AQRL: QUASIMETRIC REINFORCEMENT LEARNING WITH SPLIT STATE ENCODING AND APPROXIMATE NEAREST COMPLETION

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-conditioned tasks require an agent to reach a state satisfying a given goal. Such a goal often constrains only some state coordinates, which we define as specified coordinates; the rest are free coordinates. Quasimetric Reinforcement Learning (QRL) is a goal-conditioned reinforcement learning method that performs well. It takes states as inputs to estimate state-reaching cost. However, it has two limitations. First, its actor receives sampled real states as goals during training but artificially constructed states during testing, creating a mismatch in goal inputs. Second, QRL trains its actor to reach a single specific state for each training sample, even though other states that satisfy the same goal may have lower reaching costs. To address both limitations, we propose **Approximate nearest-goal Quasimetric Reinforcement Learning (AQRL)** with two components: **split state encoding (SSE)** and **approximate nearest completion (ANC)**. SSE encodes specified and free coordinates separately, enabling the actor to use only specified-coordinate latents as goal inputs in training and testing, resolving the mismatch. ANC searches over free-coordinate latents during training to find a completion with lower estimated reaching cost. Across fourteen online tasks with five training seeds, AQRL outperforms QRL on thirteen tasks and ranks first among all compared methods on twelve, achieving 66.1% mean success versus 24.5% for QRL and 45.3% for the strongest compared baseline.

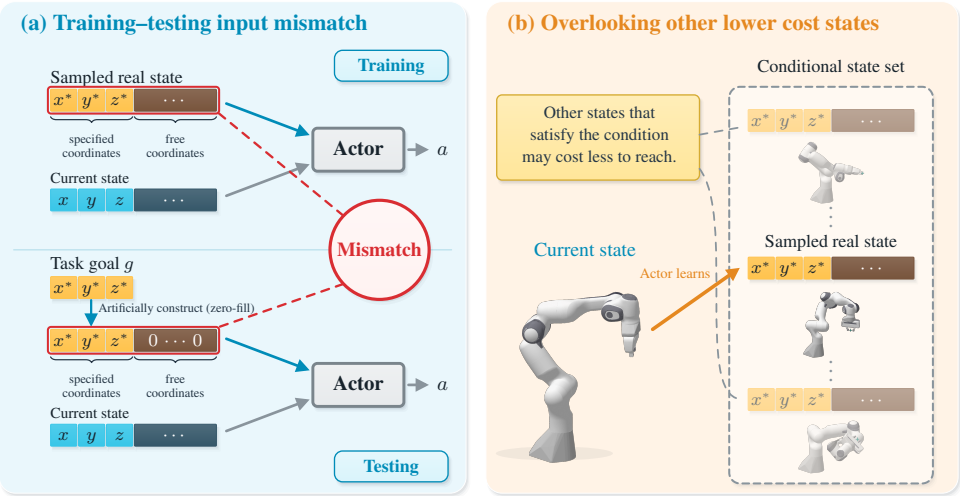


Figure 1: Two limitations of QRL. (a) The actor receives sampled real states as goals during training and artificially constructed states during testing, creating a goal-input mismatch. (b) Each training sample guides the actor toward a single specific state, overlooking other states that satisfy the same goal and may cost less to reach.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Goal-conditioned reinforcement learning provides explicit goal inputs, enabling agents to learn policies for reaching different goals through interaction with the environment. For example, a robotic arm may learn to place an object at a desired position, while a mobile robot may learn to navigate to a specified destination. Recent work has made progress in online goal-reaching performance (Bortkiewicz et al., 2025; Wang et al., 2025).

