,.$$

Proof. See Section A.3. □

For every choice of the $\lambda(s,a)$ function, variable Q(λ) converges to the value function corresponding to a policy which mixes the behavior and greedy policies. For clarity, we show the point-wise definition of π_{mix} which is derived directly from the operators' definitions:

$$\pi_{\text{mix}}(a|s) \coloneqq b(a|s)\lambda(s,a) + (1 - c(s))\pi_{\text{greedy}}(a|s) \,.$$

Kozuno et al. (2021, Th. 2) proved that Peng's Q(λ) converges to the value function of a mixture policy formed by the convex mixture $\lambda b(a|s) + (1-\lambda)\pi_{\text{greedy}}(a|s)$, which matches our result above when $\lambda(s,a) = \lambda$ for all (s,a). Thus, our Theorem 2 is a pure generalization.

This result is fascinating in that it formalizes the precise way in which variable Q(λ) methods trade off trace preservation with off-policy bias. In particular, on-policy values seep into the return estimation whenever we simultaneously have $\lambda(s,a) > 0$ and $b(a|s) > 0$ for a non-greedy pair (s,a), indicating that any amount of preserved trace during exploration contributes bias. This is because such a pair has $\pi_{\text{mix}}(a|s) = b(a|s)\lambda(s,a) > 0$, whereas $\pi_{\text{greedy}}(a|s) = 0$, altering the fixed point because the mixture policy is no longer greedy. This affirms that Watkins' Q(λ) is the only way to truly eliminate off-policy bias by setting $\lambda(s,a) = 0$ whenever (s,a) is exploratory. Nevertheless, biased Q(λ) methods can still converge to a *nearby* value function, and converge to it much faster than Watkins' Q(λ), making them practically useful. Adapting λ based on the state-action pair provides the exact capability to adjust where and how much off-policy bias is added to the return estimation. Gated Q(λ) offers this capability via the gating mechanism $\chi \in [0,1]$ which only targets exploratory actions, enabling users to fine-tune the degree of off-policy bias without negatively impacting unbiased greedy actions.

[5] If the agent is executing an ϵ-greedy policy, then exploration is state-wise uniform and we simply substitute the probability of taking a greedy action: $p_g(s) = 1 - \epsilon + \epsilon \,/\, |\mathcal{A}|$.

7 Conclusion

We introduced Gated Q-learning, a novel algorithmic framework that resolves the 30-year tension between the aggressive trace-cutting of Watkins' Q(λ) and the uncorrected bias of Peng's Q(λ). By formalizing partial off-policy corrections through a state-action-dependent gating mechanism, we provide a new continuum of algorithms that finely balances long-term credit assignment with off-policy bias. Our theoretical analysis proves that this strategy always guarantees a contraction mapping and convergence, with smoothly bounded (but generally nonzero) fixed-point bias. Key limitations include the newly added gate hyperparameter χ, which in theory must be tuned in conjunction with λ. However, the wide plateau near $\chi \approx 0.5$ in Figure 2 suggests a favorable margin of error when setting this hyperparameter, at least in the tested environment. Another limitation is that we did not make any direct empirical comparisons with importance-sampled estimators like Retrace (Munos et al., 2016), as we focused specifically on greedy Q-learning algorithms. Evaluating the relative effectiveness of these two distinct off-policy corrections remains an important direction for future work. Finally, our analysis did not consider the variance of the return estimates due to the complexity of the problem, though it seems likely that forgoing importance-sampling ratios is a significant boon to variance reduction.

Although not evaluated in the deep RL setting, Gated Q-learning can seamlessly integrate into existing agents, including trajectory-replay methods that use λ-returns such as A3C (Mnih et al., 2016), DQN(λ) (Daley & Amato, 2019), and PQN (Gallici et al., 2025), as well as minibatch-replay methods in the DQN family (Mnih et al., 2015; Hessel et al., 2018) that use n-step returns (recall Section 4.2). We see significant potential to improve performance in these algorithms, as off-policy bias tends to be extreme under experience replay. Here, Gated Q-learning can serve as a simpler and more efficient alternative to importance sampling, though we make no claim of superior performance yet.

References

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

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

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2014.

Brett Daley. *Multistep Credit Assignment in Deep Reinforcement Learning*. PhD thesis, University of Alberta, 2025.

Brett Daley and Christopher Amato. Reconciling λ-returns with experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

Brett Daley, Martha White, and Marlos C. Machado. Averaging n-step returns reduces variance in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Esraa Elelimy, Brett Daley, Andrew Patterson, Marlos C. Machado, Adam White, and Martha White. Deep reinforcement learning with gradient eligibility traces. *Reinforcement Learning Journal*, 2025.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Anna Harutyunyan, Marc G. Bellemare, Tom Stepleton, and Rémi Munos. Q(λ) with off-policy corrections. In *International Conference on Algorithmic Learning Theory (ALT)*, 2016.

