The University of Texas at Austin

# Horizon-Scalable Q-Learning

**Haoran Xu  Pragya Srivastava  Xinjie Liu  Siddhant Agarwal  Amy Zhang**

The University of Texas at Austin

haoran.xu@utexas.edu

**Website** • **GitHub**

## Abstract

Learning accurate off-policy state-action value functions (Q-functions) from offline datasets becomes harder as the task horizon grows. One-step temporal-difference (TD) learning bootstraps after every transition, so small per-step errors accumulate along the horizon, and early states can be badly misvalued even when the TD loss is small. Multi-step TD shortens this chain by following the dataset trajectory for several steps; however, since the behavior policy is often suboptimal, every followed action adds a bias, and no fixed number of steps suits a whole trajectory. In this paper, we propose Horizon-Scalable Q-Learning (HQL), a simple method that enables accurate long-horizon value learning by automatically balancing between one-step and multi-step returns. HQL extends a target through a dataset action only when the critic values it at least as highly as the best-of-N actions sampled from the BC policy, and otherwise falls back to a one-step target. We show that this rule is a principled off-policy correction that needs no action densities, which makes HQL compatible with expressive flow and diffusion policies. Experiments show that HQL avoids the failures of both one-step and multi-step TD on a combination-lock task across horizons, and outperforms prior horizon-reduction methods on long-horizon OGBench and OCBench tasks.

## 1 Introduction

Off-policy reinforcement learning (RL) learns the state-action value function (Q-function) of a target policy from an offline dataset collected by a different behavior policy, which enables RL to reuse logged experience in real-world domains such as robotics (Levine et al., 2016; Kalashnikov et al., 2018) and industrial control (Zhan et al., 2022). However, off-policy value learning becomes harder as the *effective horizon* grows (Laidlaw et al., 2023), i.e., the number of transitions across which the critic must carry value information before reaching a state whose value it already knows. Most off-policy actor-critic methods learn the critic with one-step temporal-difference (TD) backups (Fujimoto et al., 2019; Kumar et al., 2020; Kostrikov et al., 2022; Xu et al., 2023; Park et al., 2025c). A one-step target bootstraps after a single logged transition, which makes it robust to off-policy data but moves value back by only one transition per update. We show that the value error at a state is exactly the sum of the one-step Bellman residuals along the rest of the trajectory, so a critic can reach a small TD loss and still misvalue the start states by an amount that grows with the horizon.

Multi-step returns (Watkins, 1989; Sutton and Barto, 2018) or Q-chunking (Li et al., 2025b; Zhao et al., 2023) shorten this chain by following the dataset trajectory for several steps before bootstrapping. Nevertheless, this imposes a trade-off between fast value propagation (longer backups) and low off-policy bias (shorter backups): we show that each spanned suboptimal action adds a systematic bias that grows toward a limit linear in $H$. Since offline trajectories could be mixed with suboptimal actions, no fixed backup length is optimal for every TD update. Per-decision off-policy corrections such as Retrace (Precup et al., 2000; Munos et al., 2016) make this choice at every transition with importance ratios. However, these ratios require policy densities, which flow or diffusion-based policies (Park et al., 2025c; Mao et al., 2024a; Li and Levine, 2026; Xu et al., 2026) do not expose.

In this paper, we propose Horizon-Scalable Q-Learning (HQL), which shortens the effective horizon of Q-learning without inheriting the bias of suboptimal dataset actions. HQL achieves this by automatically balancing between one-step and multi-step returns at every transition (Figure 1). More specifically, HQL attaches a binary *gate* to every logged action, i.e., every action stored in the dataset: if the target critic values it at least as highly as the best-of-$N$ actions sampled from the BC policy, the gate is open and the

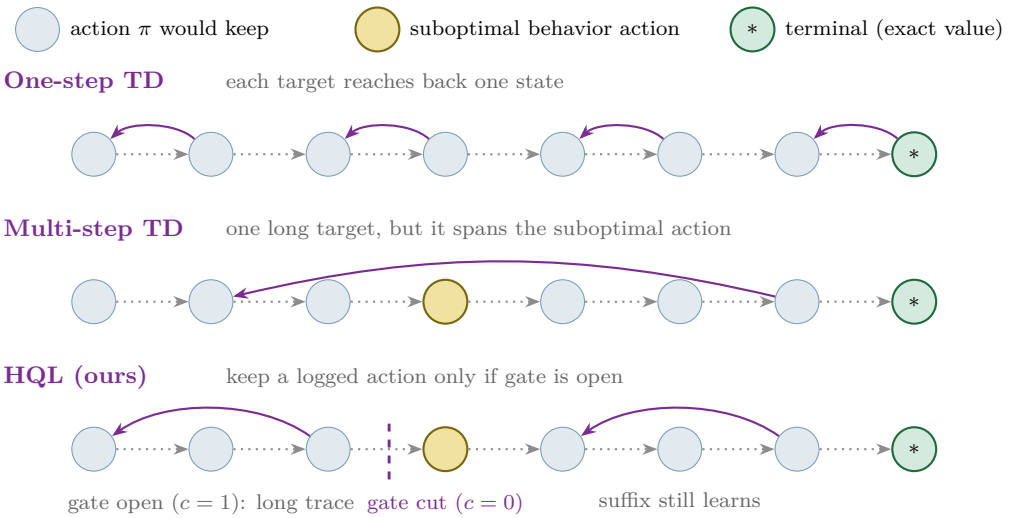



Figure 1: **Horizon-Scalable Q-Learning (HQL).** Purple arcs show the reach of one training target. One-step TD accumulates residuals, and multi-step TD inherits the bias of the suboptimal action it spans (yellow). HQL compares each logged action with the best-of-$N$ policy samples, continuing the trace where it passes ($c$=1) and cutting it where it fails ($c$=0, dashed), so reward travels far without inheriting off-policy bias.

target continues through it for one more step; otherwise, the target falls back to a one-step bootstrap. Our key observation is that this gate is principled without any density: for the optimal policy $\pi \propto \mu \exp(\tau Q)$ that behavior-regularized RL approximates (Peters et al., 2010; Peng et al., 2019; Xu et al., 2025b), the behavior density cancels from the importance ratio, and as $\tau \to \infty$ the largest gate that preserves $Q^\pi$ becomes exactly the comparison of critic values. HQL uses an estimated behavior model only to sample action candidates and never evaluates its density. Since the gate depends only on the ordering of critic values, it is also invariant to the reward scale, and the only hyperparameter in HQL is the number of candidates $N$.

We evaluate HQL on a combination-lock task, whose true $Q$-values are known exactly, and on long-horizon OGBench and OCBench tasks (Park et al., 2025a; Park and Levine, 2026). On the combination lock, HQL keeps the lowest value estimation error across different horizon, whereas one-step TD accumulates error and $n$-step TD fails on corrupted dataset. On OGBench and OCBench, at a matched budget of critic targets, HQL outperforms one-step TD, $n$-step TD, Q-chunking (Li et al., 2025b), and Long-horizon Q-Learning (Abraham et al., 2026), especially on the most challenging long-horizon tasks. Perhaps surprisingly, we find that most of HQL's targets are one or few-step targets: HQL wins by choosing *which* targets to extend, not by extending them far.

## 2 Related Work

**Error propagation in value learning.** Error propagation through bootstrapping has been studied for approximate value and policy iteration (Munos, 2005; Farahmand et al., 2010), TD with function approximation (Tsitsiklis and Roy, 1997), and $n$-step and TD($\lambda$) updates (Kearns and Singh, 2000); its instability with off-policy data is the deadly triad (van Hasselt et al., 2018), and the horizon governs the difficulty of evaluation and learning (Liu et al., 2018; Laidlaw et al., 2023).

**Multi-step off-policy correction.** Multi-step returns and eligibility traces shorten the backup chain (Watkins, 1989; Sutton and Barto, 2018; De Asis et al., 2018), and a small fixed $n$ is standard in deep RL (Hessel et al., 2018; Kapturowski et al., 2019; Hernandez-Garcia and Sutton, 2019). Per-decision importance sampling (Precup et al., 2000) and its clipped variants (Retrace (Munos et al., 2016; Touati et al., 2018), off-policy $Q(\lambda)$ (Harutyunyan et al., 2016), V-trace (Espeholt et al., 2018), trajectory-aware traces (Daley et al., 2023)) correct the mismatch step by step, and emphatic TD reweights updates (Sutton et al., 2016). Watkins's $Q(\lambda)$ cuts traces at non-greedy actions while Peng's $Q(\lambda)$ keeps them (Peng and Williams, 1996; Kozuno et al., 2021; Kim and hwan Oh, 2026); other work adapts the trace (White and White, 2016; Rowland et al., 2020), avoids ratios via action-dependent bootstrapping (Mahmood et al., 2017), gates traces at non-greedy actions (Daley, 2026), or adds multi-step inequalities (Abraham et al., 2026). However, these corrections need action probabilities or a discrete greedy target policy.

**Horizon reduction.** Instead of correcting within a backup, horizon-reduction methods change what a decision is. Hierarchical methods act over options or subgoals (Bacon et al., 2017; Vezhnevets et al., 2017; Nachum et al., 2018, 2019; Xu et al., 2022), and action chunking treats a short action sequence as one decision (Seo and Abbeel, 2024; Li et al., 2025b,a; Song et al., 2026). However, within each segment they use a fixed, uncorrected backup and inherit the bias; HQL is orthogonal and contains both as ungated special cases. Another line exploits the structure of goal-reaching tasks through quasimetric value functions (Wang et al., 2023a; Zheng et al., 2026), graph search over subgoals (Eysenbach et al., 2019), and goal-conditioned hierarchies (Park et al., 2023; Ahn et al., 2025; Zhou and Kao, 2025; Park et al., 2026). These methods rely on a goal-reaching reward and thus apply only to goal-conditioned RL; in contrast, HQL makes no assumption on the reward.