Learning reaching costs is one approach to goal-conditioned control. Quasimetric Reinforcement Learning (QRL) estimates the cost of reaching one state from another using a learned quasimetric (Wang et al., 2023). This distance satisfies the triangle inequality but need not be symmetric. For example, driving from the bottom of a hill to the top may cost more than driving in the opposite direction. QRL encourages distances between sampled states to increase while using observed transitions to constrain distances between states directly connected by those transitions. If one action takes the agent from one state to another, the estimated reaching cost between them should not exceed the cost of that action. Through the triangle inequality, these constraints on individual transitions also bound distances between more distant states. This process can be understood as stretching a net: states are nodes, and observed transitions form connections between them. Increasing distances between sampled states stretches the net outward, while transition costs limit how far the corresponding connections can stretch, thereby bounding distances between nodes connected by paths through the net. To train its actor, QRL encodes the current state and a sampled goal state into latents. A learned dynamics model predicts the next-state latent under the actor's action, and the quasimetric estimates the remaining reaching cost to the goal state. The actor is then optimized to choose actions with lower estimated reaching costs. Recent work further learns quasimetric distances from multistep returns (Zheng et al., 2026).

Building on QRL, we study reaching costs when goals constrain only some state coordinates. For example, an object's target position leaves robot joint configurations and velocities unconstrained. Recent research studies how to represent goals for the actor (Park et al., 2026; Chuck et al., 2025). We define the constrained coordinates as specified coordinates and the rest as free coordinates. Applying QRL to these goals has two limitations. First, QRL's actor receives sampled real states as goals during training but artificially constructed states during testing, creating a mismatch in goal inputs, as shown in Figure 1(a). A typical test goal retains the specified-coordinate values but fills free coordinates with zeros. Second, QRL trains its actor to reach a single specific state for each training sample, even though other states that satisfy the same goal may have lower reaching costs. Different joint configurations can satisfy the same object position at different costs, but QRL does not compare these completions, as shown in Figure 1(b).

We propose **Approximate nearest-goal Quasimetric Reinforcement Learning (AQRL)** with two components to address these limitations. AQRL keeps QRL's state-to-state distance learning, but changes the actor's goal input and the target used to compute its training loss. These changes address the input mismatch and single-state training target illustrated in Figure 1.

The first component, **split state encoding (SSE)**, encodes specified and free coordinates separately. This enables the actor to use only the specified-coordinate latent as its goal input in both training and testing. We concatenate the two encoders' outputs to represent the full current state for the actor, distance model, and dynamics model. During training, we extract the specified coordinates from a sampled real goal state and encode only those coordinates as the actor's goal input. During testing, we encode the task goal with the same encoder. The actor thus receives the same type of goal input in both stages, without filling in free coordinates, removing input mismatch.

The second component, **approximate nearest completion (ANC)**, changes the target used to train the actor. During training, it fixes the desired specified-coordinate latent and searches over free-coordinate latents for a completion with lower estimated reaching cost from the predicted next state. This completion guides the actor update; it is not an extra actor input. During training and testing, the actor receives the current state and task goal. During testing, it acts directly without search.

We evaluate AQRL on fourteen online tasks with five training seeds per task. We compare methods with matched parameter counts, the same interaction budgets, and the same evaluation protocol. AQRL outperforms QRL on thirteen tasks and ranks first among all compared methods on twelve, with 66.1% mean test success versus 24.5% for QRL and 45.3% for TD-InfoNCE (Zheng et al.,

2024). Over 20%–100% of the interaction budget, it also achieves the highest normalized area under the validation-success curve on twelve tasks. This indicates better sample efficiency. In a six-task search ablation, AQRL with SSE alone outperforms QRL. Adding ANC with 32 search updates raises mean success across these tasks by 4.83 percentage points.

We make three contributions to learning with partially specified goals:

1. **Split state encoding.** SSE encodes both coordinate groups separately to enable consistent goal inputs across training and testing while retaining full current-state information.
2. **Completion search during training.** We develop ANC to search over free-coordinate latents during training and use the selected completion to guide actor updates. The trained actor acts directly from the current state and task goal without search.
3. **Results on fourteen online tasks.** With five training seeds, AQRL outperforms QRL on thirteen tasks and ranks first among the compared methods on twelve. A six-task ablation illustrates additional gains from ANC, with individual-seed results in the appendix.

2 Related Work

Goal-conditioned learning. GCSL imitates goal-conditioned actions from self-collected trajectories (Ghosh et al., 2021), while C-Learning estimates future-state densities through recursive classification (Eysenbach et al., 2021). CRL relates contrastive inner products to goal-conditioned values (Eysenbach et al., 2022), and TD-InfoNCE connects trajectory fragments through temporal-difference updates (Zheng et al., 2024). These methods provide alternative learning signals.