J. Fernando Hernandez-Garcia and Richard S. Sutton. Understanding multi-step deep reinforcement learning: A systematic study of the DQN target. In *NeurIPS Deep Reinforcement Learning Workshop*, 2018.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Herman Kahn and Andy W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Tadashi Kozuno, Yunhao Tang, Mark Rowland, Rémi Munos, Steven Kapturowski, Will Dabney, Michal Valko, and David Abel. Revisiting Peng's Q(λ) for modern reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

Seyed Sajad Mousavi, Michael Schukat, Enda Howley, and Patrick Mannion. Applying Q(λ)-learning in deep reinforcement learning to play Atari games. In *AAMAS Adaptive Learning Agents Workshop*, 2017.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

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

Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning (ICML)*, 2000.

Matthew Schlegel, Wesley Chung, Daniel Graves, Jian Qian, and Martha White. Importance resampling for off-policy prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1st edition, 1998.

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

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

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, 2015.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Proofs

This section contains the full mathematical proofs omitted from the main paper for exposition ease.

A.1 Proof of Proposition 1

Proposition 1. *The operator T^Λ for variable Q(λ) has the closed-form definition*

$$T^\Lambda \boldsymbol{q} \coloneqq (\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda})^{-1}(T\boldsymbol{q} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*\boldsymbol{q})\,. \tag{6}$$

Proof. We first rewrite the recursive λ-return formula from Eq. (3) as

$$\tilde{G}_t^\lambda = R_{t+1} + \gamma V(S_{t+1}) + \gamma\lambda_{t+1}\Big(\tilde{G}_{t+1}^\lambda - V(S_{t+1})\Big)\,.$$

To convert this to an operator equation, we must convert the expected values of these quantities to their matrix or vector equivalents:

$$T^\Lambda \boldsymbol{q} = T\boldsymbol{q} + \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}(T^\Lambda \boldsymbol{q} - \boldsymbol{J}E_*\boldsymbol{q})\,. \tag{7}$$

This is because the first term is the Bellman operator from Eq. (5), the time-step shift and decay are handled by multiplying $\gamma\boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}$, and the inner term references the operator itself again, minus the state-value estimate (which must be expanded by $\boldsymbol{J}$ to match the $\mathcal{S}\times\mathcal{A}$ dimension).

We conclude by solving Eq. (7) algebraically for T^Λ. (Here and throughout, the unsubscripted identity matrix $\boldsymbol{I}$ has dimension $|\mathcal{S}\times\mathcal{A}| \times |\mathcal{S}\times\mathcal{A}|$.) Rearranging gives

$$(\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda})T^\Lambda \boldsymbol{q} = T\boldsymbol{q} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*\boldsymbol{q}\,, \tag{8}$$

and left-multiplying both sides by $(\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda})^{-1}$ yields Eq. (6), completing the derivation. □

A.2 Proof of Theorem 1

Theorem 1. *Let $\boldsymbol{c} \coloneqq \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{1}_{\mathcal{S}\times\mathcal{A}}$ and $c^- \coloneqq \min_{s\in\mathcal{S}} c(s)$. The operator T^Λ is a contraction mapping with modulus*

$$\beta = \frac{\gamma(1-c^-)}{1-\gamma c^-}\,.$$

Proof. Let $\boldsymbol{q}_1, \boldsymbol{q}_2 \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$. To simplify notation, we define the differences $\Delta T^\Lambda \boldsymbol{q} \coloneqq T^\Lambda \boldsymbol{q}_1 - T^\Lambda \boldsymbol{q}_2$, $\Delta\boldsymbol{q} \coloneqq \boldsymbol{q}_1 - \boldsymbol{q}_2$, and $\Delta\boldsymbol{v} \coloneqq E_*\boldsymbol{q}_1 - E_*\boldsymbol{q}_2$. Note that $\Delta T\boldsymbol{q} = \gamma\boldsymbol{P}\Delta\boldsymbol{v}$. From substitution into the recursive definition of T^Λ in Eq. (7) and by linearity, we have

$$\begin{aligned}\Delta T^\Lambda \boldsymbol{q} &= \gamma\boldsymbol{P}\Delta\boldsymbol{v} + \gamma\boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}(\Delta T^\Lambda\boldsymbol{q} - \boldsymbol{J}\Delta\boldsymbol{v})\\ &= \gamma\boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\Delta T^\Lambda\boldsymbol{q} + \gamma\boldsymbol{P}(\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J})\Delta\boldsymbol{v}\,.\end{aligned}$$