**Policy extraction with expressive policies.** Offline RL methods regularize the policy toward the data, either explicitly through divergence penalties (Wu et al., 2019; Kumar et al., 2019; Xu et al., 2021; Fujimoto and Gu, 2021) or implicitly through in-sample value learning and weighted behavior cloning (Peng et al., 2019; Nair et al., 2020; Kostrikov et al., 2022; Xu et al., 2022, 2023, 2025a). Recent work extracts expressive diffusion or flow policies by value-guided training (Wang et al., 2023b; Mao et al., 2024a; Park et al., 2025c), rejection sampling from a behavior model (Chen et al., 2023; Hansen-Estruch et al., 2023), adjoint matching (Li and Levine, 2026), or value gradient flow (Xu et al., 2026). These methods keep a one-step critic; HQL only changes the critic's target and can serve any of them.

## 3 Preliminaries

We consider the problem presented as a finite-horizon Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, H)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $P(s'|s,a)$ and $r(s,a)$ denote the transition dynamics and reward function, and $H$ is the horizon; an episode that reaches its goal early enters a zero-reward absorbing state, so every episode has length $H$. For a policy $\pi(a|s)$, the action value is $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_t r(s_t,a_t) \mid s_0 = s, a_0 = a]$, with $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)]$ and advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. We consider the off-policy setting, where one needs to estimate $Q^\pi$ of a target policy $\pi$ from a dataset (or replay buffer) $\mathcal{D}$ collected by a possibly suboptimal behavior policy $\mu(a|s)$.

### 3.1 Error Accumulation in One-Step and $n$-Step TD

**Error decomposition along a chain.** To understand how critic error scales with the horizon, we derive one error law for one-step and $n$-step backups (derivations in Appendices A.1–A.2). Consider the deterministic chain $s_0 \to s_1 \to \cdots \to s_H$ induced by a fixed deterministic target policy, with an exactly known terminal value. Write $Q_t = Q(s_t, a_t)$, let $r_t$ be the reward on the transition out of $s_t$, and let $\overline{Q}$ be the target critic, with $\overline{V}^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[\overline{Q}(s,a)]$. The $n$-step target is

$$Y_t^{(n)} = \sum_{j=0}^{n-1} r_{t+j} + \overline{V}^\pi(s_{t+n}), \tag{1}$$

where a true terminal state replaces the last term by its known value. The loss at lookahead $n$ is $L_{\mathrm{TD}}^{(n)} = \mathbb{E}_t[(Q_t - \mathrm{sg}(Y_t^{(n)}))^2]$, where sg stops gradients (Sutton and Barto, 2018); for $n = 1$ we write $L_{\mathrm{TD}}$. With the one-step Bellman residual $e_t = Q_t - r_t - Q_{t+1}$ and the value error $\epsilon_t = Q_t - Q_t^\pi$, we have $e_t = \epsilon_t - \epsilon_{t+1}$ because $Q_t^\pi = r_t + Q_{t+1}^\pi$, and with an exact terminal value ($\epsilon_H = 0$) the recursion telescopes:

$$\epsilon_t = \sum_{\tau=t}^{H-1} e_\tau. \tag{2}$$

In other words, the value error is *exactly* the sum of the downstream residuals, for any critic at any point in training.

**A unified error law.** To see how the residuals add up, we model them as random noise across training runs: each residual has zero mean and variance $\sigma^2$, and residuals at different steps are only weakly correlated (Appendix A.1). Under this model, applying Eq. 2 to $n$-step backups, one block of $n$ steps at a time, gives a single expression for the start-state error (Appendix A.2):

$$\mathrm{RMSE}(\epsilon_0)^2 \;\approx\; \underbrace{\frac{H}{n}\, L_{\mathrm{TD}}^{(n)}}_{\text{residual accumulation}} \;+\; \underbrace{\left(1 - \tfrac{1}{n}\right)^2 H^2\, \bar{A}_\mu^2}_{\text{behavior-action bias}}, \tag{3}$$

where $\bar{A}_\mu \le 0$ is the average advantage of the logged actions that the targets span. Intuitively, the first term decreases with $n$ while the second increases with $n$, so a longer backup trades residual accumulation for behavior-action bias.

**One-step TD: residuals accumulate as $\Theta(\sqrt{H})$.** A one-step target spans no logged action, so the bias term vanishes, and summing Eq. 2 under the residual model gives, as $H - t \to \infty$,

$$\mathrm{RMSE}(\epsilon_t) = \sqrt{\mathbb{E}[\epsilon_t^2]} \sim \sigma_{\mathrm{eff}} \sqrt{H - t}, \tag{4}$$

with $\sigma_{\mathrm{eff}}^2 = \sigma^2 + 2\sum_{\ell \ge 1} \mathrm{Cov}(e_t, e_{t+\ell})$ the long-run variance of the residuals. This result is important as it shows that a small training loss does not certify a small value error: since $L_{\mathrm{TD}} = \mathbb{E}[e_t^2] = \sigma^2$ measures only the per-step residual, Eq. 3 at $n = 1$ reads

$$\mathrm{RMSE}(\epsilon_0) = \sqrt{H\, L_{\mathrm{TD}}} \tag{5}$$

up to the factor $\sigma_{\mathrm{eff}}/\sigma$. Appendix A.1 discusses when this rate fails and how a discount $\gamma < 1$ caps it at a horizon of order $1/(1-\gamma)$.

**$n$-step TD: bias grows toward $H|\bar{A}_\mu|$.** Bootstrapping $n$ steps ahead divides the residual term by $n$. However, along a logged trajectory $Y_t^{(n)}$ now deviates from $Q^\pi(s_t, a_t)$ by $\sum_{j=1}^{n-1} A^\pi(s_{t+j}, a_{t+j})$ plus the bootstrap error at the endpoint (Eq. 14). For a near-optimal target policy every spanned advantage is at most zero, so this bias is systematic and does not average out; as $n$ grows, the square root of the second term of Eq. 3 approaches $H|\bar{A}_\mu|$. This indicates that no fixed $n$ suits all logged actions: the best lookahead $n$ changes from step to step, which calls for a per-transition rule to minimize the total error.

## 4 Horizon-Scalable Q-Learning

In this section, we introduce Horizon-Scalable Q-Learning (HQL). We begin with a gated multi-step target and design the gate based on value comparison (Section 4.1). We then show that this gate is a limiting case of per-decision importance sampling yet needs no action density (Section 4.2). Finally, we provide several practical considerations for HQL and discuss its connection to previous work (Section 4.3).

### 4.1 Gated Multi-Step Targets

Our key idea is to let a backup continue through a logged action only when that action is at least as good as the actions the target policy would take, i.e., its advantage under the target policy is non-negative, which keeps it out of the bias term of Eq. 3 while reducing the horizon of TD learning via multi-step bootstrapping. To do so, we attach a gate $c_i \in \{0, 1\}$ to every transition and define the HQL target backward along a logged segment,

$$Y_t^{\mathrm{HQL}} = \underbrace{r_t + \gamma\, \overline{V}^\pi(s_{t+1})}_{\text{one-step target}} + \gamma\, c_{t+1} \underbrace{\big(Y_{t+1}^{\mathrm{HQL}} - \overline{Q}(s_{t+1}, a_{t+1})\big)}_{\text{downstream correction}}, \tag{6}$$

with the correction set to zero at the end of the segment, and $\gamma$ kept for generality. Intuitively, the first term is the usual one-step bootstrap from the target policy, and the second term carries back whatever the logged continuation has learned beyond the critic's current estimate at $(s_{t+1}, a_{t+1})$. When the gate accepts $a_{t+1}$, this correction flows back and the target extends one more step; when it rejects $a_{t+1}$, the target is a plain one-step target, and the states after the cut still start their own targets (as shown in Figure 1, bottom). Unrolling the recursion, a run of accepted actions yields a multi-step target whose length is set by the data rather than fixed in advance. Note that $c \equiv 0$ recovers one-step TD and $c \equiv 1$ is the uncorrected multi-step return.

It remains to decide when the gate is open, i.e., whether the target policy would keep the logged action. Intuitively, a logged action is safe to follow if it is at least as good as what the policy itself would propose, which the critic can check by comparing the logged action with candidates drawn from the policy's own sampler:

$$c_i = \mathbb{I}\left[\overline{Q}(s_i, a_i) \;\ge\; \max_{1 \le m \le N} \overline{Q}\big(s_i, a_i^{(m)}\big)\right], \qquad a_i^{(m)} \sim \mu(\cdot|s_i), \tag{7}$$

where $\mu$ is the behavior policy that can be estimated using any advanced generative models from $\mathcal{D}$. In other words, the logged action is kept if it is at least as good as the best of $N$ policy proposals. Note that the gate depends only on an ordering of critic values, so it is invariant to the scale of $Q$, needs no behavior density, and has $N$ as its only hyperparameter; it costs $N$ behavior samples and critic evaluations per transition, the same as the best-of-$N$ bootstrap already requires.