Structured goal-reaching costs. MRN, triangle-inequality-preserving neural distances, and PQE/IQE provide structured distance representations (Liu et al., 2023; Pitis et al., 2020; Wang & Isola, 2022b;a). QRL supplies the state-to-state distance-learning objective used in AQRL and analyzes optimal-value recovery (Wang et al., 2023). Beyond the distance architecture, learning objectives also shape reaching-cost estimates. Contrastive successor features yield temporal distances (Myers et al., 2024), and subsequent offline work combines successor representations with quasimetric constraints (Myers et al., 2025). MQE fits quasimetric distances using multistep Monte Carlo returns (Zheng et al., 2026). For offline state–goal pairs, conservative value learning additionally penalizes estimates for unconnected pairs while using quasimetric structure for connected pairs (Ke et al., 2025). These studies improve state-to-state reaching-cost estimation. AQRL retains this distance model and searches over target completions to guide actor training.

Goal representations and target search. Goal representations determine how the desired target is presented to an actor. Dual Goal Representations characterize a goal state through temporal distances *from other states to that goal* (Park et al., 2026). QRL also outlines a goal-set extension using dedicated goal markers and added transitions from successful states, but does not provide a separate analysis or conclusions specific to goal sets (Wang et al., 2023, Appendix A). AQRL uses a task-provided coordinate split and searches over free-coordinate latents to guide actor training. Learned distances also define rewards in Dynamical Distance Learning (Hartikainen et al., 2020) and guide graph-based subgoal search over replay observations in SoRB (Eysenbach et al., 2019). Unlike replay-subgoal planning, ANC searches for a latent target completion during actor training; the trained actor executes directly without search.

3 Problem Setup

Objective. We seek an actor that takes the complete current state and a desired goal as inputs and reaches a state satisfying that goal with low expected cost. Goal-conditioned learning shares value functions (Schaul et al., 2015) and policies (Park et al., 2025) across goals. Goals may constrain some or all state coordinates.

Task definition. Let $\mathcal{S}$ and $\mathcal{G}$ be the state and goal spaces. Each task provides specified-coordinate indices I_g and free-coordinate indices I_n. The task goal $g \in \mathcal{G}$ specifies desired values for coordinates s_{I_g}, while s_{I_n} denotes the free coordinates. A navigation goal, for example, may specify

