Long-Horizon Q-Learning: Accurate Value Learning via n-Step Inequalities

Armaan A. Abraham **Lucy Xiaoyang Shi** **Chelsea Finn**
Stanford University
armaana@stanford.edu

Abstract

Off-policy, value-based reinforcement learning methods such as Q-learning are appealing because they can learn from arbitrary experience, including data collected by older policies or other agents. In practice, however, bootstrapping makes long-horizon learning brittle: estimation errors at later states propagate backward through temporal-difference (TD) updates and can compound over time. We propose *long-horizon Q-learning* (LQL), which introduces a principled backstop against compounding error when learning the optimal action-value function. LQL builds on a prior *optimality tightening* observation: any realized action sequence lower-bounds what the optimal policy can achieve in expectation, so acting optimally earlier should not be worse than following the observed actions for several steps before switching to optimal behavior. Our contribution is to turn this inequality into a practical stabilization mechanism for Q-learning by using a hinge loss to penalize violations of these bounds. Importantly, LQL computes these penalties using network outputs already produced for the TD error, requiring no auxiliary networks and no additional forward passes relative to Q-learning. When combined with multiple state-of-the-art methods on a range of online and offline-to-online benchmarks, LQL consistently outperforms both 1-step TD and n-step TD learning at similar runtime. Code is available at `https://github.com/armaan-abraham/lql`.

1 Introduction

Off-policy reinforcement learning holds the promise of turning past experience into future competence: by learning a value function, an agent can improve from data collected by older policies, other agents, or imperfect exploration, without requiring fresh on-policy rollouts at every update [Watkins and Dayan, 1992, Sutton and Barto, 1998]. This promise is particularly compelling in domains where interaction is expensive and long-horizon behavior matters (e.g., robotics), where we would like to extract as much learning signal as possible from each transition. Classic value-based methods such as Q-learning do exactly this by *bootstrapping*: they train a Q-function so that the utility of a state-action pair agrees with the immediate reward plus

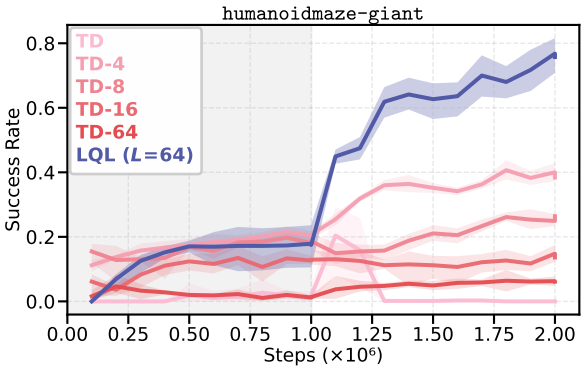


Figure 1: **LQL with long trajectories scales to the longest task in OGBench; n-step TD degrades as n grows.** Sparse-reward `humanoidmaze-giant` (all tasks) with Best-of-N policies. TD-n at increasing n and LQL with trajectory length 64.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

a discounted estimate of what the learned policy could achieve next, i.e., a counterfactual continuation under the policy being learned.

However, bootstrapping over long horizons is fragile in practice because estimation errors can amplify as they propagate backward through time [Jaakkola et al., 1993, Sutton and Barto, 1998, Asis et al., 2018, Park et al., 2025b]. A common mitigation is to reduce reliance on the next-state estimate by using multi-step targets (e.g., n-step or λ-returns), replacing a 1-step backup with a longer segment of observed rewards before switching back to a bootstrapped value estimate [Sutton and Barto, 1998, Hessel et al., 2017, Asis et al., 2018]. Yet this remedy introduces its own tension in off-policy settings: an n-step target for $Q(s_t, a_t)$ necessarily incorporates the rewards generated by the *logged* actions $a_{t+1}, \dots, a_{t+n-1}$ that followed in the replayed trajectory. As a result, even if a_t is a high-quality decision, the resulting backup can be pessimistic when the subsequent recorded actions are low-quality. Put differently, multi-step returns partially evaluate (s_t, a_t) under the incorrect assumption that the agent will continue to follow the behavior that produced the trajectory for several more steps, rather than switching immediately to the (improving) policy being learned.

This paper asks: can we keep the simplicity and low-variance of 1-step off-policy TD learning, while introducing a principled long-horizon backstop that detects and corrects the kinds of inconsistencies that lead to compounding error? Our starting point is a standard consequence of optimality—and one previously leveraged for "optimality tightening"—that any realized sequence of actions provides a lower bound on what the optimal policy could achieve in expectation from the same starting state [He et al., 2016]. Concretely, for any trajectory of experience, the optimal value of the initial decision cannot be *worse* (in expected return) than committing to the observed actions for a while and only then behaving optimally. We refer to this constraint as an *optimality inequality* over trajectories.

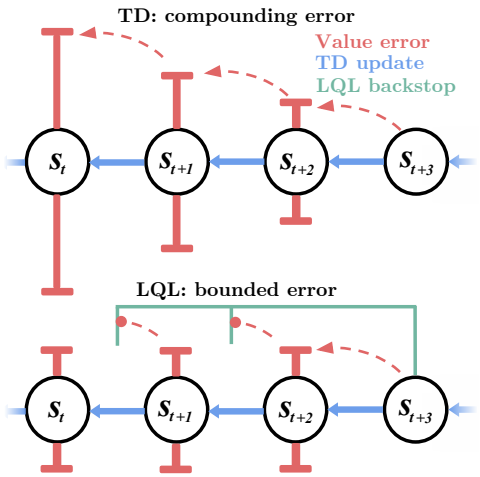


Figure 2: **LQL establishes a backstop against compounding TD error over time.** Standard 1-step TD can amplify estimation errors as they propagate backward through bootstrap updates (top). LQL's long-horizon constraints provide additional correction signals that bound these inconsistencies across multiple steps (bottom).

Motivated by this inequality, we introduce a training objective that augments the standard TD error with *hinge* penalty terms that enforce long-horizon consistency asymmetrically. Intuitively, if a value estimate at some state-action is *below* the return realized by an actual action sequence, the estimate is pushed upward; conversely, if combining a later value estimate with the observed actions leading up to it would imply doing *better* than acting optimally earlier, the later estimate is pushed downward. These hinge losses provide a trajectory-level correction signal, but can be computed from the same Q-values already used in standard TD updates.

Our main contribution is *long-horizon Q-learning* (LQL), a general off-policy value learning approach that is agnostic to the policy extraction mechanism. LQL samples trajectories from the replay buffer and constructs hinge penalties using only network outputs already computed within a typical TD update (e.g., $Q(s_t, a_t)$ and the bootstrapping value at the next state), requiring no auxiliary networks and no additional forward passes per update compared to Q-learning. Empirically, across locomotion and robot manipulation tasks from OGBench [Park et al., 2025a] and RoboMimic [Mandlekar et al., 2021], and across multiple ways of extracting a policy from the learned value function, LQL improves over TD learning while incurring minor additional runtime cost per training iteration.

2 Related work

Off-policy learning. Off-policy value-based RL (e.g., Q-learning) can in principle learn from arbitrary experience [Watkins and Dayan, 1992, Sutton and Barto, 1998, Mnih et al., 2015], but is

often unstable in practice. This is commonly attributed to the *deadly triad*—off-policy learning, bootstrapping, and function approximation—which causes TD errors to propagate and amplify over long horizons [Baird, 1995, Tsitsiklis and Van Roy, 1996, Sutton and Barto, 1998, van Hasselt et al., 2018, Park et al., 2025b]. LQL addresses this issue by augmenting TD learning with a temporally extended *consistency backstop* that penalizes value inconsistencies across temporally distant states.

Multi-step backups. Multi-step backups, such as n-step returns and λ-returns, accelerate reward propagation and reduce sensitivity to compounding 1-step TD error [Sutton, 1988, Watkins, 1989, Peng and Williams, 1996, Sutton and Barto, 1998, Mnih et al., 2016, Hessel et al., 2017, Asis et al., 2018, Schulman et al., 2018, Chebotar et al., 2023, Schwarzer et al., 2023, Daley et al., 2025]. However, they define targets from logged trajectories, coupling early actions to subsequent (possibly suboptimal) behavior. LQL instead uses trajectories to impose an *optimality inequality*, regularizing long-horizon consistency while retaining standard TD updates.

Off-policy corrections for multi-step learning. In off-policy settings, multi-step targets are biased under policy mismatch [Sutton and Barto, 1998]. Importance sampling provides principled corrections [Precup et al., 2000] but suffers from high variance at long horizons [Hernandez-Garcia and Sutton, 2019], motivating truncated variants such as Retrace and V-trace [Munos et al., 2016, Espeholt et al., 2018]. Many such methods require behavior and target action likelihoods [Sutton and Barto, 1998], which is inconvenient for expressive generative policies (e.g., diffusion or flow-matching) increasingly used in robotics [Ho et al., 2020, Song et al., 2021, Chi et al., 2023, Ren et al., 2024, Wagenmaker et al., 2025, Park et al., 2025c, Black et al., 2025, Ai et al., 2026]. LQL is complementary: it derives hinge losses from an inequality characterization of the optimal $Q^\star$ and uses only existing TD network outputs, without action likelihoods or additional forward passes.

Optimality tightening. LQL builds on the classical characterization of optimal value functions via Bellman (in)equalities [Bellman, 1958, Puterman, 1994, Sutton and Barto, 1998]. A closely related method, *optimality tightening* [He et al., 2016], also derives hinge losses from optimality inequalities. LQL differs by formulating the inequality using policy-generated actions, allowing reuse of the same network outputs as the TD loss and avoiding auxiliary networks and extra forward passes. In contrast, optimality tightening requires additional Q evaluations per update (at least $2\times$, and $4\times$ in He et al. [2016]), which can substantially reduce wall-clock efficiency [Hessel et al., 2017, Lee et al., 2019].

3 Preliminaries

We consider a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with discount $\gamma \in [0, 1)$. A (stochastic) policy $\pi(\cdot \mid s) \in \Delta(\mathcal{A})$ induces trajectories $(s_0, a_0, r_0, s_1, \ldots)$ with $r_t = R(s_t, a_t)$ and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The goal is to maximize the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. The action-value function for π is $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a]$, and the optimal action-value function Q^* satisfies the Bellman optimality equation

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \Big[R(s, a) + \gamma \max_{a'} Q^*(s', a') \Big]. \tag{1}$$

Q-learning and target networks. In the function approximation setting, Q-learning trains a network Q_θ by minimizing a temporal-difference (TD) loss on replayed transitions (s, a, r, s'):

$$\ell_{\mathrm{TD}}(\theta) = \big(Q_\theta(s, a) - \big[r + \gamma\, Q_{\bar{\theta}}(s', a^*(s')) \big] \big)^2, \tag{2}$$

where $Q_{\bar{\theta}}$ is a slowly-updated target network (e.g., soft updates $\bar{\theta} \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$ with $\tau \ll 1$), and $a^*(s')$ denotes the action used for bootstrapping. In discrete action spaces $a^*(s') = \arg\max_{a'} Q_{\bar{\theta}}(s', a')$; in continuous control we typically use a learned actor π_ϕ as a computational proxy for the maximizer, i.e., $a^*(s') = \pi_\phi(s')$.

Trajectory notation. We will work with *trajectories* of experience of length L drawn from the replay buffer, $(s_{t:t+L},\, a_{t:t+L-1},\, r_{t:t+L-1})$, where $s_{t:t+L}$ denotes $(s_t, s_{t+1}, \ldots, s_{t+L})$ and similarly for actions/rewards. For indices $i < j \le t + L$, define the discounted partial return along the observed trajectory as $G_{i:j} \triangleq \sum_{u=i}^{j-1} \gamma^{u-i} r_u$.