**Algorithm 1** Horizon-Scalable Q-Learning (critic update)

**Require:** dataset $\mathcal{D}$, window length $H$, gate width $N$, target budget $B$

1: Sample $B/H$ windows $(s_t, a_t, r_t, \ldots, s_{t+H})$ of $H$ consecutive transitions from $\mathcal{D}$
2: At every position $i$: draw $a_i^{(1:N)} \sim \mu(\cdot|s_i)$ and compute the gate $c_i$ (Eq. 7)
3: For $j = H-1, \ldots, 0$: compute $Y_{t+j}^{\text{HQL}}$ by Eq. 6, with no correction at $j = H-1$
4: Regress all $B$ pairs $(s_{t+j}, a_{t+j})$ onto $\text{sg}(Y_{t+j}^{\text{HQL}})$; update $\overline{Q}$

### 4.2 A Density-Free Gate via Per-Decision Importance Sampling

A remaining question is whether cutting multi-step backups with this gate is principled, since cutting changes what a multi-step backup converges to. Per-decision importance sampling answers this (Precup et al., 2000; Munos et al., 2016): in the tabular setting, any gate $c_i \in [0, 1]$ with $c_i \leq \pi(a_i|s_i)/\mu(a_i|s_i)$ preserves $Q^\pi$ as the fixed point of Eq. 6, and we call such a gate *valid*. However, evaluating this bound requires the density ratio $\pi/\mu$, which is unavailable and hard to estimate (Mao et al., 2024b; Xu et al., 2025a) when using generative policies, or when policies are implicit, e.g., $\mu$ is a flow sampler and $\pi$ selects among its samples. Fortunately, the ratio has a closed form for the policy that behavior-regularized RL methods approximate, i.e., the exponential tilt of the behavior policy (Peters et al., 2010; Peng et al., 2019; Xu et al., 2023; Li and Levine, 2026),

$$\pi_\theta(a|s) = \frac{\mu(a|s)\exp(\tau Q(s,a))}{Z(s)}, \qquad Z(s) = \mathbb{E}_{a\sim\mu(\cdot|s)}\left[\exp(\tau Q(s,a))\right], \tag{8}$$

so the behavior density cancels and $\log(\pi/\mu)(a_i|s_i) = \tau Q(s_i, a_i) - \log Z(s_i)$. We estimate the normalizer from $N$ behavior samples as $\widehat{\log Z}(s_i) = \text{LSE}_m(\tau\overline{Q}(s_i, a_i^{(m)})) - \log N$, where $\text{LSE}_m(x_m) = \log\sum_{m=1}^N \exp(x_m)$ is the log-sum-exp, and clipping the ratio at one gives the largest valid gate

$$c_i = \exp\left(\min\left\{0,\, \tau\overline{Q}(s_i, a_i) - \widehat{\log Z}(s_i)\right\}\right). \tag{9}$$

This soft gate interpolates between two extremes. As $\tau \to 0$, every gate equals one and the target is an uncorrected multi-step return. As $\tau \to \infty$, the tilt concentrates on the best behavior-supported action, and because $\text{LSE}_m(x_m) \leq \max_m x_m + \log N$, the exponent is nonnegative at any $\tau$ when no candidate beats the logged action and tends to $-\infty$ otherwise, so the soft gate becomes exactly the value comparison of Eq. 7. Therefore, HQL's gate is the $\tau \to \infty$ limit of a valid gate, which is more scalable and robust since it avoids a temperature and can be computed without density. It also outperforms the soft gate at every $\tau$ we tried (Appendix A.7).

### 4.3 Practical Considerations

**Dense targets over a trajectory window.** Since HQL is not restricted to a fixed $n$-step backup, it does not need to compute a target only at the first state of a sampled segment. Instead, HQL samples a trajectory window $(s_t, a_t, r_t, \ldots, s_{t+H})$ of $H$ consecutive transitions and computes the gated multi-step target for *every* state–action pair in the window. More specifically, it computes the gates $c_i$ once at every position, evaluates Eq. 6 backward from the window end, and regresses each $(s_{t+j}, a_{t+j})$ onto its own target $Y_{t+j}^{\text{HQL}}$, $j = 0, \ldots, H-1$, masking positions beyond an episode boundary (Algorithm 1). By doing so, a window yields $H$ targets whose maximal reach decreases from $H$ to 1, and the gate can shorten each of them further. To enable a fair comparison, we hold the number of critic targets per update fixed: with a budget of $B$, one-step TD samples $B$ transitions and HQL samples $B/H$ windows.

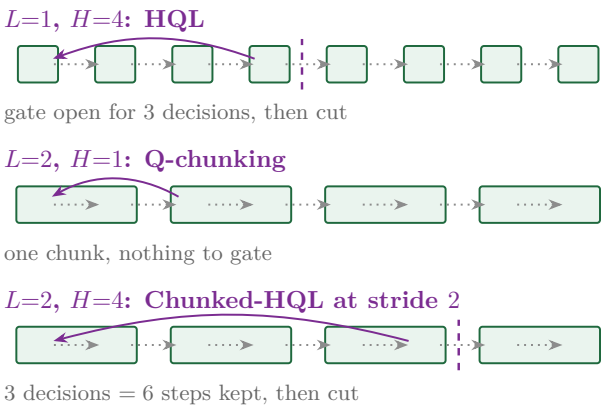



Figure 2: **HQL and Chunked-HQL.** $H$ sets how many *decisions* one target chains, and $L$ how many *environment steps* each decision covers.

**Multi-step decisions via Chunked-HQL.** So far the decision stride is fixed to 1, meaning each decision corresponds to a single environment step. To allow for more flexible decision-making, we let one *decision* span several environment steps. More specifically, with stride $L$, a decision is a chunk of $L$ primitive actions that the critic scores as a unit, $Q(s_t, a_{t:t+L})$, and a window of $H$ chunks reaches $L{\cdot}H$ environment steps while still producing $H$ targets. Note that the gate is unchanged, now comparing the logged chunk with the best-of-$N$ chunks drawn from the policy, and the backup of Eq. 6 holds with $\gamma^L$ in place of $\gamma$ and per-chunk reward sums.

**Connections to existing methods.** To summarize, in HQL or Chunked-HQL, $L$ sets how many environment steps one decision covers, and $H$ sets how many decisions one target may chain (Figure 2). With $L$=1, $H$=1 gives one-step TD, an ungated window of size $H$ gives uncorrected $H$-step TD, and a gated window gives HQL. With $L$=$k$, $H$=1 is Q-chunking with chunks of $k$ actions (Li et al., 2025b), since a single chunk leaves nothing to gate, and $H$>1 gives Chunked-HQL at stride $k$. Note that the two variables extend the maximal reach of a target, $L \cdot H$ environment steps, in different ways. Increasing $H$ lets one target makes more decisions while the policy outputs less actions per step, and the gate decides for each target how many of these decisions it actually uses. In contrast, increasing $L$ makes each decision cover more environment steps, but each chunk then runs open loop and the policy must search a larger, $L \cdot \dim(\mathcal{A})$-dimensional action space.

## 5 Experiments

The goal of our experiments is to answer four questions. **(Q1)** Does HQL enable better value learning under different horizon lengths and dataset suboptimality? **(Q2)** How does HQL compare with TD and other baselines on OGBench and OCBench? **(Q3)** Does HQL work with different policy-extraction methods? **(Q4)** How do the different components of HQL affect its performance?

### 5.1 Experimental Setup

**Combination lock.** For Q1, we use a modified `combination-lock` chain (Park et al., 2025b) with H linearly ordered states and two actions: the "correct" one advances to the next state and the other returns to the first state, with reward -1 per step and 0 at the terminal state. The offline dataset consists of length-n=64 windows (64 transitions each): for every state–action pair we roll out M=100 windows that start with that pair and then follow the correct action, truncating a window early if it reaches the terminal state. Following Park et al. (2025b), we construct datasets of $n$-step trajectory windows with uniform coverage of starting state-action pairs. To study the effect of increasing action corruption, we independently replace each of the $n-1=63$ subsequent actions with the non-optimal action with probability $p \in \{0, 0.002, 0.005, 0.011, 0.022\}$, chosen so that in expectation $(1-p)^{63} \in \{100, 88, 73, 50, 25\}\%$ of windows are clean, i.e. contain no corrupted action. We use $H \in \{512, 1024, 2048, 4096\}$, giving 2HM windows per dataset. Because the dynamics are deterministic, the ground-truth Q is available in closed form (A.3).

**OGBench.** Q2–Q4 use six state-based OGBench configurations (Park et al., 2025a): `cube-double`, `cube-triple`, `scene` (sparse reward), `humanoidmaze-medium`, `antmaze-giant` and `humanoidmaze-giant`, whose task horizons last 500 to 4000 steps (Table 5); we report the mean score over each configuration's five tasks. Runs train for 1M offline and then 1M online steps, with 50 evaluation episodes every $10^5$ steps; a run's score is the mean of its last three evaluations.

**OCBench.** To test HQL on even longer horizons, we also use OCBench (Park and Levine, 2026), a robotic manipulation benchmark whose datasets come from scripted policies with properties similar to human demonstrations. We take eight full-frequency (50 Hz) tasks from four domains (`block`, `chamber`, `switch`, `hanoi`), whose episodes last 1000 to 5000 steps. Following the RL baselines of OCBench, all methods use 25-step action chunks, $8 \times 2048$ residual MLPs, an HL-Gauss critic and $\gamma$=0.9999, with 1024 critic targets per update. Runs train for 1M offline and 1.5M online steps with 500 evaluation episodes every $5 \times 10^4$ steps (Appendix A.6).