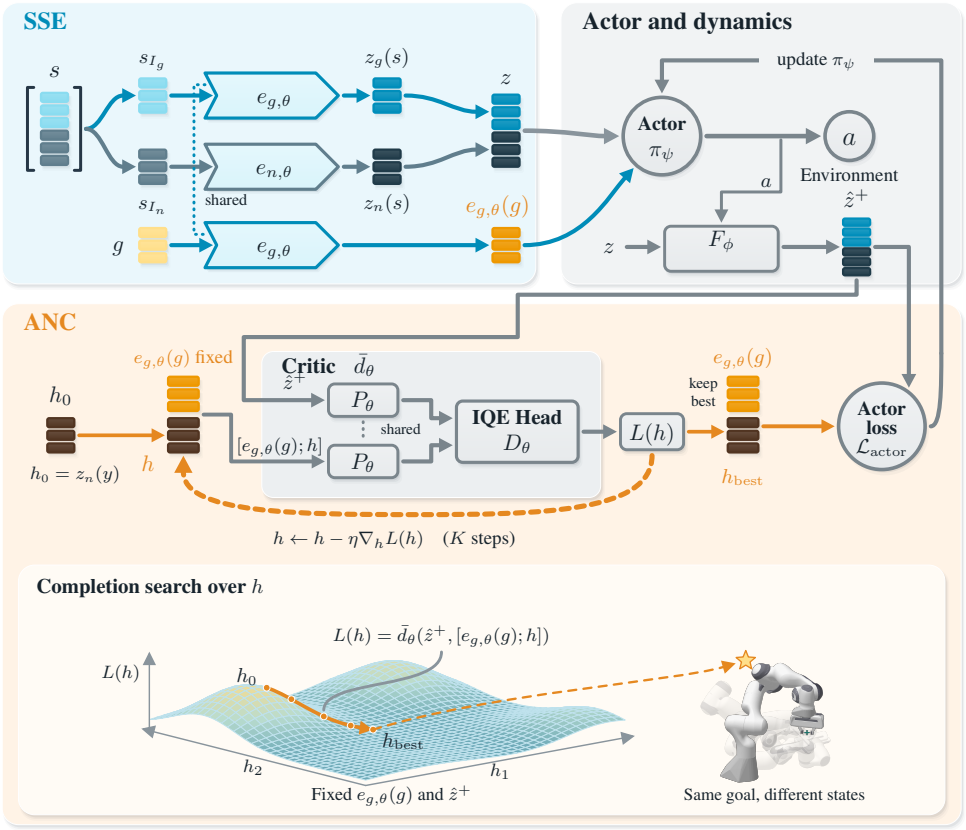


Figure 2: AQRL pipeline. SSE separately encodes specified and free coordinates. ANC holds $e_{g,\theta}(g)$ and $\hat{z}^+$ fixed, updates h using the shared critic, and detaches the best candidate for the actor update. During testing, the actor outputs actions without completion search.

position without specifying posture or velocity. Object-position goals likewise leave other state factors unspecified in robotic manipulation (Plappert et al., 2018; Chuck et al., 2025). Complete-state goals have $I_n = \varnothing$. We use unit transition cost to model goal-reaching costs.

Agent–environment interaction. The environment is a Markov decision process with action space $\mathcal{A}$, transition law $P(s' \mid s, a)$, and nonnegative cost $c(s, a, s')$. We write the goal-conditioned actor as π. At each step, it selects $a_t \sim \pi(\cdot \mid s_t, g)$ and observes $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The actor is shared across goals and learned from interaction with the initially unknown environment.

Task completion. Given a task-defined tolerance region $\mathcal{B}(g)$, success means entering

$$\mathcal{S}_g = \{s \in \mathcal{S} : s_{I_g} \in \mathcal{B}(g)\}. \tag{1}$$

Any state in this set satisfies the goal, including different completions of free coordinates.

Reaching the goal. Action selection must account for future reaching costs, which can differ from s to y and from y to s because of dynamics and action constraints. Quasimetrics allow these costs to differ while satisfying

$$d(s, y) \geq 0, \qquad d(s, s) = 0, \qquad d(s, y) \leq d(s, u) + d(u, y). \tag{2}$$

These properties motivate quasimetric models of reaching costs (Wang et al., 2023); the next section describes their use in our method.

4 APPROXIMATE NEAREST-GOAL QUASIMETRIC REINFORCEMENT LEARNING

Figure 2 summarizes AQRL's encoders, actor, and completion search during training.

To model the reaching costs introduced in Section 3, let E_θ encode a state, P_θ project the encoding, and D_θ compute a quasimetric distance satisfying Eq. 2:

$$\bar{d}_\theta(u, v) = D_\theta(P_\theta(u), P_\theta(v)), \qquad d_\theta(s, y) = \bar{d}_\theta(E_\theta(s), E_\theta(y)). \tag{3}$$

We parameterize the distance with Interval Quasimetric Embeddings (IQE) (Wang & Isola, 2022a).

4.1 SPLIT STATE ENCODING (SSE)

SSE uses separate encoders for the two coordinate groups (I_g, I_n):

$$E_\theta(s) = [e_{g,\theta}(s_{I_g}); e_{n,\theta}(s_{I_n})] = [z_g(s); z_n(s)]. \tag{4}$$

The SSE module in Figure 2 shows the two encoders and their concatenated output. The actor receives the full current-state latent $E_\theta(s)$ and the specified-coordinate latent $e_{g,\theta}(g)$ as its goal input during both training and testing. The same encoder $e_{g,\theta}$, with shared parameters, encodes both the current state's specified coordinates s_{I_g} and the desired goal g. The critic and dynamics model also receive both state parts. Appendix B.1 gives the AntNavigate split. When $I_n = \varnothing$, only the specified-coordinate encoder is needed.