4 Long-horizon Q-learning

This section introduces *long-horizon Q-learning* (LQL). The goal is to preserve the strengths of standard TD learning—transition-level, counterfactual bootstrapping—while adding a *long-horizon backstop* that discourages the inconsistent value estimates responsible for compounding error. Concretely, we: (i) derive trajectory-wise optimality inequalities, (ii) interpret them as *soft constraints* on the learned value function over replay-buffer trajectories, and (iii) enforce these constraints using asymmetric (hinge) penalties that can be computed by *reusing the same network evaluations* already required for TD learning. We first present the constraint formulation, then derive the resulting penalty-based objective, and finally describe the practical sampled approximation and implementation details.

4.1 Optimality inequalities over trajectories

A consequence of optimality is that, from any state, acting optimally immediately cannot be worse (in expectation) than committing to an arbitrary sequence of actions for some duration and only then behaving optimally. One convenient form is: for any $i < j$,

$$Q^*(s_i, a_i) \;\geq\; \mathbb{E}\big[\, G_{i:j} + \gamma^{j-i} Q^*(s_j, a_j) \,\big]\,, \tag{3}$$

where the expectation is over the MDP dynamics (and any stochasticity in the action sequence, if applicable). We will use two variants that replace one side by an *optimal action* to match the computation available in TD learning (for $j > i$):

$$Q^*(s_i, a^*(s_i)) \geq \mathbb{E}\Big[G_{i:j} + \gamma^{j-i} Q^*(s_j, a_j)\Big], \tag{4}$$

$$Q^*(s_i, a_i) \geq \mathbb{E}\Big[G_{i:j} + \gamma^{j-i} Q^*(s_j, a^*(s_j))\Big]. \tag{5}$$

When $j = i + 1$, Equation (5) reduces to the Bellman optimality equation (1).

4.2 From optimality constraints to optimization objective

The inequalities in Equations (4)–(5) specify desirable properties of Q^* over temporally distant states: the value of acting optimally earlier should upper bound the value implied by deferring optimality until later, and conversely, observed actions followed by later optimal behavior should lower bound what can be achieved when acting optimally earlier. A natural way to encode such inequalities in learning is via a constrained problem $\min_\theta \; \mathbb{E}[\ell_{\mathrm{TD}}(\theta)]$ subject to Equations (4) and (5). Rather than enforcing hard constraints, we use a standard penalty/Lagrangian-style relaxation: each inequality violation contributes a nonnegative penalty, yielding an unconstrained objective consisting of the TD loss plus weighted constraint penalties. Concretely, for each replayed trajectory we will create two families of penalties: **Lower-bound (LB) penalties** that push *up* $Q_\theta(s_k, a_k)$ when it falls below a return realized by rolling forward along the trajectory and then bootstrapping with an (approximate) optimal action at a later state. **Upper-bound (UB) penalties** that push *down* $Q_\theta(s_k, a_k)$ when combining it with preceding observed rewards would imply outperforming an optimal action taken earlier.

4.3 Two-sided hinge penalties

Consider a replay trajectory $(s_{t:t+L}, a_{t:t+L-1}, r_{t:t+L-1})$. For brevity, re-index positions within the trajectory as follows: for $k \in \{0, \ldots, L-1\}$, $(s_k, a_k, r_k) \triangleq (s_{t+k}, a_{t+k}, r_{t+k})$, and define $s_L \triangleq s_{t+L}$. Let $a_k^* \triangleq a^*(s_k)$ denote the bootstrap action at state s_k (argmax in discrete actions, or $\pi_\phi(s_k)$ in continuous control). We use $G_{k:\ell}$ as defined above.

Lower-bound penalties. Using Equation (5), for any $\ell > k$ we would like $Q(s_k, a_k) \gtrsim G_{k:\ell} + \gamma^{\ell-k} Q(s_\ell, a_\ell^*)$. Formally, Equation (5) holds *in expectation* over trajectories (and any randomness in the environment and policy). In practice, we approximate this expectation using a single sampled trajectory, i.e., we treat $G_{k:\ell}$ and (s_ℓ, a_ℓ^*) from the replayed trajectory as a Monte Carlo estimate of the right-hand side.

We implement a soft version with a hinge-squared penalty where the bootstrap side uses the target network:

$$\delta_{\mathrm{LB}}(k,\ell;\theta) \;=\; \Big[G_{k:\ell} + \gamma^{\ell-k} Q_{\bar{\theta}}(s_\ell, a_\ell^*) - \; Q_\theta(s_k, a_k)\Big]_+^2. \tag{6}$$

Intuitively, if the observed rewards plus a (target) bootstrap at time ℓ exceed the current estimate at time k, we push $Q_\theta(s_k, a_k)$ upward. We use a hinge-squared form: the hinge makes the constraint one-sided (we only penalize violations), while squaring yields smoother gradients when violations occur. We aggregate over a set of future indices $\mathcal{F}(k) \subseteq \{k+1, \ldots, L\}$:

$$\ell_{\mathrm{LB}}(k;\theta) = \frac{1}{|\mathcal{F}(k)|} \sum_{\ell \in \mathcal{F}(k)} \delta_{\mathrm{LB}}(k,\ell;\theta). \tag{7}$$

A practical choice is to exclude $\ell = k+1$ since that case largely overlaps with the 1-step TD target; e.g., $\mathcal{F}(k) = \{k+2, \ldots, L\}$, and $\ell_{\mathrm{LB}}(k;\theta) = 0$ if $\mathcal{F}(k)$ is empty.

Upper-bound penalties. Using Equation (4), for any $i \le k$ we would like $Q(s_i, a_i^*) \gtrsim G_{i:k} + \gamma^{k-i} Q(s_k, a_k)$. Violations correspond to $Q(s_k, a_k)$ being too large relative to what could be achieved by acting optimally earlier, as measured by $Q(s_i, a_i^*)$. We penalize such violations with

$$\delta_{\mathrm{UB}}(i,k;\theta) \;=\; \Big[G_{i:k} + \gamma^{k-i} Q_\theta(s_k, a_k) - \; Q_{\bar{\theta}}(s_i, a_i^*)\Big]_+^2, \tag{8}$$

where the optimal-action value on the right is evaluated by the target network and treated as a stable reference. Importantly, the special case $i = k$ yields a same-state upper bound: since $G_{k:k} = 0$ and $\gamma^0 = 1$, the constraint reduces to $Q_{\bar{\theta}}(s_k, a_k^*) \ge Q_\theta(s_k, a_k)$, providing direct downward pressure when a suboptimal action is valued above the (approximate) greedy action at the same state. We aggregate upper-bound penalties for each k by averaging over indices i for which the target-network quantity $Q_{\bar{\theta}}(s_i, a_i^*)$ is already available from the TD computation on the same replay chunk. Concretely, since TD targets evaluate $Q_{\bar{\theta}}$ on next-states, we have target evaluations for states up to and including s_k (via the immediately preceding transition), so we use $\mathcal{P}(k) \subseteq \{1, \ldots, k\}$, which includes the same-state upper bound term $i = k$ without requiring additional forward passes. We then define

$$\ell_{\mathrm{UB}}(k;\theta) = \frac{1}{|\mathcal{P}(k)|} \sum_{i \in \mathcal{P}(k)} \delta_{\mathrm{UB}}(i,k;\theta), \tag{9}$$

and set $\ell_{\mathrm{UB}}(k;\theta) = 0$ if $\mathcal{P}(k)$ is empty (e.g., $k = 0$). In contrast to our formulation of the upper bounds, He et al. [2016] formulates them using logged actions rather than actions from the learned policy at the earlier states, which precludes both the reuse of target-network outputs from earlier in the trajectory and the same-state upper bound. This difference in formulation allows us to use no additional forward passes in computing the hinge penalties.

4.4 Final training objective

For each transition index $k \in \{0, \ldots, L-1\}$ in the trajectory, let $\ell_{\mathrm{TD}}(k;\theta)$ denote the 1-step TD loss (2) computed on (s_k, a_k, r_k, s_{k+1}). LQL augments TD learning with the long-horizon penalties above:

$$\ell_{\mathrm{LQL}}(k;\theta) = \ell_{\mathrm{TD}}(k;\theta) + \lambda_{\mathrm{UB}}\, \ell_{\mathrm{UB}}(k;\theta) + \lambda_{\mathrm{LB}}\, \ell_{\mathrm{LB}}(k;\theta), \tag{10}$$

with nonnegative weights $\lambda_{\mathrm{UB}}, \lambda_{\mathrm{LB}}$. *In this work, we do not tune these weights for the main baseline comparison experiments (e.g., Figure 3) and use* $\lambda_{\mathrm{UB}} = \lambda_{\mathrm{LB}} = 1$; we only vary them in the ablations in Figures 11 and 10 and Table 10. We use the descriptor "long-horizon" in naming long-horizon Q-learning due to the fact that the hinge penalties $\ell_{\mathrm{UB}}, \ell_{\mathrm{LB}}$ relate value estimates across temporal distances greater than one, in contrast to ℓ_{TD}. The overall loss for a sampled trajectory is the average over k: $\mathcal{L}_{\mathrm{LQL}} = \frac{1}{L} \sum_{k=0}^{L-1} \ell_{\mathrm{LQL}}(k;\theta)$.

4.5 Practical notes

The inequalities in Equation (4)–(5) are stated in expectation over dynamics. In LQL, we approximate these expectations using a sampled replay trajectory: the observed rewards within the trajectory provide an unbiased sample of the return prefix, while the remaining tail is approximated by a

Algorithm 1 LQL update on a replay trajectory

Input: replay buffer $\mathcal{D}$; online Q_θ; target $Q_{\bar{\theta}}$; bootstrap policy/actor π_ϕ (or argmax); step size η
Sample a trajectory $(s_{0:L}, a_{0:L-1}, r_{0:L-1}) \sim \mathcal{D}$
Compute bootstrap actions $a_k^* \leftarrow \pi_\phi(s_k)$ for $k = 0, \dots, L$ (or $a_k^* = \arg\max_a Q_{\bar{\theta}}(s_k, a)$)
Evaluate $Q_\theta(s_k, a_k)$ for $k = 0, \dots, L-1$
Evaluate $Q_{\bar{\theta}}(s_k, a_k^*)$ for $k = 0, \dots, L$
$\mathcal{L}_{\mathrm{LQL}} = \frac{1}{L} \sum_{k=0}^{L-1} \Big(\ell_{\mathrm{TD}}(k;\theta) + \lambda_{\mathrm{UB}} \ell_{\mathrm{UB}}(k;\theta) + \lambda_{\mathrm{LB}} \ell_{\mathrm{LB}}(k;\theta) \Big)$
$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\mathrm{LQL}}$
Update target network (e.g., $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$)

bootstrap term $Q_{\bar{\theta}}(s_\ell, a_\ell^*)$. Under deterministic transition dynamics, this sample-based approximation is unbiased; under stochastic dynamics, it introduces bias, which we explore further in Section 5.5 (Figure 9), Section 6, and Appendix C.1. Crucially, the additional penalties in Equations (6) and (8) can be computed using the same forward passes already required by TD learning on the trajectory:

- We evaluate $Q_\theta(s_k, a_k)$ for all observed state-action pairs in the trajectory (needed for ℓ_{TD}).
- We evaluate $Q_{\bar{\theta}}(s_k, a_k^*)$ for bootstrap actions at the corresponding states (needed for TD targets).

No auxiliary networks are introduced, and we do not require additional forward passes beyond those used to compute TD losses along the trajectory. In implementation, we stop gradients through all target-network quantities $Q_{\bar{\theta}}(\cdot, \cdot)$, and treat a_k^* as a bootstrap action (computed via argmax or sampling from the actor) exactly as in standard off-policy TD learning. Finally, we compute the discounted partial returns $G_{i:j}$ efficiently using prefix sums along the trajectory, enabling evaluation of all selected pairs (i, k) and (k, ℓ) with negligible additional overhead.