**Baselines.** We compare HQL ($H = 16$) with existing baselines under a matched compute budget: every method receives 512 critic targets per gradient step. We consider three TD baselines. 16-window TD uses the same windows and batch as HQL but removes the gate, so comparing the two isolates the gate's effect. 8-step TD ($n$=8 matches the average reach of a 16-step window) and 1-step TD each regress a single target per sequence with a batch of 512. We also compare with Q-chunking (QC) (Li et al., 2025b), which treats a chunk of 8 actions as a single decision and bears no off-policy bias, and Long-horizon Q-Learning (LQL) (Abraham et al., 2026), which adds hinge penalties to one-step TD that enforce

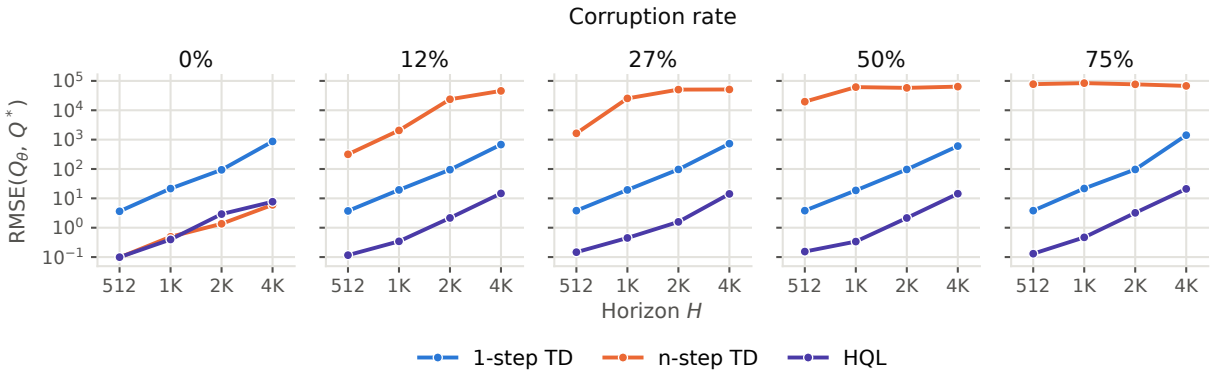



Figure 3: **Combination lock.** HQL remains accurate under data corruption as the horizon grows. The figure illustrates the RMSE between the learned $Q_\theta$ and the true $Q^*$, averaged uniformly over all $2H$ state–action pairs of the corrupted combination lock, after $1M$ updates (log scale). Panels vary the corruption fraction of the dataset. n-step TD ($n = 64$) diverges as soon as any corruption is present; 1-step TD is unaffected by corruption but its error grows sharply with the horizon $H$.

multi-step inequalities along the logged trajectory. All methods share the same critic architecture and use the same behavior flow policy with best-of-N sampling.

### 5.2 Q1: A Toy Example with Corrupted Windows

Figure 3 supports both halves of the analysis. On clean data, the error of 1-step TD grows by three orders of magnitude from $H$=512 to 4096, while 64-step TD and HQL avoid this accumulation. As the corruption rate increases, 64-step TD degrades sharply, because each target sums the negative advantages of the actions it spans. HQL cuts the trace at those actions: its error stays below both 1-step TD and 64-step TD at every horizon.

### 5.3 Q2: Results on OGBench and OCBench

**HQL is the strongest method across all six domains.** HQL reaches a mean success rate of 0.74 against 0.49 for the best TD baseline, 8-step TD (Figure 4). Consistent with our analysis in Section 3.1, the gap is largest on the four configurations with the largest horizons of 1000 steps or more (Table 5); on `cube-triple`, HQL reaches 0.54 while both multi-step baselines score 0. 16-window TD matches HQL in everything except the gate, yet it is the weakest method on average, this demonstrates the effectiveness of gating in HQL.

**Comparison with other baselines.** At the same budget and sampling width (Table 1), HQL improves the mean success rate by 12% over LQL and by 29% over QC. QC is best on `cube-double` and `scene` but scores 0% on both humanoid mazes.

Although Q-chunking avoids the off-policy bias by having the critic to score the whole L-sized action chunk $Q(s_t, a_{t:t+L})$, it falls short because the critic must choose among $L \cdot \dim(\mathcal{A})$-dimensional chunks, and a fixed best-of-$N$ search covers that space increasingly sparsely as the chunk size $L$ grows. Second, each chunk runs open loop for $L$-steps (we choose $L = 8$ while running QC), so the agent cannot correct its action within a chunk. This also explains why QC performs well on `cube-double` and `scene`, whose chunks are low-dimensional (please refer to table 5 for action dimensions of the different OGBench tasks), but trails on `antmaze-giant` and fails on both humanoid mazes. We expect a wider best-of-$N$ search to help QC, since it covers the chunk space more densely, though at a cost in samples and critic evaluations that grows with $N$. The gate is also complementary to chunking: adding it on top of chunked actions (Chunked-HQL) raises QC's success rate from 45% to 81% (Table 1).

Table 1: **HQL and Chunked-HQL against LQL and QC.** Success rates (%) are averaged over five tasks and reported as offline → online with standard error over five seeds; we bold the best offline and the best online score at integer level. HQL and Chunked-HQL achieve the best results, especially on the long-horizon tasks.

| | HQL | Chunked-HQL | LQL | QC |
|---|---|---|---|---|
| `antmaze-giant` | **1** → **87**±2 | 0 → 86±1 | 0 → 44±4 | 0 → 46±1 |
| `cube-double` | 67 → 87±1 | **73** → **100**±0 | 45 → 83±2 | 28 → **100**±0 |
| `cube-triple` | **12** → 54±1 | 7 → **84**±1 | 5 → 35±3 | 1 → 26±3 |
| `hmaze-giant` | 10 → 28±1 | 0 → **29**±2 | **12** → 26±1 | 0 → 0±0 |
| `hmaze-medium` | 56 → 86±1 | **68** → **88**±2 | 48 → 86±1 | 0 → 0±0 |
| `scene` | 93 → 99±0 | **98** → **100**±0 | 96 → 97±1 | 97 → **100**±0 |
| **mean** | 40 → 74±1 | **41** → **81**±1 | 34 → 62±1 | 21 → 45±1 |

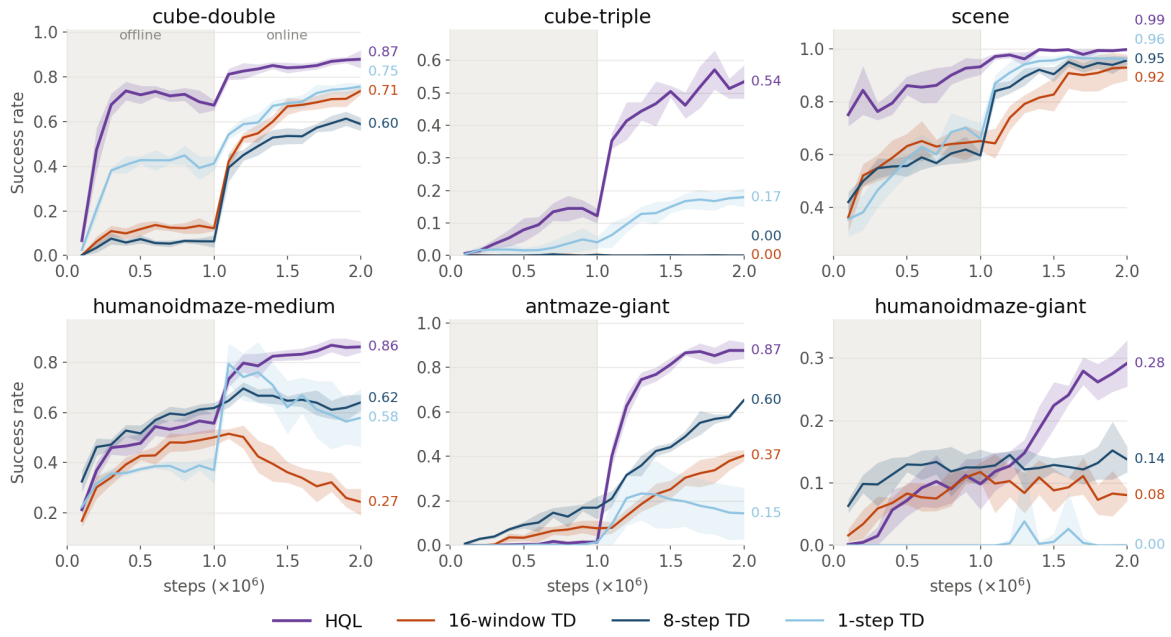



Figure 4: **HQL versus baseline TD methods at matched critic update budget.** Success rate averaged over the five tasks of each configuration and over 5 seeds, with standard deviation shown as the band; gray marks the offline phase. HQL outperforms all baselines on all six configurations, especially on the challenging long-horizon ones.

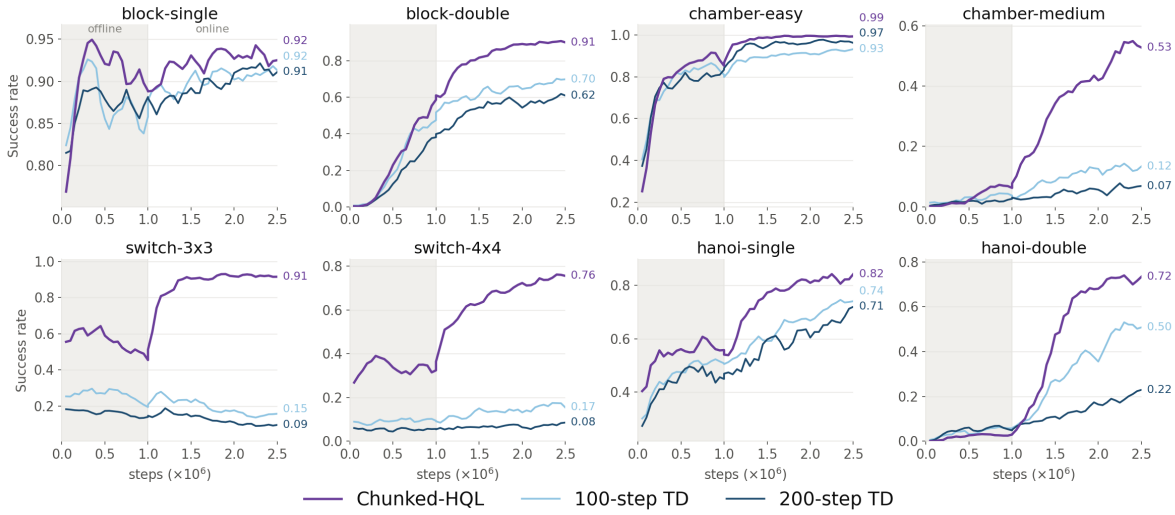



Figure 5: **Chunked-HQL versus $n$-step TD on OCBench.** Success rate on eight 50 Hz tasks (one seed, smoothed over three evaluations; numbers are the mean of the last three). All methods use 25-step chunks and the same best-of-16 bootstrap. Chunked-HQL matches or outperforms both baselines on all eight tasks, especially on `switch` and `chamber-medium`.