4.2 APPROXIMATE NEAREST COMPLETION (ANC)

For each replay pair (s, y), QRL trains its actor toward a single specific state y. ANC instead keeps $g = y_{I_g}$ fixed and searches over free-coordinate latents for a completion with lower estimated reaching cost to guide the actor update. We define the latent conditional target set $\mathcal{Z}_g = \{[e_{g,\theta}(g); h] : h \in \mathbb{R}^{d_n}\}$, where h is the free-coordinate latent and has dimension d_n. The specified-coordinate latent is fixed and h is free to vary. This set need not equal $E_\theta(\mathcal{S}_g)$ or contain only physically feasible encodings. When $I_n = \varnothing$, there is no h to optimize and no search is needed.

For an action a, the dynamics model predicts $\hat{z}^+ = F_\phi(E_\theta(s), a)$. The ideal cost is

$$C_{\theta,\phi}(s, a, g) = \inf_h \bar{d}_\theta\big(F_\phi(E_\theta(s), a), [e_{g,\theta}(g); h]\big). \tag{5}$$

This cost depends on the predicted next state. It is defined in latent space and need not equal the cost of reaching a physically feasible state.

The ANC module in Figure 2 searches over h, starting at $h_0 = z_n(y)$ for a replay target y with $g = y_{I_g}$. Let $\widehat{h}_K$ be the lowest-cost candidate among the initialization and K gradient updates. This search aims to approximate the minimum estimated reaching cost in Eq. 5. The specified-coordinate latent, prediction, and network parameters remain fixed during this search. The selected completion replaces the sampled target in the actor loss, not in the actor's input (Section 4.3).

The search also keeps its initialization among the candidates. For a fixed model and predicted next state, selecting the best candidate means that the chosen completion has no higher estimated cost than the initial sampled target. This property applies within one search; it does not compare costs across successive model updates. It explains why ANC retains the best candidate instead of always using the final gradient iterate: a later update may increase the estimated cost, and the earlier candidate can still guide the actor update.

ANC may support self-correction as new interaction data correct errors in the distance and dynamics models guiding subsequent searches. Algorithm 1 summarizes SSE and ANC in an actor update; Appendix A.2 gives the complete training procedure.

4.3 TRAINING OBJECTIVES

Actor loss. We stop gradients through the completion selected by ANC and minimize the following actor loss:

$$\mathcal{L}_{\text{actor}} = \mathbb{E}\left[\bar{d}_\theta\big(F_\phi(E_\theta(s), a), \text{sg}([e_{g,\theta}(g); \widehat{h}_K])\big) + \mathcal{L}_{\text{entropy}}\right]. \tag{6}$$

Algorithm 1 AQRL actor update with SSE and ANC

Require: Replay pairs (s, y); coordinate split (I_g, I_n); search steps K; step size η

Hold encoder, distance-model, and dynamics parameters θ, ϕ fixed
$z \leftarrow \mathrm{sg}([e_{g,\theta}(s_{I_g}); e_{n,\theta}(s_{I_n})])$ ▷ SSE
$g \leftarrow y_{I_g}$
Sample $a_\psi \sim \pi_\psi(\cdot \mid z, \mathrm{sg}(e_{g,\theta}(g)))$ by reparameterization
$\hat{z}^+ \leftarrow F_\phi(z, a_\psi)$
$h \leftarrow \mathrm{sg}(e_{n,\theta}(y_{I_n}))$, $h_{\mathrm{best}} \leftarrow h$
Define $L(h) = \bar{d}_\theta(\mathrm{sg}(\hat{z}^+), [\mathrm{sg}(e_{g,\theta}(g)); h])$
for $k = 1, \ldots, K$ **do** ▷ ANC; independently for each replay pair
    $h \leftarrow h - \eta \nabla_h L(h)$
    If $L(h) < L(h_{\mathrm{best}})$, set $h_{\mathrm{best}} \leftarrow h$
end for
$z^\star \leftarrow \mathrm{sg}([e_{g,\theta}(g); h_{\mathrm{best}}])$
Update ψ to minimize $\mathbb{E}[\bar{d}_\theta(\hat{z}^+, z^\star) + \mathcal{L}_{\mathrm{entropy}}]$ ▷ Through a_ψ and F_ϕ

Here, $a \sim \pi_\psi(\cdot \mid E_\theta(s), e_{g,\theta}(g))$ is sampled by reparameterization, and sg stops gradients. $\mathcal{L}_{\mathrm{entropy}}$ is the entropy regularization term (Haarnoja et al., 2018); details are given in Appendix A.4.