To form an upper bound, we apply the *element-wise* absolute value to each vector and invoke the triangle inequality. We then use the fact that $|\boldsymbol{x}| \le \|\boldsymbol{x}\|_\infty \boldsymbol{1}$ holds element-wise. Additionally, observe that $(\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J})\boldsymbol{1}_{\mathcal{S}} = \boldsymbol{1}_{\mathcal{S}} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{1}_{\mathcal{S}\times\mathcal{A}} = \boldsymbol{1}_{\mathcal{S}} - \boldsymbol{c}$. Most of the terms have nonnegative

components and can be safely pulled out of the absolute value:

$$\begin{aligned}
\left|\Delta T^{\Lambda}\boldsymbol{q}\right| &\leq \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\left|\Delta T^{\Lambda}\boldsymbol{q}\right| + \gamma \boldsymbol{P}(\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J})|\Delta \boldsymbol{v}| \\
&\leq \gamma \boldsymbol{P}\Big(\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \boldsymbol{E}_b\boldsymbol{\Lambda}\mathbf{1}_{\mathcal{S}\times\mathcal{A}} + \|\Delta \boldsymbol{v}\|_\infty (\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J})\mathbf{1}_{\mathcal{S}}\Big) \\
&= \gamma \boldsymbol{P}\Big(\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \boldsymbol{c} + \|\Delta \boldsymbol{v}\|_\infty (\mathbf{1}_{\mathcal{S}} - \boldsymbol{c})\Big) \\
&\leq \gamma \boldsymbol{P}\Big(\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \boldsymbol{c} + \|\Delta \boldsymbol{q}\|_\infty (\mathbf{1}_{\mathcal{S}} - \boldsymbol{c})\Big)\,,
\end{aligned}$$

where the last step follows because the $\max$ operator is non-expansive, hence $\|\Delta \boldsymbol{v}\|_\infty \leq \|\Delta \boldsymbol{q}\|_\infty$. Note that $\|\boldsymbol{P}\|_\infty = 1$ because $\boldsymbol{P}$ is a stochastic matrix. Taking the norm of both sides yields

$$\begin{aligned}
\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty &\leq \gamma \Big\| \left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \boldsymbol{c} + \|\Delta \boldsymbol{q}\|_\infty (\mathbf{1}_{\mathcal{S}} - \boldsymbol{c})\Big\|_\infty \\
&= \gamma \max_s \Big(c(s)\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty + (1 - c(s))\|\Delta \boldsymbol{q}\|_\infty \Big)\,. \qquad (9)
\end{aligned}$$

This inequality presents a convex combination of $\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty$ and $\|\Delta \boldsymbol{q}\|_\infty$. If we assume for a moment that $\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty > \|\Delta \boldsymbol{q}\|_\infty$, the convex combination would be strictly less than $\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty$. This would imply $\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty < \gamma\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty$, which is impossible since $\gamma \leq 1$. Therefore, it must be universally true that $\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \leq \|\Delta \boldsymbol{q}\|_\infty$.

Because $\|\Delta \boldsymbol{q}\|_\infty$ is the larger of the two quantities, the convex combination is maximized by placing as much weight as possible on it. This means maximizing $1 - c(s)$ in Eq. (9), which is achieved by setting $c(s) = c^-$. Substituting c^- into the bound yields

$$\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \leq \gamma c^- \left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty + \gamma(1 - c^-)\|\Delta \boldsymbol{q}\|_\infty\,,$$

which rearranges to

$$\left\|\Delta T^{\Lambda}\boldsymbol{q}\right\|_\infty \leq \underbrace{\frac{\gamma(1 - c^-)}{1 - \gamma c^-}}_{\beta} \|\Delta \boldsymbol{q}\|_\infty\,.$$

Because $0 \leq c^- \leq 1$ by definition of $\lambda(s, a)$, the coefficient β is strictly less than 1 whenever $\gamma < 1$, confirming that T^{Λ} is a contraction mapping with the stated modulus. □

A.3 Proof of Theorem 2

Theorem 2. *Let π_{mix} be the mixture policy defined by the following policy operator:*

$$\boldsymbol{E}_{\pi_{mix}} \coloneqq \boldsymbol{E}_b\boldsymbol{\Lambda} + (\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J})E_*\,.$$

The unique fixed point of operator T^{Λ} is exactly the action-value function of π_{mix}:

$$\boldsymbol{q}_{\pi_{mix}} \coloneqq (\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_{\pi_{mix}})^{-1}\boldsymbol{r}\,.$$