Compatibility with existing value learning methods. We derive the above results by considering the augmentation of 1-step TD learning, in which case we establish hinge penalties that apply across transitions with a minimum separation of one time step—ultimately, for simplicity. However, these hinge penalties can also be applied to n-step TD learning and action chunking [Li et al., 2025], where the hinge penalties would be established over a minimum separation of n (or the action chunk size) time steps; we also experimentally evaluate LQL on action chunking below.

5 Experiments

We test when and why LQL helps:

1. Across diverse offline-to-online tasks and policy classes, how does LQL compare to 1-step TD, length-matched n-step TD, and the closest prior optimality-inequality method, OT [He et al., 2016]?
2. Does trajectory length offer a useful scaling axis, particularly where TD batch-size scaling is known to fail [Fu et al., 2025]?
3. Does LQL retain its benefit under stochastic dynamics, where the in-expectation inequalities introduce a theoretical bias?
4. Are gains attributable to the hinge penalties themselves rather than to side effects of trajectory sampling, and do they leave a measurable signature on value-function stability?

We apply LQL to two of its three compatible settings—1-step TD and action chunking—leaving the n-step TD case to future work.

5.1 Environments and datasets

We evaluate on robot manipulation and navigation tasks from OGBench [Park et al., 2025a] and RoboMimic [Mandlekar et al., 2021], which have several favorable properties for this evaluation (more details in Appendix A.2): **Long horizons:** In OGBench's `humanoidmaze-giant`, a 21-DoF humanoid must traverse mazes that take thousands of environment steps to solve, with a single sparse reward upon reaching the goal. 1-step TD must therefore propagate signal across an extreme bootstrap chain, which is the failure mode LQL targets. **Behavioral heterogeneity:** In RoboMimic, we use

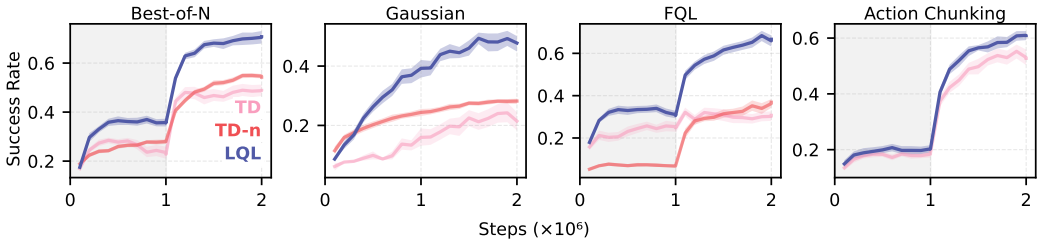


Figure 3: **Across policy-extraction families, LQL achieves higher average success than TD and TD-n.** Mean success rate averaged over all environments, separated by policy type. Background shading indicates whether the update includes an environment interaction step (white) or is purely offline (gray).

multi-human datasets in which trajectories were collected by operators of varying proficiency. This is a regime where n-step targets in principle suffer from the contribution of low-quality logged actions. **Stitching dependence:** In OGBench, `humanoidmaze` and `cube-triple` use the `navigate`-style and `play`-style datasets respectively, which are collected by exploratory rather than task-directed policies. This makes stitching together trajectories from these exploratory policies important, which is also a known theoretical limitation of TD-n [Precup et al., 2000, Munos et al., 2016]. We use two standard protocols (Appendix A.6): RLPD-style continual mixing for the Gaussian actor, and two-stage offline→online training for the others. Behavior-regularization coefficients (e.g., α in FQL/QC-FQL) are identical for LQL, TD, and TD-n (Table 3).

5.2 Comparison methods and protocol

To isolate the effect of the value-learning rule, we train Q-functions using **(a) 1-step TD**, **(b) length-matched n-step TD** ($n = L = 8$), and **(c) LQL**, then apply identical policy extraction mechanisms to each learned value function. We treat 1-step TD as a baseline to isolate the contribution of LQL's backstop beyond standard bootstrapped learning and length-matched n-step TD as an existing recipe for incorporating multi-step information into off-policy value learning. For breadth, we also report **IQL** [Kostrikov et al., 2021] and **ReBRAC** [Tarasov et al., 2023]. We compare directly to optimality tightening [He et al., 2016] in Section 5.5.

Compute parity. For all critic-learning rules we match the total number of Q-network forward passes per gradient step. With a batch size of 1024 transitions and LQL trajectory length 8, LQL samples 128 starting indices uniformly from the replay buffer and unrolls length-8 segments; TD and n-step TD use the same effective number of transitions per step. More details are in Appendix A.3.

Policy extraction families. To test that improvements from the critic carry across actor parameterizations, we evaluate four policy extraction mechanisms representative of contemporary offline-to-online RL: a **Gaussian actor** (RLPD-style [Ball et al., 2023]), **Best-of-N (BFN)** sampling from a flow-matching behavior policy [Stiennon et al., 2022], **Flow Q-learning (FQL)** [Park et al., 2025c], and an **action-chunked policy** via the QC-FQL recipe of Li et al. [2025]. Action chunking is particularly important because it underlies many state-of-the-art real-world robot policies [Zhao et al., 2023, Amin et al., 2025].

5.3 Comparative evaluation across tasks and policy classes

Per-task and per-suite results appear in Tables 4 and 6; Best-of-N aggregates are in Figure 4, and the cross-policy aggregation is in Figure 3. Across all four policy extraction mechanisms, LQL improves average success rate over both 1-step TD and length-matched n-step TD. LQL's advantage over n-step TD indicates that its improvement is not just from using n-step returns. Length-matched n-step TD often provides only a modest improvement over 1-step TD, and on several tasks (e.g., `cube-triple` with FQL) it performs worse, consistent with the well-known off-policy bias of n-step targets [Precup et al., 2000, Munos et al., 2016]. LQL's more favorable usage of the multi-step target is most visible on tasks (`humanoidmaze-md`, `cube-triple`) where success is most reliant on stitching together segments of independently suboptimal offline experience.

Compute overhead. These gains come with a small per-update slowdown of 4.7% on average across policy classes (Table 8, Appendix B.8), arising from computing pairwise discounted returns within

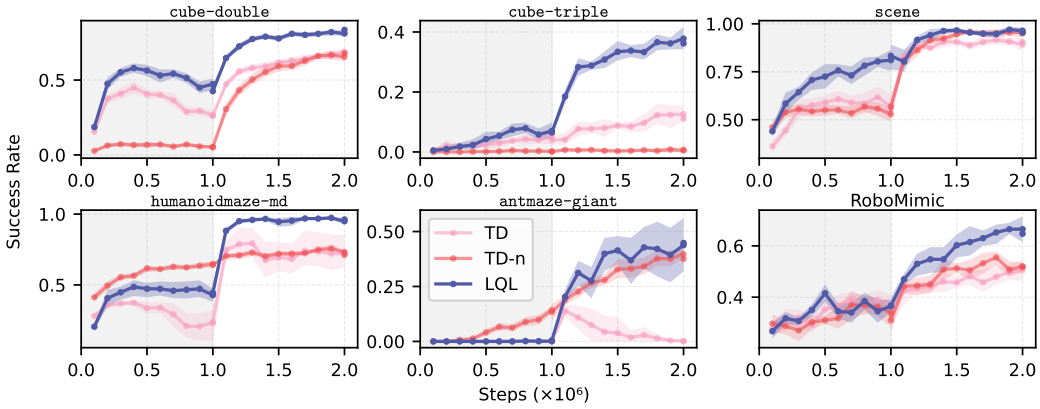


Figure 4: **For Best-of-N policies, LQL improves over both 1-step TD and TD-n across task groups.** Each panel aggregates success rates within a task group over training.

each sampled trajectory. This overhead does not scale with critic or actor size, so its relative cost shrinks as networks grow. We also observe negligible costs for $L = 64$ (Appendix B.8).

5.4 Trajectory length as an axis for scaling

A practical concern with deep value-based RL is that, unlike supervised learning, increasing the batch size for TD does not reliably improve performance and frequently degrades it past a threshold [Fu et al., 2025] (we reproduce this in Figure 13). This is hypothesized to reflect overfitting to inaccurate target estimates. We investigate whether LQL's trajectory length offers an alternative scaling axis that does not suffer this pathology. Figure 5 sweeps trajectory length L at fixed number of trajectories per batch (so larger L means more transitions per update). In all cases, LQL's final success rate improves with L over the swept range. This contrasts with the corresponding TD batch-size sweep, where additional transitions yield no improvement or hurt performance (Figure 13).

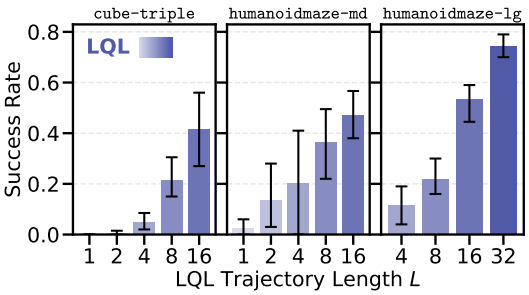


Figure 5: **LQL performance improves as trajectory length L grows.** Each panel sweeps L at fixed trajectories per batch, so larger L means strictly more compute per step. *Left, middle:* FQL actor, 128 trajectories per batch, on `task3` and `task4` respectively. *Right:* Best-of-N actor, 64 trajectories per batch, on `task3`. Network sizes are held fixed. Increasing L raises final success rate in all three settings.

We stress-test this in the longest task in OGBench, sparse-reward `humanoidmaze-giant`, with $L = 64$ (Figure 1, Table 5). Here 1-step TD never solves a single task (0% across all five tasks). n-step TD partially helps but plateaus at $n = 4$ (38.4% averaged) and *degrades* as n grows further, with $n = 64$ achieving only 6.1%. LQL with $L = 64$ achieves 75.7%, saturating two tasks (97.3% and 98.7%). We read this as direct evidence that the hinge constraints, unlike multi-step TD targets, can absorb information from very long trajectories without inheriting the off-policy bias that causes n-step TD to deteriorate at large n.

5.5 Further experiments

Mechanism: hinge penalties vs. trajectory sampling. LQL differs from 1-step TD in two ways simultaneously: it samples short trajectories rather than independent transitions, and it adds hinge penalties on top of the TD loss. To attribute the gains correctly, we ablate $\lambda_{\text{LB}} = \lambda_{\text{UB}} = 0$, which preserves the trajectory sampling but removes the long-horizon constraints, leaving 1-step TD trained on trajectory-sampled minibatches (Figure 11, Appendix B.4). Removing the hinge penalties substantially reduces performance, verifying their importance. Interestingly, we see that trajectory sampling sometimes outperforms standard transition-level TD, which we discuss further in Appendix B.4.

Robustness to environment stochasticity. We next ask whether the theoretical bias induced by environmental stochasticity materially harms practical performance. We construct stochastic versions of OGBench tasks by adding zero-mean Gaussian noise with standard deviation σ to actions before execution and recollecting matched offline datasets at each σ (Appendix B.2). Across all measured σ, LQL matches or outperforms 1-step TD (Figure 9).

Sensitivity to hinge coefficient. All comparative results above use $\lambda_{\text{LB}} = \lambda_{\text{UB}} = 1$, chosen as a simple default rather than tuned per task. Figure 10 sweeps these coefficients while holding all other hyperparameters fixed. Performance is broadly stable across roughly an order of magnitude around the default.

Q-value stability. We also analyzed the learned value function for TD and LQL directly. Figure 6 plots the average online Q-value during training on `humanoidmaze-giant`, where the reward is in $\{-1, 0\}$, so Q^* must be non-positive everywhere. For 1-step TD, 14/15 training runs blow up, with the average Q crossing zero and growing to magnitudes exceeding 900. With LQL, Q-values remain in the analytically valid range across all tasks. This is direct evidence of the hinge upper-bound penalty preventing the common runaway overestimation of TD.