During the actor update, the completion and the encoder, critic, and dynamics parameters remain fixed. Gradients pass through the critic and dynamics model to the actor's action, updating only the actor parameters. This follows stochastic value-gradient methods (Heess et al., 2015) and latent-imagination control (Hafner et al., 2020), using a one-step prediction and a detached ANC target.

Dynamics loss. QRL trains its dynamics model using bidirectional squared IQE error (Wang et al., 2023). AQRL adds a full-latent mean squared error (MSE) term. In some IQE-only training runs, we observed stable IQE prediction error while full-latent MSE continued to increase and training failed catastrophically. This observation motivated adding MSE to directly constrain errors in the predicted latent. Defining $\delta_\leftrightarrow(u, v) = \frac{1}{2}(\bar{d}_\theta(u, v)^2 + \bar{d}_\theta(v, u)^2)$, we use

$$\mathcal{L}_{\mathrm{dyn}} = w\,\mathbb{E}\left[w_q \delta_\leftrightarrow(\hat{z}^+, E_\theta(s')) + w_m\,\mathrm{MSE}(\hat{z}^+, E_\theta(s'))\right]. \tag{7}$$

Critic loss. Following Wang et al. (2023), we learn quasimetric state-to-state distances with $\mathcal{L}_{\mathrm{critic}} = \mathcal{L}_{\mathrm{local}} + \mathcal{L}_{\mathrm{global}}$:

$$\mathcal{L}_{\mathrm{local}} = \lambda\left(\mathbb{E}_{s,s'}[\mathrm{ReLU}(d_\theta(s, s') - c)^2] - \varepsilon^2\right), \tag{8}$$

$$\mathcal{L}_{\mathrm{global}} = \mathbb{E}_{s,\tilde{s}}\left[\frac{1}{\beta}\log(1 + \exp(\beta(b - d_\theta(s, \tilde{s}))))\right]. \tag{9}$$

The local term penalizes distances exceeding the observed one-step cost $c = 1$, with tolerance ε and positive multiplier λ. The global term separates randomly paired replay states, preventing all distances from becoming zero; b and β control its offset and smoothness. Appendix A.3 gives the settings and multiplier update.

We jointly update the encoders, critic, and dynamics model by minimizing $\mathcal{L}_{\mathrm{critic}} + \mathcal{L}_{\mathrm{dyn}}$ with one optimizer, then update the actor using Eq. 6.

5 Experiments

5.1 Experimental setup

We compare methods on fourteen online tasks, including Fetch manipulation (Plappert et al., 2018), Panda tasks built on panda-gym (Gallouédec et al., 2021), Maze2D from D4RL (Fu et al., 2020), and a DeepMind Control Suite reacher (Tunyasuvunakool et al., 2020).

All methods use matched parameter counts and the same interaction budgets, training seeds, and evaluation protocol. Appendices A and B detail the network architectures, implementation sources, training configurations, and evaluation procedures.

Table 1: Success rate (%): mean ± sample SD over five training seeds. Success means satisfying the goal at least once during a test episode. Bold marks the highest mean; the macro mean weights tasks equally and is computed before rounding the task means. PnP = PickAndPlace, J = Joints, U/L/M/H = UMaze/large/Medium/hard. Full task names and budgets appear in Table 6 and Table 8; individual runs appear in Appendix C.4.