**Why does HQL help?** One might expect a multi-step method to win by sending reward information far back along the trajectory. To understand why HQL works better than the TD baselines, we log how many steps each target's trace survives (Figure 6). Perhaps surprisingly, the actual traces are short: 77% of targets are exactly one-step targets while 17% extend to two steps and 6% to three or more steps, and none exceeds six of the sixteen available steps. This indicates that HQL helps not by making targets longer but by choosing *which* ones to extend: $n$-step TD extends all of them and inherits the bias of every action spanned, 1-step TD extends none, and HQL extends a target only when its logged action passes the gate.

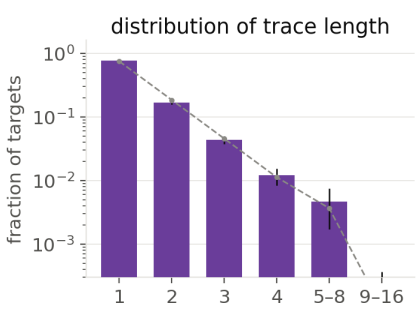



Figure 6: **Per-target trace length** with the memoryless prediction (dashed).

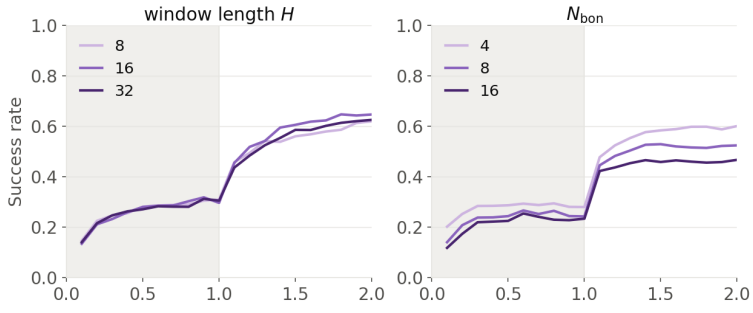



Figure 7: **HQL ablations** over window length $H$ (left) and best-of-$N$ gate width $N$ (right). $H$ matters less than $N$, and a smaller $N$ gives better results.

**Chunked-HQL scales to the longer horizons of OCBench.** On OCBench, we compare Chunked-HQL ($L$=25, $H$=8, i.e., a 200-step window) with $n$-step TD at $n \in \{100, 200\}$, which regresses the chunk critic onto an uncorrected $n$-step return and is the backup used by OCBench's own RL baselines ($n$=200). Motivated by Section 5.5, we decouple the two widths of HQL: the gate uses $N$=2, while the bootstrap and acting use best-of-16 as in the baselines. Chunked-HQL reaches a mean success rate of 0.82 at 2.5M steps against 0.53 for 100-step TD and 0.46 for 200-step TD (Figure 5). The gap is largest on `switch-3x3` (0.91 vs. 0.15), `switch-4x4` (0.76 vs. 0.17) and `chamber-medium` (0.53 vs. 0.12), where the $n$-step baselines barely improve or even degrade during online training. Note that 200-step TD is worse than 100-step TD on seven of the eight tasks, which agrees with Eq. 3: a longer uncorrected return sums the negative advantages of more logged actions, while the gate lets HQL extend a target only through chunks that pass the value comparison.

### 5.4 Q3: Combining HQL with Other Policy Extraction Methods

**HQL with QAM.** Q-learning with adjoint matching (QAM) (Li and Levine, 2026) trains a flow policy to sample from the tilted distribution $\pi \propto \mu \exp(\tau Q)$. To combine it with HQL, we keep the hard gate unchanged and only replace the best-of-$N$ policy with the QAM policy. With HQL as its critic, QAM improves on all four tasks at the end of online training, especially on `cube-triple` (1% → 80%) and `humanoidmaze-giant` (0% → 63%).

Table 2: **HQL as a drop-in critic for different policy extraction methods.** Success rates (%) on task 2, offline → online; we bold the better score of each pair. HQL improves both QAM and VGF on all four tasks.

| | cube-double | cube-triple | antmz-giant | hmz-giant |
|---|---|---|---|---|
| QAM | 29 → 50 | 1 → 1 | 14 → 83 | 1 → 0 |
| + HQL | **38 → 99** | 0 → **80** | 0 → **98** | 0 → **63** |
| VGF | 69 → 74 | 3 → 25 | 1 → 28 | 0 → 0 |
| + HQL | **80 → 89** | **4 → 37** | 0 → **90** | 0 → **4** |

**HQL with value gradient flow (VGF).** VGF (Xu et al., 2026) can be thought of an implicit policy that transports from behavior samples toward high-value actions with a few steps of $Q$-gradient ascent. With HQL as its critic, VGF again improves on all four tasks, most on `antmaze-giant` (28% → 90%).

### 5.5 Q4: Ablations of HQL

**Does the window length $H$ matter?** We find the effect is small. At a fixed budget of 512 targets per update, $H \in \{8, 16, 32\}$ (batch 64/32/16) gives similar performance: 0.607, 0.646, and 0.621 (Figure 7, left). This agrees with the trace length analysis in Section 5.3: since empirically no target used more than six steps, a window of 16 already exceeds what the gate uses. HQL is therefore not sensitive to $H$ unless it is too small.

**Does the best-of-$N$ width matter for HQL?** We try best-of-$N$ width from $\{4, 8, 16\}$. On OGBench datasets we find that a smaller $N$ is better: $N \in \{4, 8, 16\}$ gives the mean value of 0.596, 0.521, and 0.461, three times the spread of $H$ (Figure 7, right). Intuitively, a logged action exchangeable with the $N$ proposals beats all of them with probability $1/(N+1)$, so a larger $N$ accepts fewer logged actions and HQL approaches 1-step TD when $N$ is large. We provide more details on the ablation setup in Appendix A.8.

## 6 Conclusion and Limitations

In this paper, we propose HQL, which shortens the effective horizon of Q-learning without inheriting the bias of suboptimal logged actions. HQL gates each logged action with a density-free value comparison and achieves strong results on long-horizon OGBench tasks, both alone and as a drop-in critic for QAM and VGF. We believe that HQL represents a concrete step toward horizon-scalable off-policy RL. One limitation is that the gate relies on the critic's own ranking. One future work is to apply HQL over subgoals in a hierarchical manner.


## Acknowledgement

We thank Claas Voelcker and members from MIDI lab for discussions on the method and feedback on the early draft of the paper. HX is supported by the Amazon Fellowship. This research used the computational cluster resource provided by the Texas Advanced Computing Center at UT Austin.


## References


Armaan A. Abraham, Lucy Xiaoyang Shi, and Chelsea Finn. Long-horizon Q-learning: Accurate value learning via $n$-step inequalities. *arXiv preprint arXiv:2605.05812*, 2026.

Hongjoon Ahn, Heewoong Choi, Jisu Han, and Taesup Moon. Option-aware temporally abstracted value for offline goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, 2017.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Brett Daley. Gated Q-learning: Add off-policy bias to taste. In *Reinforcement Learning Journal*, volume 7, 2026.

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.

Kristopher De Asis, J. Fernando Hernandez-Garcia, G. Zacharias Holland, and Richard S. Sutton. Multi-step reinforcement learning: A unifying algorithm. In *AAAI Conference on Artificial Intelligence*, 2018.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

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

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

J. Fernando Hernandez-Garcia and Richard S. Sutton. Understanding multi-step deep reinforcement learning: A systematic study of the dqn target. *arXiv preprint arXiv:1901.07510*, 2019.

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*, 2018.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Byeongchan Kim and Min hwan Oh. Peng's q($\lambda$) for conservative value estimation in offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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($\lambda$) for modern reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

Aviral Kumar, Justin Fu, 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.

Cassidy Laidlaw, Stuart Russell, and Anca Dragan. Bridging RL theory and practice with the effective horizon. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled Q-Chunking. *arXiv preprint arXiv:2512.10926*, 2025a.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, 2025b.

Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ashique Rupam Mahmood, Huizhen Yu, and Richard S. Sutton. Multi-step off-policy learning without importance sampling ratios. *arXiv preprint arXiv:1702.03006*, 2017.

Liyuan Mao, Haoran Xu, Xianyuan Zhan, Weinan Zhang, and Amy Zhang. Diffusion-DICE: In-sample diffusion guidance for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. ODICE: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. In *International Conference on Learning Representations (ICLR)*, 2024b.

Rémi Munos. Error bounds for approximate value iteration. In *AAAI Conference on Artificial Intelligence*, 2005.

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.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Seohong Park and Sergey Levine. Behavioral cloning mystery. *arXiv preprint arXiv:2610.07056*, 2026.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes RL scalable. In *Advances in Neural Information Processing Systems*, 2025b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *International Conference on Machine Learning (ICML)*, 2025c.

Seohong Park, Deepinder Mann, and Sergey Levine. Dual goal representations. In *International Conference on Learning Representations (ICLR)*, 2026.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

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

Mark Rowland, Will Dabney, and Rémi Munos. Adaptive trade-offs in off-policy learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Younggyo Seo and Pieter Abbeel. Coarse-to-fine Q-network with action sequence for data-efficient reinforcement learning. *arXiv preprint arXiv:2411.12155*, 2024.

Gwanwoo Song, Kwanyoung Park, and Youngwoon Lee. Chunk-guided Q-Learning. *arXiv preprint arXiv:2603.13971*, 2026.

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

Richard S. Sutton, A. Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, 2016.

Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent. Convergent tree backup and Retrace with function approximation. In *International Conference on Machine Learning (ICML)*, 2018.

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

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023a.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023b.

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

Martha White and Adam White. A greedy approach to adapting the trace parameter for temporal difference learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2016.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Haoran Xu, Xianyuan Zhan, Jianxiong Li, and Honglei Yin. Offline reinforcement learning with soft behavior regularization. In *NeurIPS Offline Reinforcement Learning Workshop*, 2021.

Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline RL with no OOD actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations (ICLR)*, 2023.