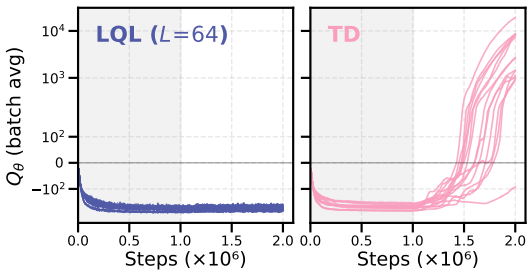


Figure 6: **LQL keeps Q-values within the analytically valid range; 1-step TD diverges.** Average online $Q_\theta(s, a)$ during training on `humanoidmaze-giant` (rewards in $\{-1, 0\}$), one curve per task/seed. Since rewards are non-positive, $Q^* \leq 0$ everywhere.

Comparison to optimality tightening. LQL's most similar prior work, OT [He et al., 2016], differs from our implementation as described in Sections 2 and 4. We adapt the public OT codebase to run on our evaluations and show in Appendix B.1 that LQL outperforms OT, supporting the design choices that distinguish the two methods.

6 Conclusion

We introduced *long-horizon Q-learning* (LQL), a practical algorithm for mitigating compounding TD error in off-policy value learning. Starting from the optimality inequality—a principled relationship that constrains value estimates separated by any time horizon—we derived two-sided hinge penalties that reuse the network outputs already produced by standard TD updates, requiring no auxiliary networks and no additional forward passes. Across locomotion and manipulation tasks from OGBench and RoboMimic and across four policy-extraction families, LQL consistently improves over both 1-step TD and n-step TD at a small runtime cost. Most strikingly, on the sparse-reward `humanoidmaze-giant`—the longest task in OGBench—LQL achieves 75.7% success across the five tasks while 1-step TD fails to solve any of them and n-step TD plateaus at n=4 and degrades thereafter. These results carry two broader implications: LQL provides a path for absorbing information from very long trajectories without inheriting the off-policy bias that limits n-step targets [Precup et al., 2000, Munos et al., 2016], and trajectory length emerges as a scaling axis that does not exhibit the pathological degradation of batch scaling in TD [Fu et al., 2025].

The main limitation of this work is theoretical: the optimality inequality is an *in-expectation* statement, so per-sample penalties introduce a bias in stochastic environments. We give a horizon-independent bound on this bias in Appendix C.1; the bound tightens as the suboptimality of the behavior data grows, and Figure 9 shows LQL matching or outperforming 1-step TD across all noise levels we tested. Several directions follow naturally: applying the same hinge framework on top of n-step TD, which could combine LQL's stability with n-step TD's faster reward propagation; a more mechanistic study of how the upper- and lower-bound terms each contribute to the Q-value stability we observe in Figure 6; and pushing trajectory length beyond $L = 64$, where our monotonic sweep suggests substantial headroom remains.

Acknowledgments and Disclosure of Funding

This work was supported by the Robotics and AI Institute (RAI) and ONR grant N00014-22-1-2621. We would also like to thank Lars Ankile for helpful discussions about this work.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice, 2021. URL https://arxiv.org/abs/2108.13264.

Xinyue Ai, Yutong He, Albert Gu, Ruslan Salakhutdinov, J. Zico Kolter, Nicholas Matthew Boffi, and Max Simchowitz. Joint Distillation for Fast Likelihood Evaluation and Sampling in Flow-based Models, January 2026. URL http://arxiv.org/abs/2512.02636. arXiv:2512.02636 [cs].

Ali Amin, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^{*}_{0.6}$: a VLA That Learns From Experience, November 2025. URL http://arxiv.org/abs/2511.14759. arXiv:2511.14759 [cs].

Kristopher De Asis, J. Fernando Hernandez-Garcia, G. Zacharias Holland, and Richard S. Sutton. Multi-step Reinforcement Learning: A Unifying Algorithm, June 2018. URL http://arxiv.org/abs/1703.01327. arXiv:1703.01327 [cs].

Leemon C. Baird. Residual algorithms: reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, page 30–37, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603778.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data, May 2023. URL http://arxiv.org/abs/2302.02948. arXiv:2302.02948 [cs].

Richard Bellman. Dynamic programming and stochastic control processes. *Information and Control*, 1(3):228–239, 1958. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(58)80003-0. URL https://www.sciencedirect.com/science/article/pii/S0019995858800030.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization, April 2025. URL http://arxiv.org/abs/2504.16054. arXiv:2504.16054 [cs].

F. P. Cantelli. Sui confini della probabilita. In *Atti del Congresso Internazional del Matematici*, 1928.

Yevgen Chebotar, Quan Vuong, Alex Irpan, Karol Hausman, Fei Xia, Yao Lu, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Sontakke, Grecia Salazar, Huong T. Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singht, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn, and Sergey Levine. Q-Transformer: Scalable Offline Reinforcement Learning via Autoregressive Q-Functions, October 2023. URL http://arxiv.org/abs/2309.10150. arXiv:2309.10150 [cs].

Cheng Chi, S. Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, B. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Robotics: Science and Systems*, 2023. doi: 10.1177/02783649241273668.

Brett Daley, Martha White, and Marlos C. Machado. Averaging n-step Returns Reduces Variance in Reinforcement Learning, December 2025. URL http://arxiv.org/abs/2402.03903. arXiv:2402.03903 [cs].

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad 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 Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Preston Fu, Oleh Rybkin, Zhiyuan Zhou, Michal Nauman, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Compute-Optimal Scaling for Value-Based Deep RL, August 2025. URL http://arxiv.org/abs/2508.14881. arXiv:2508.14881 [cs].

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Frank S. He, Yang Liu, Alexander G. Schwing, and Jian Peng. Learning to Play in a Day: Faster Deep Reinforcement Learning by Optimality Tightening, November 2016. URL http://arxiv.org/abs/1611.01606. arXiv:1611.01606 [cs].

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), June 2023. URL http://arxiv.org/abs/1606.08415. arXiv:1606.08415 [cs].

J. Fernando Hernandez-Garcia and Richard S. Sutton. Understanding Multi-Step Deep Reinforcement Learning: A Systematic Study of the DQN Target, 2019. URL https://arxiv.org/abs/1901.07510.

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, October 2017. URL http://arxiv.org/abs/1710.02298. arXiv:1710.02298 [cs].

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Tommi Jaakkola, Michael Jordan, and Satinder Singh. Convergence of Stochastic Iterative Dynamic Programming Algorithms. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper_files/paper/1993/file/5807a685d1a9ab3b599035bc566ce2b9-Paper.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning, October 2021. URL http://arxiv.org/abs/2110.06169. arXiv:2110.06169 [cs].

Su Young Lee, Sungik Choi, and Sae-Young Chung. Sample-Efficient Deep Reinforcement Learning via Episodic Backward Update, November 2019. URL http://arxiv.org/abs/1805.12375. arXiv:1805.12375 [cs].

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement Learning with Action Chunking, October 2025. URL http://arxiv.org/abs/2507.07969. arXiv:2507.07969 [cs].

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

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, February 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning, June 2016. URL http://arxiv.org/abs/1602.01783. arXiv:1602.01783 [cs].

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and Efficient Off-Policy Reinforcement Learning, November 2016. URL http://arxiv.org/abs/1606.02647. arXiv:1606.02647 [cs].

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL, February 2025a. URL http://arxiv.org/abs/2410.20092. arXiv:2410.20092 [cs].

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon Reduction Makes RL Scalable, October 2025b. URL http://arxiv.org/abs/2506.04168. arXiv:2506.04168 [cs].

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-Learning, May 2025c. URL http://arxiv.org/abs/2502.02538. arXiv:2502.02538 [cs].

Jing Peng and Ronald J. Williams. Incremental multi-step q-learning. *Mach. Learn.*, 22(1–3): 283–290, January 1996. ISSN 0885-6125. doi: 10.1007/BF00114731. URL https://doi.org/10.1007/BF00114731.

Doina Precup, Richard Sutton, and Satinder Singh. Eligibility Traces for Off-Policy Policy Evaluation. *Computer Science Department Faculty Publication Series*, June 2000.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL https://doi.org/10.1002/9780470316887.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization, December 2024. URL http://arxiv.org/abs/2409.00588. arXiv:2409.00588 [cs].

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation, October 2018. URL http://arxiv.org/abs/1506.02438. arXiv:1506.02438 [cs].

Max Schwarzer, Johan Obando-Ceron, Aaron Courville, Marc Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, Better, Faster: Human-level Atari with human-level efficiency, November 2023. URL http://arxiv.org/abs/2305.19452. arXiv:2305.19452 [cs].

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021. URL http://arxiv.org/abs/2011.13456. arXiv:2011.13456 [cs].

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, February 2022. URL http://arxiv.org/abs/2009.01325. arXiv:2009.01325 [cs].

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44, August 1988. ISSN 0885-6125. doi: 10.1023/A:1022633531479. URL https://doi.org/10.1023/A:1022633531479.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the Minimalist Approach to Offline Reinforcement Learning, October 2023. URL http://arxiv.org/abs/2305.09836. arXiv:2305.09836 [cs].

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL https://proceedings.neurips.cc/paper_files/paper/1996/file/e00406144c1e7e35240afed70f34166a-Paper.pdf.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad, 2018. URL https://arxiv.org/abs/1812.02648.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering Your Diffusion Policy with Latent Space Reinforcement Learning, June 2025. URL http://arxiv.org/abs/2506.15799. arXiv:2506.15799 [cs].

Christopher Watkins. Learning from delayed rewards. 01 1989.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00992698. URL http://link.springer.com/10.1007/BF00992698.

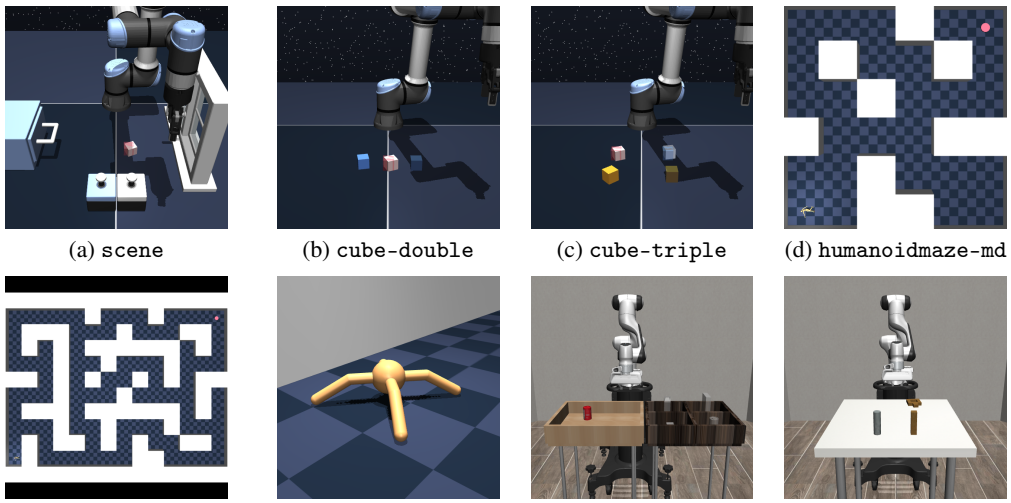
Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, April 2023. URL http://arxiv.org/abs/2304.13705. arXiv:2304.13705 [cs].

A Experiment Details

A.1 Evaluation Protocol

For all methods, we run 4 seeds for each `task` in each OGBench task group (e.g., `cube-double-play-singletask-task1-v0`) and 4 seeds for each task in RoboMimic (e.g., `can`). Throughout the paper, confidence intervals are 95% intervals from a curve-level bootstrap (1000 iterations) over training runs; for multi-task aggregates, we resample runs within each task following Agarwal et al. [2021].

A.2 Tasks



(a) `scene` (b) `cube-double` (c) `cube-triple` (d) `humanoidmaze-md`

(e) `humanoidmaze-giant` (f) `antmaze-giant` (g) `can` (h) `square`

Figure 7: Task panel for the OGBench [Park et al., 2025a] and RoboMimic [Mandlekar et al., 2021] domains we evaluate on. The `antmaze-giant` maze layout matches `humanoidmaze-giant`.