Task	QRL	TD-InfoNCE	GCSL	C-Learning	CRL	AQRL (Ours)
AntMaze-U	0.0±0.0	0.8±0.4	4.6±3.6	0.0±0.0	1.8±1.1	**42.4±4.6**
AntNavigate	4.9±4.6	17.9±1.8	12.5±1.3	4.3±9.5	11.9±1.7	**98.0±1.1**
FetchPnP	13.3±4.3	35.8±19.0	6.8±0.7	4.3±0.1	6.8±3.7	**72.3±15.6**
FetchPush	16.8±12.4	69.0±8.5	9.0±0.8	7.1±0.1	29.0±22.2	**91.3±3.6**
FetchSlide	26.3±5.8	**45.1±6.4**	4.6±2.4	1.4±1.8	25.2±5.3	24.4±1.0
Maze2D-L	65.8±7.7	24.9±1.7	6.4±0.6	11.8±19.0	39.3±17.2	**73.2±3.9**
PandaFlip	18.4±4.3	19.4±8.2	18.0±4.1	4.8±1.0	13.4±5.1	**35.0±6.2**
PandaPnP	37.4±7.9	51.0±2.1	11.9±2.2	2.6±0.2	10.5±1.3	**53.5±4.4**
PandaPnP-J	5.3±1.5	36.9±18.8	7.1±1.8	2.5±0.0	6.8±4.0	**51.8±3.9**
PandaPush-J	11.8±0.7	77.8±32.4	19.9±3.5	9.5±0.0	26.5±9.7	**95.1±1.9**
PandaSlide	28.0±8.4	48.4±25.9	20.2±3.5	0.3±0.4	22.2±5.7	**53.9±7.5**
PointMaze-M	86.2±1.9	37.4±5.0	8.8±3.7	8.0±1.7	60.6±5.9	**92.6±2.9**
Pusher	24.8±8.2	**79.3±42.6**	0.1±0.2	0.7±1.0	10.1±5.0	44.1±21.1
Reacher-H	4.1±0.6	90.5±1.4	20.7±3.3	72.8±30.9	11.2±2.8	**98.5±1.0**
Macro mean	24.5	45.3	10.8	9.3	19.7	**66.1**

5.2 Goal-reaching success

AQRL achieves higher mean test success than QRL on thirteen of fourteen tasks and leads all compared methods on twelve (Table 1). The macro mean, which weights tasks equally, is 66.1% for AQRL, 24.5% for QRL, and 45.3% for TD-InfoNCE. These gains span navigation and manipulation: AQRL achieves 98.0% success on AntNavigate and 95.1% on PandaPushJoints, compared with QRL's 4.9% and 11.8%, respectively. On PointMaze, where QRL already reaches 86.2%, AQRL achieves 92.6%. Thus, gains also occur on a task where QRL already has a high success rate. TD-InfoNCE leads on FetchSlide and Pusher.

5.3 Sample efficiency

We assess sample efficiency using validation success over training. Figure 3 shows four representative tasks; Appendix C.2 provides learning curves and AUC results for all fourteen tasks.

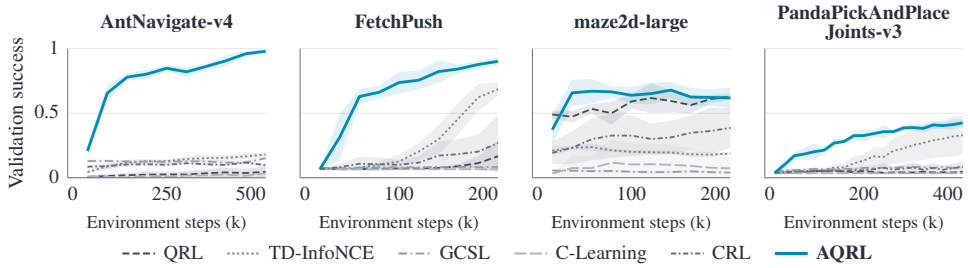


Figure 3: Validation success on four representative tasks. Lines show five-seed means; bands show sample SD, clipped to $[0, 1]$. Checkpoints are linearly aligned within their common observed interval, without smoothing or extrapolation.

For each task and seed, normalized AUC is the integral of validation success over 20%–100% of the interaction budget divided by the interval length. Table 10 reports five-seed means and sample SDs as percentages. AUC summarizes performance during training, whereas checkpoint test success evaluates the policy selected by validation performance. AQRL leads on twelve tasks under both metrics; TD-InfoNCE leads on FetchSlide and Pusher.