Haoran Xu, Shuozhe Li, Harshit Sikchi, Scott Niekum, and Amy Zhang. An optimal discriminator weighted imitation perspective for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025a.

Haoran Xu, Liyuan Mao, Hui Jin, Weinan Zhang, Xianyuan Zhan, and Amy Zhang. Uni-RL: Unifying online and offline RL via implicit value regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Haoran Xu, Kaiwen Hu, Somayeh Sojoudi, and Amy Zhang. Reinforcement learning via value gradient flow. In *International Conference on Learning Representations (ICLR)*, 2026.

Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. DeepThermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2022.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

Bill Chunyuan Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Multistep quasimetric learning for scalable goal-conditioned reinforcement learning. In *International Conference on Learning Representations*, 2026.

John L. Zhou and Jonathan C. Kao. Flattening hierarchies with policy bootstrapping. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

# A Appendix

## A.1 Derivation of the One-Step Error Accumulation

This appendix states the residual model behind Eq. 4 and derives the rate. For a family of chains with growing horizon, assume the one-step residuals $e_t = Q_t - r_t - Q_{t+1}$ form a second-order stationary process whose statistics do not depend on $H$:

$$\mathbb{E}[e_t] = 0, \qquad \mathrm{Var}(e_t) = \sigma^2, \qquad C(\ell) = \mathrm{Cov}(e_t, e_{t+\ell}), \tag{10}$$

with weak correlation in the form $\sum_{\ell=1}^{\infty} |C(\ell)| < \infty$. Taking the mean and variance of the sum in Eq. 2 with $n = H - t$ terms gives

$$\mathbb{E}[\epsilon_t] = 0, \tag{11}$$

$$\mathrm{Var}(\epsilon_t) = n\sigma^2 + 2\sum_{\ell=1}^{n-1}(n-\ell)C(\ell). \tag{12}$$

By summability of $|C(\ell)|$, as $n \to \infty$,

$$\frac{\mathrm{Var}(\epsilon_t)}{n} \longrightarrow \sigma_{\mathrm{eff}}^2 := \sigma^2 + 2\sum_{\ell=1}^{\infty} C(\ell), \tag{13}$$

the long-run variance, which is necessarily nonnegative. If $\sigma_{\mathrm{eff}}^2 > 0$, then $\mathrm{RMSE}(\epsilon_t) \sim \sigma_{\mathrm{eff}}\sqrt{H-t}$, which is Eq. 4; this is the nondegenerate random-walk regime.

**When the rate does not apply.** The random-walk rate is conditional on the residual model and is not a worst case. Negative correlation can give $\sigma_{\mathrm{eff}}^2 = 0$ and $o(\sqrt{H-t})$ error, while a systematic same-sign residual of horizon-independent size gives $\Theta(H)$. Such a residual arises, for example, when the critic's values are shrunk toward zero under all-negative rewards: the semi-gradient fixed point with shrinkage $\kappa$ satisfies $e_t = -\kappa Q_t > 0$ at every step. Accumulation also requires persistent residuals: in a fully expressive tabular chain, exact convergence gives $Q = Q^\pi$, $e_t = 0$, and no accumulation, whereas function approximation, finite optimization, and target-network lag all leave residuals.

**Discounting.** With $\gamma < 1$ the telescoped sum is geometrically weighted, $\epsilon_t = \sum_{k=0}^{H-t-1} \gamma^k e_{t+k}$, and for uncorrelated residuals $\mathrm{Var}(\epsilon_t) = \sigma^2(1-\gamma^{2(H-t)})/(1-\gamma^2)$. The factor grows linearly in $H-t$ until $H-t$ is of order $1/(1-\gamma)$ and saturates at $1/(1-\gamma^2)$ afterwards, so the discount caps the accumulation at its own effective horizon ($\approx 100$ steps at $\gamma = 0.995$) rather than removing it; the analysis applies with $H$ replaced by $\min\{H, 1/(1-\gamma)\}$ up to constants.

## A.2 Derivation of the $n$-Step Error Accumulation

**Single-target decomposition.** With $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ and endpoint error $\epsilon_V(s) = \overline{V}^\pi(s) - V^\pi(s)$, expanding the target of Eq. 1 over the behavior continuation gives

$$\mathbb{E}_\mu\left[Y_t^{(n)} - Q^\pi(s_t,a_t) \,\middle|\, s_t, a_t\right] = \sum_{j=1}^{n-1} \mathbb{E}_\mu\left[A^\pi(s_{t+j}, a_{t+j}) \,|\, s_t, a_t\right] + \mathbb{E}_\mu\left[\epsilon_V(s_{t+n}) \,|\, s_t, a_t\right]. \tag{14}$$

The derivation uses only $r_{t+j} + V^\pi(s_{t+j+1}) = Q^\pi(s_{t+j}, a_{t+j}) = V^\pi(s_{t+j}) + A^\pi(s_{t+j}, a_{t+j})$ at each spanned transition (deterministic transitions given the action; the $j = 0$ term contributes no advantage because the target evaluates $Q^\pi(s_t, a_t)$ at the logged action itself). The first term is the behavior-action bias inside the segment, the second the endpoint bootstrap error.

**Chaining blocks along the horizon.** To turn the per-target decomposition into a start-state statement, partition the chain into $H/n$ blocks of length $n$ (assume $n \mid H$ for clarity) and define the $n$-step training residual $\tilde{e}_t = Q_t - Y_t^{(n)}$ at each block start. Adding and subtracting $Q^\pi$ and applying Eq. 14,

$$\epsilon_t = \tilde{e}_t + b_t(n) + \epsilon_{t+n}, \qquad b_t(n) = \sum_{j=1}^{n-1} A^\pi(s_{t+j}, a_{t+j}), \tag{15}$$

where we identify the endpoint state-value error with the action-value error at the next block start (exact on the deterministic evaluation chain). Iterating over blocks with $\epsilon_H = 0$,

$$\epsilon_0 = \sum_{j=0}^{H/n-1} \left[\tilde{e}_{jn} + b_{jn}(n)\right]. \tag{16}$$

Model the training residuals $\tilde{e}_{jn}$ as zero-mean, weakly correlated across blocks with per-block variance $\sigma_n^2 = L_{\mathrm{TD}}^{(n)}$ (the analogue of the model in Appendix A.1); the bias terms $b_{jn}(n)$, by contrast, share a sign for a near-optimal target policy and add coherently. Then

$$\mathrm{RMSE}(\epsilon_0)^2 \approx \frac{H}{n}\,\sigma_n^2 + \left(\sum_{j=0}^{H/n-1} b_{jn}(n)\right)^2, \tag{17}$$

and since the bias sum covers all logged actions except the $H/n$ block starts, $\sum_j b_{jn}(n) = (1-\frac{1}{n})H\bar{A}_\mu$ with $\bar{A}_\mu$ the average spanned advantage, which is Eq. 3. The result is schematic in the same sense as Appendix A.1: it conditions on the residual model, ignores cross terms between residuals and biases, and uses block-aligned targets, but none of these affect the two scalings it exhibits—a $\sqrt{H/n}$ random-walk term and a bias term that grows with $n$ toward the horizon-linear ceiling $H|\bar{A}_\mu|$.

### A.3 Combination-Lock Dataset

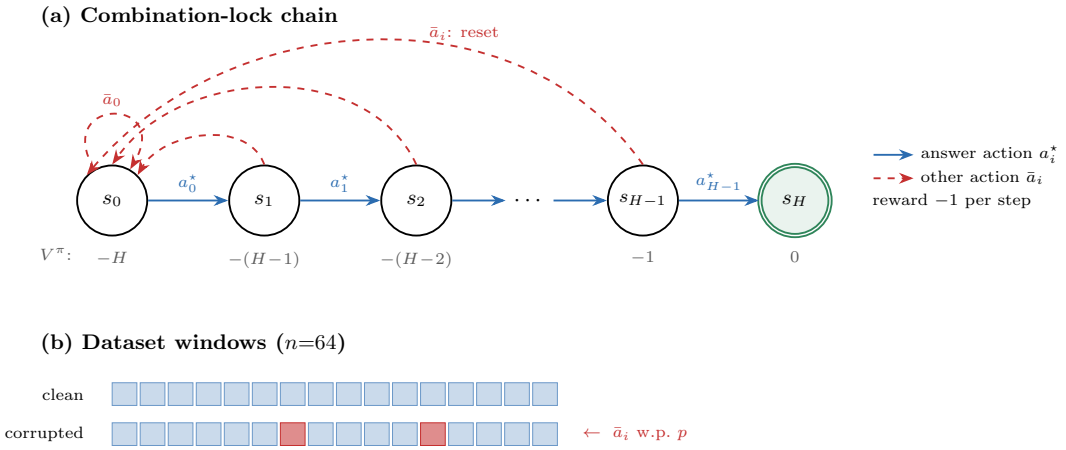



Figure 8: The combination lock dataset with linearly arranged H states.

In this didactic setup (Figure 8), the policy $\pi$ is the deterministic answer policy, and the dynamics are deterministic and acyclic under $\pi$, so the action-value function $Q^\pi$ has a closed form. We anchor at the terminal state, $V^\pi(s_H) = 0$. The Bellman equation then reduces to the recursion $V^\pi(s_i) = -1 + V^\pi(s_{i+1})$, which unrolls to

$$V^\pi(s_i) = -(H-i), \qquad i = 0, \ldots, H-1. \tag{18}$$

Taking the answer action costs one step and advances the chain. Any other action costs one step and resets to the start. Hence

$$Q^\pi(s_i, a_i^\star) = -1 + V^\pi(s_{i+1}) = -(H-i), \tag{19}$$

$$Q^\pi(s_i, \bar{a}_i) = -1 + V^\pi(s_0) = -(H+1). \tag{20}$$

### A.4 OGBench Benchmark: Experiment Details and Per-Task Results

This appendix documents the configuration behind the OGBench benchmark figure of Section 5.3 (Figure 4) and reports the per-task curves that its task means aggregate over (Figure 9).