OGBench tasks. We use the single-task variants of OGBench, which fix an evaluation goal and relabel the dataset with a corresponding (semi-)sparse reward.

- `scene`: A robot arm interacts with a drawer, a window, a cube, and two button locks that gate the drawer and window. Each task requires a sequence of manipulations such as unlocking, opening, placing, and closing. We modify the reward to be sparse, similar to Li et al. [2025].
- `cube-double`, `cube-triple`: A robot arm rearranges 2 or 3 cubes to specified target configurations.
- `humanoidmaze-md`, `humanoidmaze-giant`: A 21-DoF Humanoid agent navigates a maze. The `giant` maze contains paths that can require thousands of environment steps to traverse. For `humanoidmaze-giant`, we use the 100M-transition offline dataset released by the original authors subsequent to the OGBench publication. Sparse reward.
- `antmaze-giant`: An 8-DoF quadrupedal Ant agent navigates the giant maze. Sparse reward.

RoboMimic tasks. We use the multi-human demonstration datasets, which contain trajectories collected by multiple human operators of varying proficiency.

- `can`: A robot arm picks a soda can and places it in a target bin.
- `square`: A robot arm picks a square nut and places it on a rod.

Table 1: Task characteristics. Episode length refers to the maximum number of environment steps per episode.

Task group	Dataset size (transitions)	Episode length	Action dimension
`scene`	1M	750	5
`cube-double`	1M	500	5
`cube-triple`	3M	1000	5
`humanoidmaze-md`	2M	2000	21
`humanoidmaze-giant`	100M	4000	21
`antmaze-giant`	1M	1000	8
`can`	62756	500	7
`square`	80731	500	7

A.3 Implementation details

Here, we outline a few more implementation details of LQL. The surrounding training infrastructure is described in Appendix A.6, so we focus here on implementation details following the receipt of a shape `(batch, trajectory)` array of transitions. For learning single-action policies, the array we receive contains 128 trajectories of 8 transitions. The loss in Equation 10 is then averaged over each trajectory, and an actor loss specific to each policy class is computed from all 128×8 observations and, for policies involving behavioral cloning, corresponding realized actions. For action chunking policies, we enforce our consistent length-8 trajectory at the action chunk level, meaning we sample `(batch, trajectory * chunk_size)=(128, 8*5)=(128, 40)` transitions. In the same way as non-LQL action chunk Q learning is performed, we then evaluate the value network with the first observation of each chunk and all of the actions in the chunk as input, which yields 8 value function evaluations per trajectory, equivalently to LQL applied to single-action value learning. All `(128, 8)` action chunks are then passed to an `actor_loss` function, which performs an FQL [Park et al., 2025c] actor update on each chunk.

A.4 Policy extraction details

The four policy extraction families described in Section 5.2 share the same critic learned by TD, TD-n, or LQL, but differ in how actions are selected from the value function. Here we summarize each method and note any deviations from the original recipes.

Best-of-N (BFN). We train a flow-matching behavior policy $\mu_\theta(s, z) : \mathcal{S} \times \mathbb{R}^{\dim(\mathcal{A})} \to \mathcal{A}$ with the standard flow-matching behavioral cloning loss, as in Park et al. [2025c], integrating the flow ODE with the Euler method using 10 steps. At action-selection time, we sample $N = 16$ noise vectors $z \sim \mathcal{N}(0, I)$, push each through the flow to obtain candidate actions, and choose the one that maximizes the learned Q [Stiennon et al., 2022]. No reparameterized policy gradient is taken through the flow, so the flow is trained only to match the data distribution and the Q-function performs all of the policy improvement.

Flow Q-learning (FQL). FQL [Park et al., 2025c] keeps the same BC flow μ_θ as above and additionally trains a separate one-step policy $\mu_\omega(s, z) : \mathcal{S} \times \mathbb{R}^{\dim(\mathcal{A})} \to \mathcal{A}$ that maps a noise sample directly to an action with a single network forward pass. The one-step policy is trained with

$$\mathcal{L}_\pi(\omega) = \mathbb{E}_{s,z}[-Q_\theta(s, \mu_\omega(s, z))] + \alpha\, \mathbb{E}_{s,z}\left[\|\mu_\omega(s, z) - \mu_\theta(s, z)\|_2^2\right],$$

where the second term distills the multi-step BC flow into the one-step policy. Park et al. [2025c] show that this distillation loss is an upper bound on the squared 2-Wasserstein distance between the two implied action distributions, so α acts as a behavioral cloning coefficient that controls how tightly the one-step policy stays near the BC flow; we use the values listed in Table 3. At deployment, action selection uses the one-step policy directly.

Gaussian (entropy-regularized). Our Gaussian actor follows the soft actor-critic (SAC) formulation [Haarnoja et al., 2018] combined with the offline-data design choices from RLPD [Ball et al., 2023]. The actor outputs a tanh-squashed diagonal Gaussian $\pi_\phi(\cdot \mid s)$ and is trained with the standard maximum-entropy objective:

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_s[-Q_\theta(s, a^\pi) + \alpha_{\text{ent}} \log \pi_\phi(a^\pi \mid s)],$$

where $a^\pi \sim \pi_\phi(\cdot \mid s)$ is sampled by the standard reparameterization trick. The temperature α_{ent} is automatically tuned against a target entropy of $-0.5 \dim(\mathcal{A})$ via the standard dual update on α_{ent}. From RLPD we adopt symmetric sampling, in which each minibatch is split equally between transitions sampled from the offline dataset and transitions sampled from the online replay buffer. We deviate from the original RLPD recipe in two places: we use 2 critics with LayerNorm rather than the 10-critic ensemble used in the original paper, and we omit the entropy backup in the critic target (Table 2).

Action-chunked (QC-FQL-style). For the action-chunked policy we follow the recipe of Li et al. [2025]. Both the critic and actor operate over action chunks $a_{t:t+h}$ of length $h = 5$: the critic is $Q_\theta(s_t, a_{t:t+h})$, and the actor uses the FQL parameterization above to produce the entire chunk in a single forward pass from a single noise sample. Because the critic conditions on the full executed action sequence, the corresponding h-step return is unbiased even when the data are off-policy [Li et al., 2025]. The actor update is identical to FQL applied at the chunk level, again with α from Table 3, and the LQL hinge penalties are imposed across action chunks rather than individual transitions, as described in Appendix A.3.

A.5 Hyperparameters

Table 2: Common hyperparameters.

Parameter	Value
Batch size (number of transitions)	1024
LQL trajectories per batch	128
LQL trajectory length	8
Discount factor	0.99
Optimizer	Adam [Kingma and Ba, 2017]
Learning rate	3×10^{-4}
Target network update rate	5×10^{-3}
Critic ensemble size	2
UTD Ratio	1
Number of flow steps	10
Number of samples in Best-of-N sampling	16
Number of training steps	2×10^6
Number of online training steps	2×10^6 for RLPD-based approaches, 1×10^6 otherwise
Network width	512
Network depth	4
Activation function	GELU [Hendrycks and Gimpel, 2023]
Actor layer norm	No
Critic layer norm	Yes
Entropy backup in RLPD	No
Action chunk size	5
LQL $\lambda_{\text{UB}}, \lambda_{\text{LB}}$	1 (except in Figures 11 and 10)
Target Q aggregation	OGBench: mean, RoboMimic: min

`humanoidmaze-giant`**-specific hyperparameters.** We use the 100M-transition navigate-style dataset released by the OGBench [Park et al., 2025a] authors, and a discount factor of 0.995. We use the same number of trajectories per batch as in the main experiments (Figure 4, 128), but with a longer trajectory length $L = 64$. Similar to the earlier experiments (Table 2), we match the batch size of each TD variant to LQL at the transition level: 8192 (128×64). Besides this, all other configuration is identical to Table 2, including network architectures and the use of Best-of-N actors with $N = 16$.

Table 3: α values by task group, used for both FQL and QC.

Task group	α
`humanoidmaze-md`	10
`antmaze-giant`	5
`cube-double`, `cube-triple`, `scene`	100
RoboMimic (`can`, `square`)	250

For ReBRAC and IQL, we reuse results from Li et al. [2025] for all OGBench tasks besides `humanoidmaze-md` and `antmaze-giant`. For IQL, Li et al. [2025] use $\alpha = 0.3$ for `cube-*`, $\alpha = 10.0$ for `puzzle-3x3`, and $\tau = 0.9$. For both `humanoidmaze-md` and `antmaze-giant`, we similarly use $\tau = 0.9$ and $\alpha = 10$. For ReBRAC, Li et al. [2025] use $\alpha = 0.1$, and we use an actor behavioral cloning coefficient of 0.01 and a critic behavioral cloning coefficient of 0.01 for `humanoidmaze-md`, following Park et al. [2025c], and an actor behavioral cloning coefficient of 0.003 and a critic behavioral cloning coefficient of 0.01 for `antmaze-giant`.

A.6 Offline-to-online training and prior data sampling protocols.

Offline-to-online training. For each of the Best-of-N, FQL, and action chunking policies, we first sample from an offline dataset of demonstrations for 1×10^6 iterations for training. Afterwards, this offline dataset is used to fill a fixed-size replay buffer of capacity 2×10^6 transitions, which is then refreshed with a new transition collected from online experience every gradient step, for an additional 1×10^6 steps.

Online training with prior data. For gaussian policies, there is no offline learning phase; every gradient update is accompanied by an environment step. An empty replay buffer of capacity 2×10^6 transitions is filled only with online experience from the agent. Every gradient step, half the batch is sampled from an offline dataset of demonstrations, and the other half is sampled from the online replay buffer.

Action chunking online. Our action chunking training protocol takes inspiration from Li et al. [2025]. Every online training step, the chunked policy is sampled from, yielding a chunk of actions. Each action in the chunk is executed in the environment, one action per gradient update, until the chunk is exhausted.

B Full Results

Table 4: **OGBench results.** Each cell is the success rate (%) at the end of online training (mean across seeds, equal-weight across the 5 tasks per group). **Bold** marks methods within 95% of the row maximum; an overbar marks methods within 95% of the per-actor-type maximum. The Total row is the equal-weight mean over the 5 OGBench groups. Cells without a bootstrap confidence interval are taken from prior work [Li et al., 2025].

	Best-of-N			**FQL**			**Gaussian**			**Action Chunking**		**Other baselines**	
	TD	TD-n	LQL	TD	TD-n	LQL	TD	TD-n	LQL	TD	LQL	ReBRAC	IQL
`cube-double`	68.3 [66,71]	67.9 [66,69]	83.5 [81,86]	18.3 [16,21]	49.8 [46,53]	86.2 [79,91]	0.2 [0,1]	0.0 [0,0]	18.7 [18,19]	**94.2** [92,96]	**95.1** [94,97]	30	0
`cube-triple`	11.1 [8,14]	0.7 [0,1]	36.2 [33,39]	9.3 [7,11]	6.9 [2,12]	31.2 [27,35]	0.0 [0,0]	0.0 [0,0]	0.2 [0,0]	35.2 [33,37]	**52.1** [50,54]	0	0
`scene`	90.6 [88,93]	**95.1** [93,97]	**95.4** [94,97]	92.3 [83,98]	68.1 [61,75]	93.9 [89,98]	69.4 [67,72]	90.4 [86,94]	89.5 [88,92]	**97.3** [96,98]	**97.0** [95,98]	**99**	39
`humanoid-md`	72.8 [63,87]	71.1 [69,73]	**96.2** [94,98]	40.4 [39,42]	28.6 [25,32]	65.2 [54,76]	20.5 [11,31]	7.9 [6,10]	48.8 [36,62]	15.4 [8,25]	22.2 [14,30]	3.5 [2,5]	23.1 [20,26]
`antmaze-giant`	0.1 [0,0]	37.5 [34,41]	44.8 [33,56]	4.8 [0,14]	28.8 [26,32]	53.3 [51,56]	23.7 [12,36]	19.0 [18,20]	**65.4** [62,69]	22.2 [12,34]	40.7 [37,45]	57.1 [53,62]	3.2 [2,4]
Total	48.6 [46,51]	54.5 [54,55]	**71.2** [69,74]	33.0 [31,35]	36.4 [34,39]	66.0 [63,69]	22.8 [20,26]	23.5 [22,24]	44.5 [42,47]	52.9 [50,56]	61.4 [60,63]	38	13

Table 5: `humanoidmaze-giant` **per-task success rate (%)** at the end of online training (mean across seeds), for the runs in Figure 1. **Bold** marks methods within 95% of the per-row maximum. The Total row is the equal-weight mean over the 5 `humanoidmaze-giant` tasks. Numbers in brackets are the 95% bootstrap CI; for the Total row, the bootstrap is stratified by task.