Proof. At the fixed point, we must have $T^{\Lambda}\boldsymbol{q} = \boldsymbol{q}$. Starting from Eq. (8), expanding $T\boldsymbol{q}$ using Eq. (5), and then substituting $\boldsymbol{q}$ for $T^{\Lambda}\boldsymbol{q}$, we obtain

$$\begin{aligned}
(\boldsymbol{I} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda})\boldsymbol{q} &= \boldsymbol{r} + \gamma \boldsymbol{P}E_*\boldsymbol{q} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*\boldsymbol{q} \\
\boldsymbol{q} &= \boldsymbol{r} + \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{q} + \gamma \boldsymbol{P}E_*\boldsymbol{q} - \gamma \boldsymbol{P}\boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*\boldsymbol{q} \\
&= \boldsymbol{r} + \gamma \boldsymbol{P}(\boldsymbol{E}_b\boldsymbol{\Lambda} + E_* - \boldsymbol{E}_b\boldsymbol{\Lambda}\boldsymbol{J}E_*)\boldsymbol{q} \qquad (10) \\
&= \boldsymbol{r} + \gamma \boldsymbol{P}\boldsymbol{E}_{\pi_{\text{mix}}}\boldsymbol{q}\,, \qquad (11)
\end{aligned}$$

where the last step substitutes the policy operator defined in the theorem. Note that factoring out $\boldsymbol{q}$ in Eq. (10) is justified because, as established earlier, Q is assumed to be fixed for the operator evaluation. This allows us to locally treat the greedy expectation E_* as a linear operator.

Eq. (11) reveals that the fixed point takes the standard Bellman form for the previously defined mixture policy. To complete the proof, we must verify that $\boldsymbol{E}_{\pi_{\text{mix}}}$ is a valid stochastic matrix, thereby representing a realizable policy. This means $\boldsymbol{E}_{\pi_{\text{mix}}}$ must comprise exclusively nonnegative elements and its rows must sum to one.

We first establish nonnegativity. By definition, the policy operators $\boldsymbol{E}_b$ and E_*, as well as the trace matrix $\boldsymbol{\Lambda}$, contain exclusively nonnegative elements. The only term that could theoretically introduce negative values is the subtraction in $\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}$. However, the operator $\boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}$ effectively maps a state to itself with the conditionally expected trace decay $c(s) = \sum_a b(a|s)\lambda(s,a)$. Because $\lambda(s,a) \in [0,1]$ and the behavior policy b is a valid probability distribution, it is guaranteed that $c(s) \in [0,1]$ for all $s \in \mathcal{S}$. Consequently, $\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}$ is a diagonal matrix whose diagonal entries are $1 - c(s) \geq 0$.

Finally, we verify the row sums by applying the operator to the all-ones vector. Recall from the proof of Theorem 1 that $\boldsymbol{c} = \boldsymbol{E}_b \boldsymbol{\Lambda} \mathbf{1}_{\mathcal{S}\times\mathcal{A}}$ and $\mathbf{1}_{\mathcal{S}} - \boldsymbol{c} = (\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}) \mathbf{1}_{\mathcal{S}}$. Therefore,

$$
\begin{aligned}
\boldsymbol{E}_{\pi_{\text{mix}}} \mathbf{1}_{\mathcal{S}\times\mathcal{A}} &= \boldsymbol{E}_b \boldsymbol{\Lambda} \mathbf{1}_{\mathcal{S}\times\mathcal{A}} + (\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}) E_* \mathbf{1}_{\mathcal{S}\times\mathcal{A}} \\
&= \boldsymbol{c} + (\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}) E_* \mathbf{1}_{\mathcal{S}\times\mathcal{A}} \\
&= \boldsymbol{c} + (\boldsymbol{I}_{\mathcal{S}} - \boldsymbol{E}_b \boldsymbol{\Lambda} \boldsymbol{J}) \mathbf{1}_{\mathcal{S}} \\
&= \boldsymbol{c} + \mathbf{1}_{\mathcal{S}} - \boldsymbol{c} \\
&= \mathbf{1}_{\mathcal{S}} \,.
\end{aligned}
$$

Since applying $\boldsymbol{E}_{\pi_{\text{mix}}}$ to the all-ones vector perfectly recovers the all-ones vector (of the appropriate dimensions), the row sums of the implied transition matrix all equal 1. Coupled with the fact that it has nonnegative components, $\boldsymbol{E}_{\pi_{\text{mix}}}$ represents a valid policy and the proof is complete. □