**Tasks and datasets.** We use six environment–discount configurations: `cube-double`, `cube-triple`, `scene`, and `humanoidmaze-medium` at $\gamma = 0.99$, and `antmaze-giant` and `humanoidmaze-giant` at $\gamma = 0.995$; Table 5 lists dataset sizes, episode lengths, and dimensionalities. Every group provides five single-task variants that share one dataset and differ only in the reward relabelling, and we run all five, so each panel of Figure 4 aggregates five tasks. `scene` uses the sparse reward relabelling.

**Training runs.** Each run trains for $10^6$ offline gradient steps followed by $10^6$ online steps, evaluating every $10^5$ steps with 50 episodes; the reported score is the mean success rate over the last three evaluations. Each method is run across the five different tasks corresponding to each of the six OGBench environments for five seeds. Curves show the mean over seeds of the task-averaged score, with bands of $\pm 1$ standard deviation across seeds; we average over tasks before computing seed statistics because per-task success is bimodal, so the across-task spread would swamp seed variance. Figure 9 gives the per-task breakdown.

**HQL configuration.** HQL uses a window of $H = 16$ with 32 windows per batch, i.e., $32 \times 16 = 512$ critic targets per update under the accounting of Section 4.3. The gate is the best-of-$N$ indicator of Eq. 7 with $N = 4$, computed from the target critic. The policy is a behavior-cloned flow sampler with best-of-$N$ action selection ($N = 4$) used both for acting and for the bootstrap $\overline{V}^{\pi}$, so the gate, the bootstrap, and the actor all refer to the same target policy. Table 4 lists the full configuration.

**Baseline configurations.** The three TD baselines share all networks and optimization settings with HQL (Table 4) and differ only in how their 512 critic targets per update are formed (Table 3).

Table 3: How each method forms its 512 critic targets per update.

| Method | Seq. length | Seq. per update | Targets per seq. | Target |
|---|---|---|---|---|
| HQL | 16 | 32 | 16 | gated, every position |
| 16-window TD | 16 | 32 | 16 | $n$-step, every position |
| 8-step TD | 8 | 512 | 1 | 8-step, first state |
| 1-step TD | 1 | 512 | 1 | 1-step |

Table 4: Hyperparameters for the OGBench benchmark experiment (Figures 4 and 9).

| Parameter | Value |
|---|---|
| Window length $H$ | 16 |
| Windows per batch | 32 |
| Critic targets per update | 512 $(= 32 \times 16)$ |
| Gate width $N$ (Eq. 7) | 4 |
| Gate critic | target network $\overline{Q}$ |
| Discount factor | 0.99 (0.995 for the two giant mazes) |
| Optimizer | Adam (Kingma and Ba, 2014) |
| Learning rate | $3 \times 10^{-4}$ |
| Target network update rate | $5 \times 10^{-3}$ |
| Critic ensemble size | 2 |
| Target $Q$ aggregation | mean |
| Network width | 512 |
| Network depth | 4 |
| Activation function | GELU (Hendrycks and Gimpel, 2016) |
| Critic layer norm | Yes |
| Actor layer norm | No |
| Number of flow steps (behavior sampler) | 10 |
| Best-of-$N$ action samples (actor and bootstrap) | 4 |
| Action chunk size | 1 (no action chunking) |
| UTD ratio | 1 |
| Offline training steps | $10^6$ |
| Online training steps | $10^6$ |
| Evaluation interval / episodes | $10^5$ / 50 |
| Seeds per cell | 5 (seeds 0–4) |

### A.5 Shared Implementation Settings

Unless a later subsection states otherwise, every experiment uses the networks, optimizer and behavior flow sampler of Table 4. All agents are implemented in one JAX codebase with shared replay, evaluation and logging, so two compared methods differ only in the settings named for that comparison.

**Matched budget.** All comparisons in Section 5.3 and the window-length ablation of Section 5.5 fix the number of critic targets per update at 512, following the accounting of Park et al. (2025b): a batch of $B$ windows of length $H$ yields $B \cdot H$ targets for methods that regress every position, and $B$ targets for methods that regress only the first state. The number of transitions read per update therefore differs across methods; for example, 8-step TD reads 4096 transitions to form its 512 targets.

**Compute.** Runs of $2 \times 10^6$ steps were executed on 8-GPU A100 and H100 nodes, eight runs per node, at about 150 iterations/s per run (≈4–6 h per run).

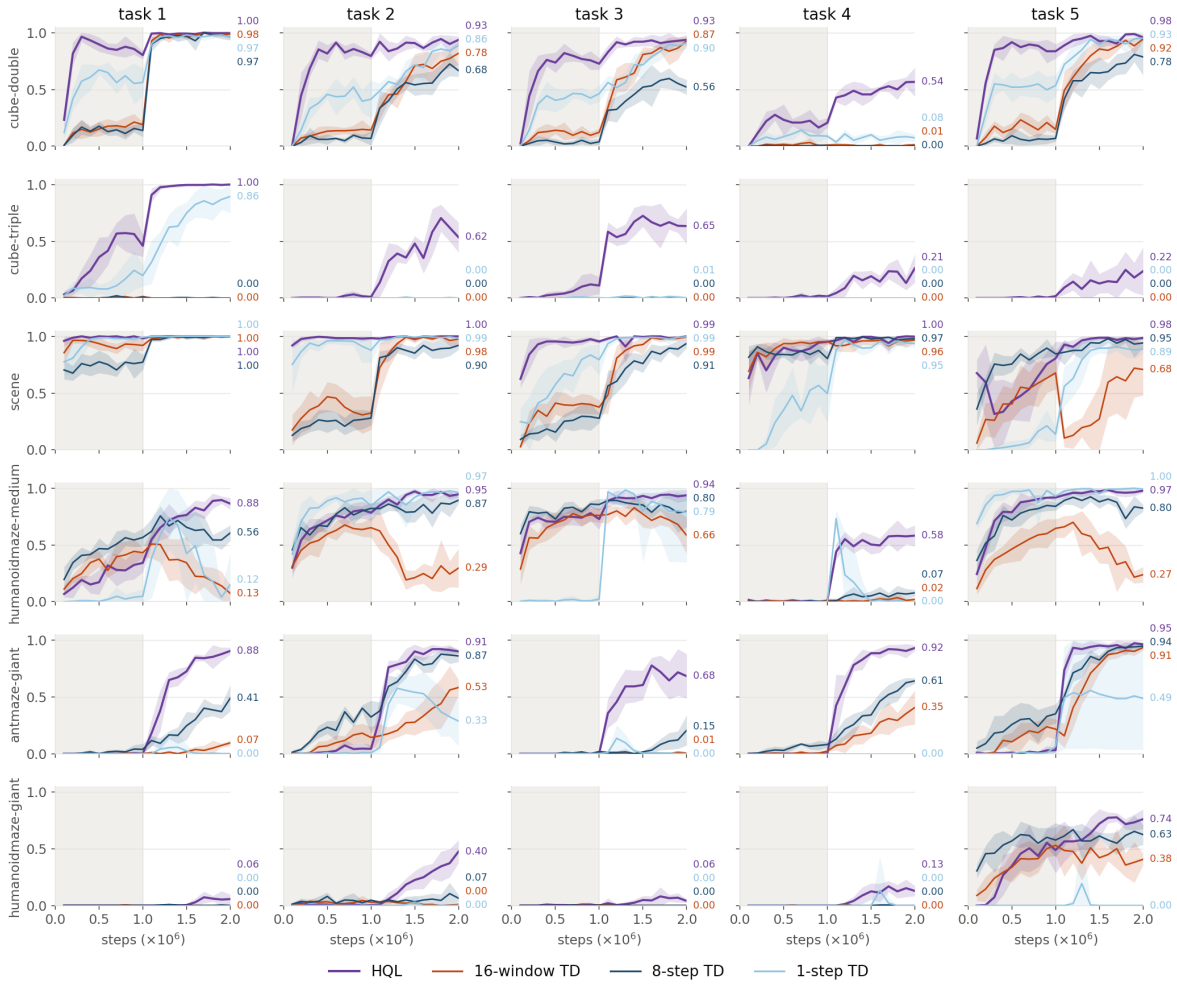



Figure 9: **Per-task view of the OGBench benchmark (Figure 4).** Reading down a column shows how one task index behaves across environments, and a row shows how one environment varies over tasks.

### A.6 OCBench Benchmark: Experiment Details

This appendix documents the configuration behind the OCBench results of Section 5.3 (Figure 5). Unlike the OGBench experiments, we follow the RL baselines of OCBench (Park and Levine, 2026) for every setting other than the critic target, so all three methods share the same networks, critic loss, action space and action chunks (Table 6).

**Tasks and datasets.** We use eight full-frequency (50 Hz) tasks, `block-single`, `block-double`, `chamber-easy`, `chamber-medium`, `switch-3x3`, `switch-4x4`, `hanoi-single` and `hanoi-double` (all `task1`), whose episode limits range from 1000 to 5000 steps. We train on all episodes of each dataset without filtering for successful ones. Actions are normalized as in OCBench, i.e., the 1st and 99th percentiles of each action dimension are mapped to $\pm 1$ and the result is clipped at $\pm 5$.

**Method configurations.** Chunked-HQL uses chunks of $L = 25$ actions and windows of $H = 8$ chunks (200 environment steps) with 128 windows per batch, i.e., $128 \times 8 = 1024$ critic targets per update. Its gate compares the logged chunk with the best of $N = 2$ policy chunks, while the bootstrap and acting take the best of 16 chunks, the same width as the baselines. The $n$-step TD baselines ($n \in \{100, 200\}$) regress a single target per sequence with a batch of 1024. Each run is evaluated with 500 episodes every $5 \times 10^4$ steps, and the reported score is the mean success rate over the last three evaluations.