	TD	TD-4	TD-8	TD-16	TD-64	LQL ($L = 64$)
`task1`	0.0 [0,0]	18.0 [12,28]	1.3 [0,2]	0.0 [0,0]	0.0 [0,0]	**31.3** [2,58]
`task2`	0.0 [0,0]	63.3 [50,76]	36.0 [24,56]	6.7 [4,10]	4.7 [2,6]	**97.3** [96,100]
`task3`	0.0 [0,0]	7.3 [4,10]	4.7 [0,12]	0.0 [0,0]	0.0 [0,0]	**70.7** [66,76]
`task4`	0.0 [0,0]	4.7 [4,6]	0.0 [0,0]	0.0 [0,0]	0.0 [0,0]	**80.7** [64,92]
`task5`	0.0 [0,0]	**98.7** [96,100]	90.0 [82,96]	61.3 [54,66]	26.0 [20,36]	**98.7** [96,100]
Total	0.0 [0,0]	38.4 [35,41]	26.4 [23,30]	13.6 [12,15]	6.1 [5,8]	**75.7** [70,81]

Table 6: **RoboMimic results.** Each cell is the success rate (%) at the end of online training (mean across seeds). The Total row is the equal-weight mean of Square and Can. **Bold** marks methods within 95% of the row maximum; an overbar marks methods within 95% of the per-actor maximum.

	Best-of-N			**FQL**			**Gaussian**			**Action Chunking**	
	TD	TD-n	LQL	TD	TD-n	LQL	TD	TD-n	LQL	TD	LQL
Square	23.0 [19,27]	23.0 [16,28]	42.5 [39,46]	0.0 [0,0]	15.5 [8,22]	61.0 [60,63]	7.5 [3,12]	**92.5** [90,95]	**92.5** [90,94]	20.5 [19,22]	23.5 [16,31]
Can	81.0 [77,84]	80.5 [77,86]	**87.0** [86,88]	3.0 [0,6]	69.5 [63,73]	**90.0** [84,96]	2.5 [1,4]	82.5 [80,88]	84.0 [78,90]	82.0 [80,84]	**85.5** [80,90]
Total	52.0 [50,55]	51.7 [48,55]	64.8 [63,66]	1.5 [0,3]	42.5 [38,47]	75.5 [72,78]	5.0 [3,8]	**87.5** [86,90]	**88.2** [85,91]	51.2 [50,52]	54.5 [50,59]

B.1 Comparison to Optimality Tightening

We cloned the optimality tightening codebase of He et al. [2016] and updated their discrete-action-only implementation to use a continuous actor with a tanh-squashed Gaussian (like the Gaussian actor we use for LQL and TD($-n$)); our adapted code is available at `https://github.com/armaan-abraham/Q-Optimality-Tightening`. We used default hyperparameters from the repository, after matching the network size of the critic and the actor to those used in our experiments. We evaluated this agent for the same number of online training steps (2×10^6) as the gaussian LQL agent, and compare them in Figure 8.

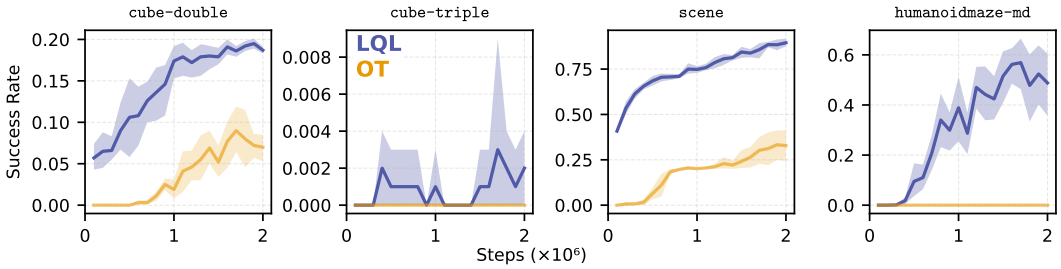


Figure 8: Mean success rate of LQL and OT across 4 task groups, evaluated on all 5 tasks in the group and with 4 seeds per task.

Table 7: **LQL vs. OT.** Each cell is the success rate (%) at the end of online training (mean across seeds, equal-weight across the 5 tasks per group). **Bold** marks the per-row maximum. The Total row is the equal-weight mean over the 4 task groups. Numbers in brackets are the 95% bootstrap CI; for the Total row, the bootstrap is stratified by task. Results for LQL are identical to the Gaussian column in Table 4.

	LQL	OT
`cube-double`	**18.7** [18,19]	7.0 [5,9]
`cube-triple`	**0.2** [0,0]	0.0 [0,0]
`scene`	**89.5** [88,92]	32.8 [25,41]
`humanoid-md`	**48.8** [36,62]	0.0 [0,0]
Total	**39.3** [36,43]	10.0 [8,12]

B.2 Stochastic environments

For the experiments in Section 5.5 (Figure 9), we created stochastic versions of OGBench environments by adding independent Gaussian noise to each dimension of the action before executing it in the environment, clipping to the action range $[-1, 1]$. For each degree of stochasticity, labeled by the standard deviation of the corresponding Gaussian, we recollected a separate offline dataset using the method from Park et al. [2025a] with this updated stochasticity, yielding a dataset of the same size as in Table 1.

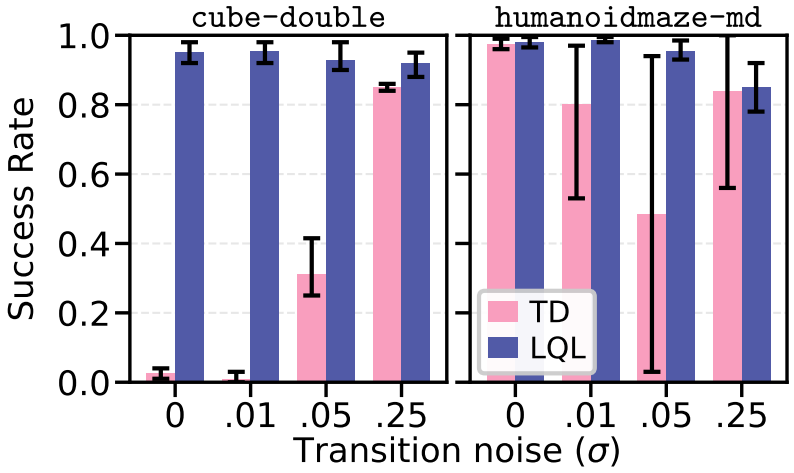


Figure 9: **LQL continues to perform on par with or better than TD in stochastic environments.** Each panel shows `task2` of the listed task group.

B.3 Hinge coefficient sweep

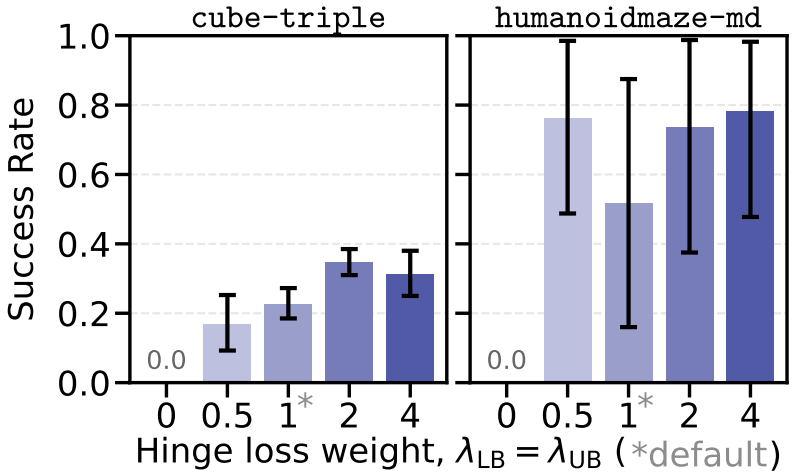


Figure 10: **Hinge coefficient sweep.** FQL actor was used with otherwise the same hyperparameters as Tables 2, 3. Each plot shows `task3` of the listed task group. Four seeds.

B.4 Isolating the effects of trajectory sampling and hinge penalties

LQL differs from standard TD both in *what* is sampled (short trajectories) and *what loss* is applied (additional hinge penalties). To disentangle these, we set the hinge-loss coefficients in Eq. 10 to zero, which reduces LQL to TD learning on sampled trajectories rather than individually sampled transitions. Figure 11 shows that LQL's hinge penalties yield further gains beyond this trajectory sampling control, supporting the role of the long-horizon backstop. Surprisingly, the trajectory sampling control in some cases performs better than standard transition-level TD. This may reflect the mechanism observed in Fu et al. [2025], where smaller batch sizes can yield better performance for TD due to reduced overfitting to inaccurate target network predictions. In this case, sampling transitions within trajectories may reduce the effective diversity of the batch in a way that produces the same effect.

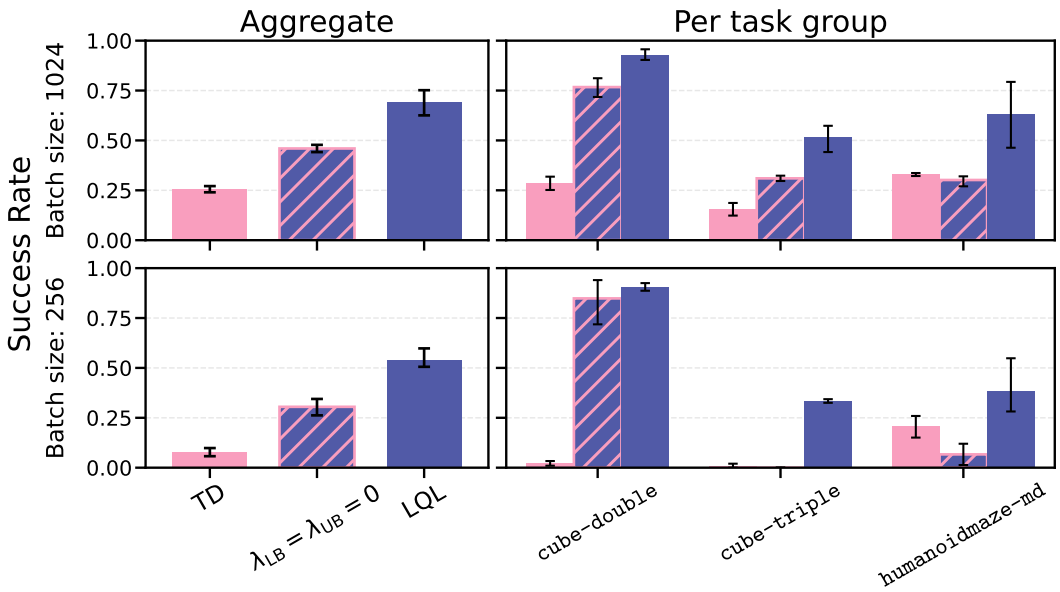


Figure 11: **LQL's gains are not explained by trajectory sampling alone; the hinge backstop contributes beyond this control.** Success rates for FQL policies are averaged over tasks 1–3 in `cube-double`, `cube-triple`, and `humanoidmaze-md`. The top row uses the configuration with the hyperparameters used in the rest of the paper, including a batch size of 1024 (LQL: 128 trajectories of length 8). The bottom row uses batch size 256 (LQL: 64 trajectories of length 4), with $\alpha = 50$ for `cube-double` and `cube-triple`, and $\alpha = 10$ for `humanoidmaze-md`.

B.5 Batch-size-controlled trajectory length sweep

Under a fixed compute-per-batch constraint, increasing the trajectory length requires reducing the number of sampled trajectories per batch. Figure 12 shows that when the batch size is held constant, the optimal trajectory length varies by task, with performance degrading at longer trajectory lengths in some environments. Taken in isolation, this degradation could be attributed to two potential causes: (a) instability from longer-horizon hinge penalties, or (b) the reduction in the number of sampled trajectories per batch, and the resulting reduced diversity of behavioral policies represented in the batch, yielding noisier hinge, TD, and actor losses alike. However, the result in Figure 5, in which the trajectory length is scaled without a corresponding reduction in the number of sampled trajectories per batch, shows consistent performance improvements with longer trajectories, supporting (b) over (a).

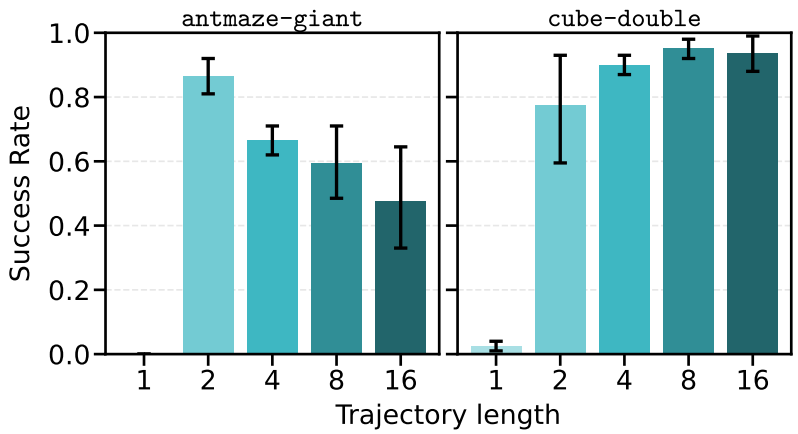


Figure 12: **With fixed batch size, optimal LQL trajectory length varies by task.** Each plot shows performance of FQL policies on `task2` of the listed environment. We keep the batch size fixed at 1024.

B.6 Scaling trajectory length for LQL vs. batch size for TD

We conduct more experiments of the form shown in Figure 5, namely scaling trajectory length, additionally showing the effect of an equivalent increase in batch size for TD. These experiments are conducted at a smaller batch size for both methods, with FQL actors with $\alpha = 10$ for `humanoidmaze-md` and $\alpha = 50$ for `cube-double`. Reflecting previous work [Fu et al., 2025], TD does not respond favorably to batch size scaling, while LQL consistently improves with longer trajectories (Figure 13).

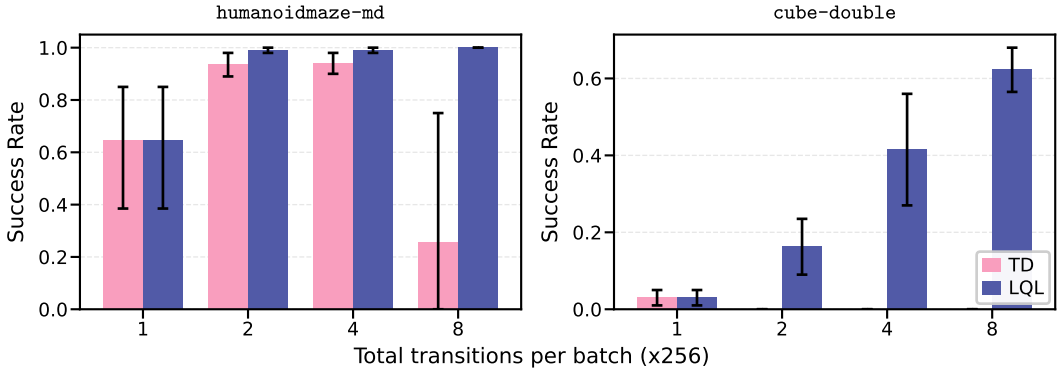


Figure 13: **Scaling compute via longer LQL trajectories is more effective than scaling TD with more independent transitions.** For matched scaling factors on the x-axis (which also correspond to LQL trajectory length), LQL benefits consistently from longer segments, while TD does not reliably improve with larger batches of individually sampled transitions. Each panel shows `task2` of the listed environment. Identical results are used for TD with batch size 256 and LQL with horizon length 1, because these two methods are identical in this case.

B.7 Hinge penalty activation vs. hinge distance

While training the LQL-with-FQL-actor agent, we recorded the hinge penalty activation frequency and magnitude throughout training, grouped by whether the penalty was a lower bound or upper bound and by the distance over which the penalty was computed in Figure 14. The penalty magnitude is measured as the average over all pairwise comparisons, including those that are zero (i.e., unactivated). The y-axis range of 2 to 8 for the lower bound and 0 to 6 for the upper bound reflects that the lower bound skips the hinge comparison for the next state since it is already included in the TD loss, while the upper bound includes the zero comparison because the same-state upper bound uses the target network evaluated on the policy-generated action at the same state, which is already computed for the

TD update at the previous transition. We see that broadly the hinge penalties activate more frequently for shorter-distance comparisons, but conversely tend to be larger in magnitude for longer-distance comparisons. One vaguely apparent pattern is that the hinge penalties amplify after a small delay from the beginning of online training.

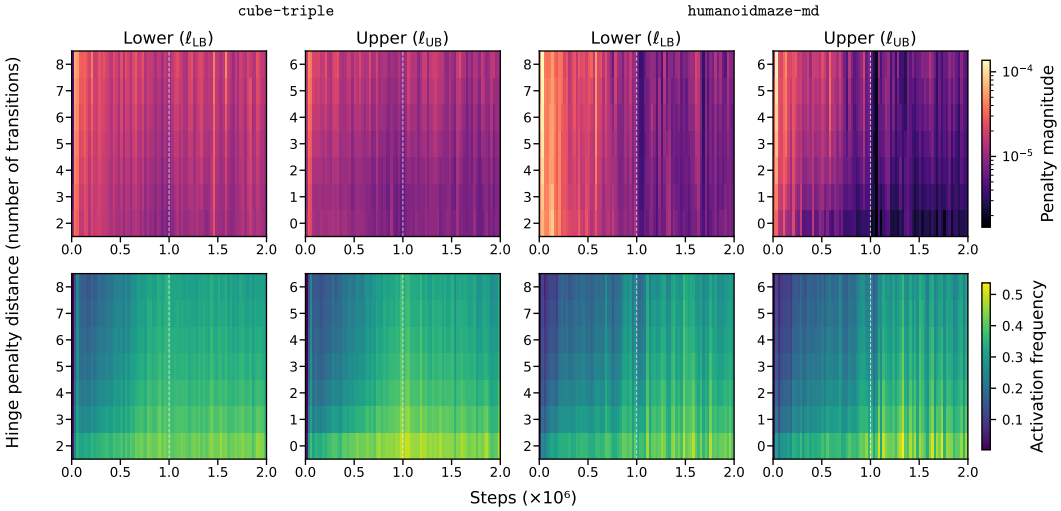


Figure 14: **Hinge penalty activation frequency and magnitude show task and training stage-dependent patterns**. Penalty magnitude is normalized by Q_θ^2 (using the batch-mean Q_θ at each step). Offline-online training transition marked by white dashed line. Averaged over two seeds per task (`task1` of each task group).

B.8 Computational requirements

Across the policy extraction families used in the main experiments, LQL incurs a 4.7% per-update slowdown on average over TD (Table 8).

Table 8: **Runtime comparison.** Iterations per second on a NVIDIA A40 GPU. Offline iter/sec for Best-of-N and FQL; online for Gaussian.

Policy type	TD (iter/sec)	LQL (iter/sec)	Slowdown
Best-of-N	57.1	56.0	2.0%
FQL	288.5	273.1	5.3%
Gaussian	138.8	132.5	4.5%
Average	161.5	153.9	4.7%

For the `humanoidmaze-giant` runs (Figure 1) which use a longer trajectory length $L = 64$, one potential concern is that the $\mathcal{O}(L^2)$ pairwise comparisons would dramatically slow down training. Table 9 shows that at this larger trajectory length, the additional runtime cost of LQL remains negligible.

Table 9: **Offline iterations per second** on a NVIDIA H100 GPU, for `humanoidmaze-giant` experiments in Figure 1.

Method	iter/sec
TD	29.91
TD-4	29.36
TD-8	30.06
TD-16	30.04
TD-64	29.76
LQL ($L = 64$)	29.72

B.9 LB-only and UB-only variants

We compare LQL with LB-only and UB-only variants, obtained by setting $\lambda_{\mathrm{UB}} = 0$ and $\lambda_{\mathrm{LB}} = 0$, respectively (Table 10). Zeroing the UB or LB penalty yields performance on par with or worse than the default values we use, supporting our design choice of including both UB and LB penalties in the LQL loss.

Table 10: **LB-only and UB-only variants.** Each cell is the success rate (%) at the end of online training (mean across 4 seeds) on `task2` of the listed task group. FQL actor was used with otherwise the same hyperparameters as Tables 2, 3. **Bold** marks the per-row maximum. Numbers in brackets are the 95% bootstrap CI.

	LQL (default) $\lambda_{\mathrm{LB}} = 1, \lambda_{\mathrm{UB}} = 1$	LB-only $\lambda_{\mathrm{LB}} = 1, \lambda_{\mathrm{UB}} = 0$	UB-only $\lambda_{\mathrm{LB}} = 0, \lambda_{\mathrm{UB}} = 1$
`humanoidmaze-md`	98 [97,100]	**100** [98,100]	0 [0,0]
`cube-triple`	**33** [12,48]	4 [1,9]	0 [0,0]

C Theoretical Analysis

C.1 Bounded false penalties due to stochasticity

We analyze both hinge penalties (LB, then UB) at $Q_\theta = Q^*$ in stochastic environments, and denote these as false penalties. We establish bounds on false penalties that depend only on the reward bound $R_{\max}$, γ, and the suboptimality of the experience in the replay buffer, and not on the number of steps L over which the penalties are computed. If the false penalty bound were necessarily unbounded with increasing L, this could indicate consistent disadvantages to scaling L, but our analysis shows that this is not the case.

LB hinge. The L-step LB violation signal at trajectory position i is

$$Z_L = G_{i:i+L} + \gamma^L Q^*(s_{i+L}, a^*(s_{i+L})) - Q^*(s_i, a_i).$$

Telescoping via the Bellman equation gives

$$Z_L = \sum_{k=0}^{L-1} \gamma^k \epsilon_{i+k} \; - \; \gamma \sum_{k=0}^{L-2} \gamma^k \Delta_{i+k+1},$$

where $\epsilon_j = r_j + \gamma Q^*(s_{j+1}, a^*(s_{j+1})) - Q^*(s_j, a_j)$ is the 1-step Bellman noise (with $\mathbb{E}[\epsilon_j \mid s_j, a_j] = 0$) and $\Delta_j = Q^*(s_j, a^*(s_j)) - Q^*(s_j, a_j) \geq 0$ is the suboptimality gap.

Per-step decomposition. Re-indexing the gap sum (substituting $j = k + 1$) yields

$$Z_L = \epsilon_i + \sum_{j=1}^{L-1} \gamma^j \big(\epsilon_{i+j} - \Delta_{i+j}\big).$$

Define per-step terms

$$W_j \; \triangleq \; \begin{cases} \epsilon_i & j = 0, \\ \epsilon_{i+j} - \Delta_{i+j} & j = 1, \ldots, L-1, \end{cases}$$

so that $Z_L = \sum_{j=0}^{L-1} \gamma^j W_j$.