### A.7 Soft Gate versus Hard Gate

Section 4.2 shows that the hard gate of Eq. 7 is the $\tau \to \infty$ limit of the soft gate of Eq. 9. A natural question is whether a finite $\tau$ works better, since the soft gate keeps partial credit for logged actions that are close to the best candidate. To answer this, we compare the soft gate at $\tau \in \{0, 0.5, 1, 3, 30\}$ with HQL's hard gate on six OGBench configurations with five tasks each. Note that $\tau$=0 sets every gate to

Table 5: OGBench state-based task characteristics. Episode length is the maximum number of environment steps per episode. Rows marked † are the groups used in this work.

| Task group | Dataset size (transitions) | Episode length | Action dimension | Observation dimension | Default task |
|---|---|---|---|---|---|
| *Locomotion* (`locomaze`) | | | | | |
| `pointmaze-medium` | 1M | 1000 | 2 | 2 | 1 |
| `pointmaze-large` | 1M | 1000 | 2 | 2 | 1 |
| `pointmaze-giant` | 1M | 1000 | 2 | 2 | 1 |
| `pointmaze-teleport` | 1M | 1000 | 2 | 2 | 1 |
| `antmaze-medium` | 1M | 1000 | 8 | 29 | 1 |
| `antmaze-large` | 1M | 1000 | 8 | 29 | 1 |
| `antmaze-giant`† | 1M | 1000 | 8 | 29 | 1 |
| `antmaze-teleport` | 1M | 1000 | 8 | 29 | 1 |
| `humanoidmaze-medium`† | 2M | 2000 | 21 | 69 | 1 |
| `humanoidmaze-large` | 2M | 2000 | 21 | 69 | 1 |
| `humanoidmaze-giant`† | 4M | 4000 | 21 | 69 | 1 |
| `antsoccer-arena` | 1M | 1000 | 8 | 42 | 4 |
| `antsoccer-medium` | 4M | 1000 | 8 | 42 | 4 |
| *Manipulation* (`manipspace`) | | | | | |
| `cube-single` | 1M | 200 | 5 | 28 | 2 |
| `cube-double`† | 1M | 500 | 5 | 37 | 2 |
| `cube-triple`† | 3M | 1000 | 5 | 46 | 2 |
| `cube-quadruple` | 5M | 1000 | 5 | 55 | 2 |
| `cube-octuple` | – | 1500 | 5 | – | 2 |
| `scene`† | 1M | 750 | 5 | 40 | 2 |
| `puzzle-3x3` | 1M | 500 | 5 | 55 | 4 |
| `puzzle-4x4` | 1M | 500 | 5 | 83 | 4 |
| `puzzle-4x5` | 3M | 1000 | 5 | 99 | 2 |
| `puzzle-4x6` | – | 1000 | 5 | – | 2 |

one and recovers uncorrected 8-step TD. All runs use $H$=8, a batch of 128 windows (1024 critic targets per update), 1M offline and 1M online steps, and one seed; the soft gate estimates $\log \hat{Z}$ and acts with 16 behavior samples, and the hard gate uses $N$=4 for both.

Figure 10 and Table 7 show that the hard gate achieves the best mean success rate (61%), and that the soft gate improves steadily as $\tau$ increases (36% $\rightarrow$ 57%), i.e., as it approaches the hard gate. The gap is largest on the long-horizon mazes: on `humanoidmaze-medium` the hard gate reaches 91% against 77% for the best soft gate, and on `antmaze-giant` 65% against 58%. The soft gate is only better on `cube-double` at $\tau$=1, while `scene` and `puzzle-3x3` end at the same value for every gate. This indicates that the hard gate is not only the limit that removes $\tau$ as a hyperparameter, but also the strongest member of the family: a finite $\tau$ keeps partial credit for suboptimal logged actions, so the multi-step target inherits part of their bias, which is exactly the failure mode of uncorrected $n$-step TD.

Table 6: Hyperparameters for the OCBench benchmark experiment (Figure 5).

| Parameter | Value |
|---|---|
| Action chunk size $L$ | 25 (cut at episode boundaries) |
| Window length $H$ (Chunked-HQL) | 8 chunks (200 environment steps) |
| Windows per batch (Chunked-HQL) | 128 |
| Critic targets per update | 1024 $(= 128 \times 8)$ |
| Batch size of $n$-step TD | 1024 (one target per sequence) |
| $n$ of $n$-step TD | 100, 200 |
| Gate width $N$ (Eq. 7) | 2 |
| Gate critic | target network $\overline{Q}$ |
| Best-of-$N$ (actor and bootstrap) | 16 |
| Discount factor | 0.9999 |
| Optimizer | Adam (Kingma and Ba, 2014) |
| Learning rate | $10^{-4}$ |
| Target network update rate | $10^{-3}$ |
| Critic ensemble size | 2 |
| Target $Q$ aggregation | mean |
| Network (actor and critic) | residual MLP, $8 \times 2048$ |
| Activation function | GELU (Hendrycks and Gimpel, 2016) |
| Critic layer norm | Yes |
| Actor layer norm | Yes |
| Critic loss | HL-Gauss, 101 bins on $[0, 1]$, $\sigma = 0.75$ bins |
| Number of flow steps (behavior sampler) | 10 |
| Flow-time distribution | Beta$(1, 1.5)$ |
| Action normalization | percentiles $1/99 \mapsto \pm 1$, clip at $\pm 5$ |
| UTD ratio | 1 |
| Offline training steps | $10^6$ |
| Online training steps | $1.5 \times 10^6$ |
| Evaluation interval / episodes | $5 \times 10^4$ / 500 |
| Seeds per cell | 1 (seed 0) |

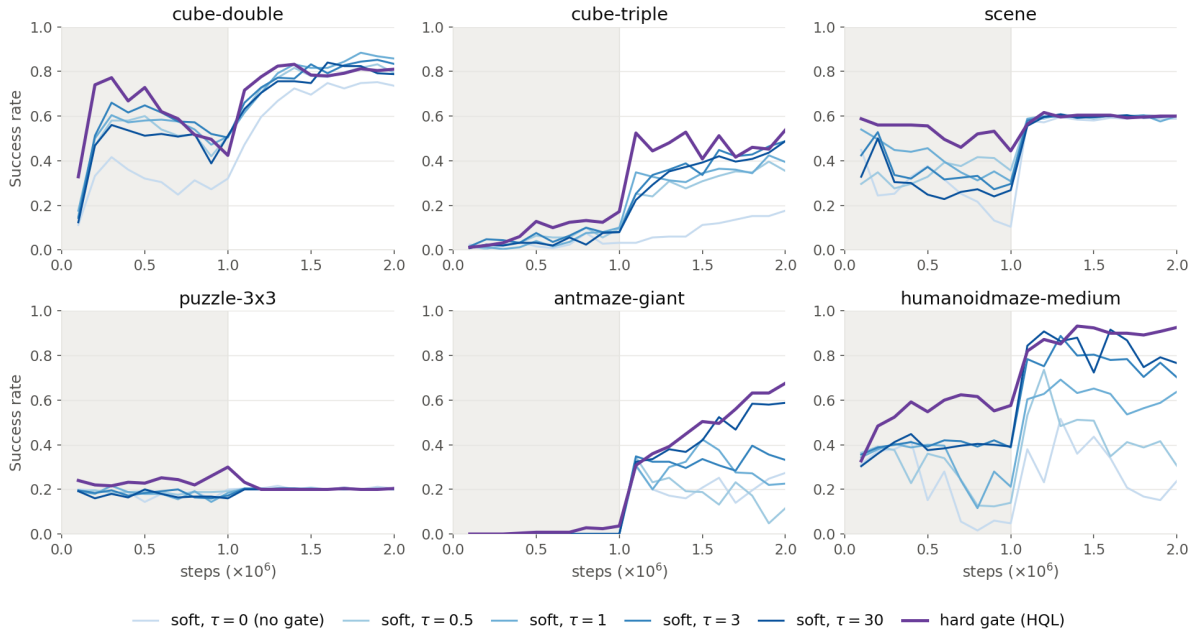



Figure 10: **Soft gate at different temperatures versus HQL's hard gate.** Success rate averaged over the five tasks of each configuration, one seed; gray marks the offline phase. The soft gate improves as $\tau$ grows toward the hard gate, and the hard gate is the best on average, especially on the long-horizon mazes.

Table 7: **Soft gate versus hard gate.** Success rate (%) at the end of online training, mean of the last three evaluations over five tasks, one seed. We bold the best score in each row at integer level.

| | soft gate | | | | | hard gate |
|---|---|---|---|---|---|---|
| | $\tau$=0 | $\tau$=0.5 | $\tau$=1 | $\tau$=3 | $\tau$=30 | (HQL) |
| cube-double | 75 | 81 | **87** | 84 | 80 | 81 |
| cube-triple | 16 | 37 | 39 | 46 | 44 | **48** |
| scene | 59 | **60** | 59 | **60** | **60** | **60** |
| puzzle-3x3 | **21** | 20 | 20 | 20 | 20 | 20 |
| antmaze-giant | 24 | 11 | 24 | 36 | 58 | **65** |
| hmaze-medium | 19 | 37 | 60 | 72 | 77 | **91** |
| **mean** | 36 | 41 | 48 | 53 | 57 | **61** |

### A.8 Ablation Details

Both panels of Figure 7 use HQL with the shared settings of Appendix A.5. Each panel averages only over the environment–task cells that were run for every value of the ablated setting, so that only that setting varies within a panel.

**Window length $H$.** We fix the budget at 512 critic targets per update, so the number of windows per batch decreases as $H$ increases (64, 32 and 16 for $H$=8, 16 and 32 respectively). The panel covers six environment–discount configurations with five tasks each, i.e., 30 cells.

**Gate width $N$.** We run this panel at $H$=8 with 128 windows per batch, i.e., 1024 critic targets per update, with two seeds per setting (we use 512 critic targets everywhere else in the paper). Here $N$ sets both the number of candidates in the gate and the number of best-of-$N$ samples used for acting, so the panel varies the two together.