Assumption. *Bounded rewards*: $|r_j| \leq R_{\max}$ for all j.

This implies $|Q^*(s,a)| \leq Q_{\max} \triangleq R_{\max}/(1-\gamma)$ and therefore $|\epsilon_j| \leq R_{\max} + (1+\gamma)Q_{\max}$ and $|\Delta_j| \leq 2Q_{\max}$. Consequently, every W_j is bounded:

$$|W_j| \; \leq \; M \; \triangleq \; R_{\max} + (1+\gamma)Q_{\max} + 2Q_{\max},$$

and in particular $\mathbb{E}[W_j^2] \leq M^2$ for all j.

Mean of Z_L. All expectations below are over the replay trajectory distribution. Since $\mathbb{E}[\epsilon_{i+j} \mid s_{i+j}, a_{i+j}] = 0$ and $\Delta_{i+j} \geq 0$, for $L \geq 2$:

$$\mathbb{E}[Z_L] = \sum_{j=0}^{L-1} \gamma^j \mathbb{E}[W_j] = -\sum_{j=1}^{L-1} \gamma^j \, \mathbb{E}[\Delta_{i+j}] \; \triangleq \; -\mu_D(L) \; \leq \; 0.$$

Defining $\bar{\Delta}_{\mathrm{LB}} \triangleq \mathbb{E}[\Delta_{i+1}]$, $\mu_D(L) \geq \gamma \bar{\Delta}_{\mathrm{LB}}$ because every $\Delta_{i+j} \geq 0$.

Variance bound. Expanding $\mathrm{Var}(Z_L)$:

$$\mathrm{Var}(Z_L) = \sum_{j=0}^{L-1} \sum_{j'=0}^{L-1} \gamma^{j+j'} \, \mathrm{Cov}(W_j, W_{j'}).$$

By Cauchy–Schwarz,

$$|\,\mathrm{Cov}(W_j, W_{j'})| \; \leq \; \sqrt{\mathrm{Var}(W_j) \; \mathrm{Var}(W_{j'})} \; \leq \; \sqrt{\mathbb{E}[W_j^2]\,\mathbb{E}[W_{j'}^2]} \; \leq \; M^2.$$

Therefore

$$\mathrm{Var}(Z_L) \;\le\; M^2 \sum_{j=0}^{L-1}\sum_{j'=0}^{L-1} \gamma^{j+j'} = M^2 \left(\sum_{j=0}^{L-1} \gamma^j\right)^2 \;\le\; M^2 \Big(\frac{1}{1-\gamma}\Big)^2 \;\triangleq\; V^\infty,$$

which is finite and independent of L.

Lemma C.1. *For any random variable X with $\mathbb{E}[X] = -\mu$ ($\mu \ge 0$) and $\mathrm{Var}(X) \le V$:*

$$\Pr(X > 0) \le \frac{V}{V + \mu^2}, \tag{11}$$

$$\mathbb{E}\big[[X]_+\big] \le \frac{V}{2\big(\sqrt{V + \mu^2} + \mu\big)}. \tag{12}$$

Proof. Inequality (11) is the Cantelli inequality [Cantelli, 1928] applied to $\tilde{X} = X + \mu$ (mean-zero, variance $\le V$) at threshold μ.

For (12), use the identity $[X]_+ = (X + |X|)/2$, so

$$\mathbb{E}\big[[X]_+\big] = \frac{\mathbb{E}[X] + \mathbb{E}\big[|X|\big]}{2} = \frac{-\mu + \mathbb{E}\big[|X|\big]}{2}.$$

To bound $\mathbb{E}\big[|X|\big]$, apply Cauchy–Schwarz:

$$\mathbb{E}\big[|X|\big] = \mathbb{E}\big[|X| \cdot 1\big] \le \sqrt{\mathbb{E}\big[|X|^2\big]}\,\sqrt{\mathbb{E}\big[1^2\big]} = \sqrt{\mathbb{E}\big[X^2\big]} = \sqrt{V + \mu^2}.$$

Substituting:

$$\mathbb{E}\big[[X]_+\big] \le \frac{\sqrt{V + \mu^2} - \mu}{2} \cdot \frac{\sqrt{V + \mu^2} + \mu}{\sqrt{V + \mu^2} + \mu} = \frac{V}{2\big(\sqrt{V + \mu^2} + \mu\big)}.$$

□

False penalty bound. The LB hinge penalty produces a false violation whenever $Z_L > 0$. Applying Lemma C.1 to Z_L for $L \ge 2$, $\mu = \mu_D(L)$, $V \le V^\infty$, and using $\mu_D(L) \ge \gamma\bar{\Delta}_{\mathrm{LB}}$:

$$\Pr([Z_L]_+^2 > 0) = \Pr(Z_L > 0) \le \frac{V^\infty}{V^\infty + \gamma^2\bar{\Delta}_{\mathrm{LB}}^2}, \tag{13}$$

$$\mathbb{E}\big[[Z_L]_+\big] \le \frac{V^\infty}{2\big(\sqrt{V^\infty + \gamma^2\bar{\Delta}_{\mathrm{LB}}^2} + \gamma\bar{\Delta}_{\mathrm{LB}}\big)}. \tag{14}$$

For the expected squared penalty, since $|W_j| \le M$,

$$[Z_L]_+ \;\le\; |Z_L| \;\le\; M\sum_{j=0}^{L-1}\gamma^j \;\le\; \frac{M}{1-\gamma} \;=\; \sqrt{V^\infty},$$

and therefore $[Z_L]_+^2 \le \sqrt{V^\infty}\,[Z_L]_+$. Taking expectations:

$$\mathbb{E}\big[[Z_L]_+^2\big] \;\le\; \sqrt{V^\infty}\,\mathbb{E}\big[[Z_L]_+\big] \;\le\; \frac{(V^\infty)^{3/2}}{2\big(\sqrt{V^\infty + \gamma^2\bar{\Delta}_{\mathrm{LB}}^2} + \gamma\bar{\Delta}_{\mathrm{LB}}\big)}.$$

All bounds are finite, independent of L, and decrease with the suboptimality of the experience in the replay buffer $\bar{\Delta}_{\mathrm{LB}}$.

UB hinge. Define the L-step UB violation signal at trajectory position i as

$$U_L = G_{i:i+L} + \gamma^L Q^*(s_{i+L}, a_{i+L}) - Q^*(s_i, a^*(s_i)).$$

Using the same telescoping as for Z_L and substituting $Q^*(s_i, a^*(s_i)) = Q^*(s_i, a_i) + \Delta_i$, we obtain

$$U_L = \sum_{k=0}^{L-1}\gamma^k\epsilon_{i+k} \;-\; \sum_{k=0}^{L}\gamma^k\Delta_{i+k}.$$

Per-step decomposition. Define

$$\widetilde{W}_k \triangleq \begin{cases} \epsilon_{i+k} - \Delta_{i+k} & k = 0, \dots, L-1, \\ -\Delta_{i+L} & k = L, \end{cases}$$

so that $U_L = \sum_{k=0}^{L} \gamma^k \widetilde{W}_k$. Since $|\epsilon_{i+k}| \le R_{\max} + (1+\gamma)Q_{\max}$ and $|\Delta_{i+k}| \le 2Q_{\max}$, every $\widetilde{W}_k$ satisfies $|\widetilde{W}_k| \le M$ with the same constant $M = R_{\max} + (1+\gamma)Q_{\max} + 2Q_{\max}$ as in the lower-bound case.

Mean of U_L. Since $\mathbb{E}[\epsilon_{i+k} \mid s_{i+k}, a_{i+k}] = 0$ and $\Delta_{i+k} \ge 0$,

$$\mathbb{E}[U_L] = -\sum_{k=0}^{L} \gamma^k \mathbb{E}[\Delta_{i+k}] \triangleq -\mu_U(L) \le 0.$$

Defining $\bar{\Delta}_{\mathrm{UB}} \triangleq \mathbb{E}[\Delta_i]$, we have $\mu_U(L) \ge \bar{\Delta}_{\mathrm{UB}}$ since every summand is nonnegative.

Variance bound. By the same Cauchy–Schwarz argument as for Z_L, using $|\widetilde{W}_k| \le M$:

$$\mathrm{Var}(U_L) \le M^2 \left(\sum_{k=0}^{L} \gamma^k \right)^2 \le M^2 \left(\frac{1}{1-\gamma} \right)^2 = V^{\infty}.$$

False penalty bound. Applying Lemma C.1 to U_L with $\mu = \mu_U(L) \ge \bar{\Delta}_{\mathrm{UB}}$ and $V \le V^{\infty}$:

$$\Pr([U_L]_+^2 > 0) \le \frac{V^{\infty}}{V^{\infty} + \bar{\Delta}_{\mathrm{UB}}^2}, \tag{15}$$

$$\mathbb{E}\big[[U_L]_+^2\big] \le \frac{(V^{\infty})^{3/2}}{2\big(\sqrt{V^{\infty} + \bar{\Delta}_{\mathrm{UB}}^2} + \bar{\Delta}_{\mathrm{UB}}\big)}. \tag{16}$$

Both bounds are finite, independent of L, and decrease with the suboptimality gap $\bar{\Delta}_{\mathrm{UB}}$ in the replay buffer. The bounds, at the very least, show that sending trajectory length $\to \infty$ will not also send the false penalty $\to \infty$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state that LQL augments TD with hinge penalties derived from optimality inequalities, requires no auxiliary networks or additional forward passes per update, and improves over 1-step and n-step TD on the evaluated benchmarks; these claims are supported by the derivation in Section 4 and the experiments in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the in-expectation nature of the optimality inequality and the resulting bias under stochastic dynamics, points to the bound on this bias derived in Appendix C.1, and references the empirical evaluation in Section 5.5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Appendix C.1 states the bounded-rewards assumption explicitly and provides the full derivation of the false-penalty bounds, including the supporting Lemma C.1 (Cantelli inequality) with proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The method is fully specified in Section 4 and Algorithm 1; environments and datasets are described in Section 5.1 and Appendix A.2; policy extraction families, offline-to-online protocols, and implementation details are in Section 5.2 and Appendices A.3–A.6; and all hyperparameters are listed in Tables 2 and 3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the LQL training code and configurations needed to reproduce the main experiments. The offline datasets we use (OGBench Park et al. [2025a] and RoboMimic Mandlekar et al. [2021]) are publicly available through the cited works.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Optimizer, learning rate, batch size, network architecture, target update rate, behavior-regularization coefficients, trajectory length, and all other relevant hyperparameters are reported in Appendix A.5 (Tables 2 and 3); the offline-to-online training and evaluation protocols are described in Appendix A.6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report 95% confidence intervals from a curve-level bootstrap (1000 iterations) over training runs, with multi-task aggregates resampled within each task following Agarwal et al. [2021]; intervals appear in all tables (Tables 4–6) and shaded regions in figures, and the protocol is described in Appendix A.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B.8 reports per-update iterations/sec on an NVIDIA A40 GPU for the main experiments (Table 8) and on an NVIDIA H100 GPU for the longer-trajectory `humanoidmaze-giant` runs (Table 9).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and the research conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper proposes a foundational improvement to off-policy value learning. We are not aware of a direct path from this methodological contribution to specific positive or negative societal impacts beyond those generally associated with reinforcement learning research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pre-trained models, datasets, or other artifacts that pose a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All third-party assets we use are credited via citation in the relevant sections, including the OGBench Park et al. [2025a] and RoboMimic Mandlekar et al. [2021] environments and datasets, the FQL Park et al. [2025c] and QC-FQL Li et al. [2025] recipes, and the optimality-tightening codebase He et al. [2016] we adapt for comparison in Appendix B.1.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the LQL training code with documentation describing how to reproduce the main experiments, including the configurations corresponding to the hyperparameters in Tables 2 and 3.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component of the proposed method or its evaluation.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.