5.4 GRADIENT-SEARCH GAINS

To quantify the contribution of search, we use the main-table configuration and vary only the number of gradient updates $K \in \{0, 8, 16, 32\}$. AQRL/0 retains SSE while disabling ANC updates ($K = 0$), so each actor update uses the sampled target's latent without optimizing its free-coordinate part. We evaluate this ablation on six tasks. Appendix C.1 reports individual-seed ablation results for these six tasks.

AQRL/0 has higher mean success than QRL on all six tasks, supporting the effectiveness of SSE without completion search on these tasks. Comparing AQRL/0 with $K > 0$ measures the additional contribution of searching for the actor-loss target during training.

Across the six tasks, macro success increases from 67.54% at $K = 0$ to 68.92%, 71.28%, and 72.37% at $K = 8, 16, 32$, respectively (Table 2). The $K = 32$ gain is 4.83 percentage points.

PandaPush-J and AntNavigate improve at each tested budget, gaining 6.30 and 2.30 percentage points from $K = 0$ to $K = 32$. PandaSlide gains 9.80 points and FetchPush gains 2.64 points, although both peak at $K = 16$. PandaPnP-J initially declines and then reaches 51.80% at $K = 32$, a 2.50-point gain over $K = 0$. Pusher similarly dips at $K = 8$ before reaching 44.14% at $K = 32$, a 5.42-point gain, but its sample SD is 21.12 points and only two of five paired seeds improve.

5.5 FRACTION OF FREE COORDINATES

We analyze one observation per task using the five-seed mean test success from Table 1 and report two-sided p-values.

Across all fourteen tasks, QRL's test success is negatively correlated with the fraction of free coordinates $f_n = |I_n|/(|I_g| + |I_n|)$, as shown in Figure 4 (Pearson $r = -0.820$, $p = 0.00033$; Spearman $\rho = -0.690$, $p = 0.0064$). These negative correlations indicate that QRL tends to achieve lower success on tasks with larger fractions of free coordinates.

QRL reaches 65.8% on Maze2D and 86.2% on PointMaze, both with $f_n = 50\%$, whereas its success is at most 11.8% on all four tasks with $f_n > 90\%$. On AntMaze UMaze, with 2 specified coordinates and 105 free coordinates, QRL has zero success in all five seeds, while AQRL reaches a mean success of 42.4% (SD: 4.6 percentage points). This pattern is consistent with the goal-input mismatch discussed in Section 1: artificially constructed goal states contain manually assigned values in a larger share of their coordinates as f_n increases, potentially widening the difference from the sampled real states used as training goals.

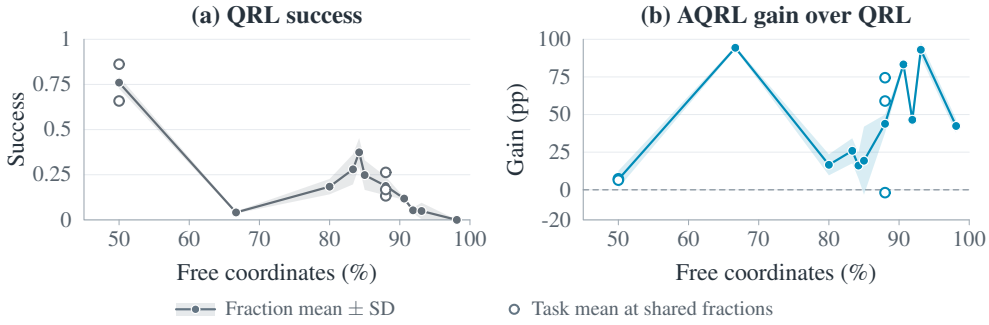


Figure 4: Fraction of free coordinates. (a) QRL test success; (b) AQRL-minus-QRL gain in percentage points. At each fraction, tasks are averaged within seed. Filled markers and bands show the five-seed mean ± sample SD; hollow markers show task means at shared fractions. Lines connect observations without smoothing. Task-level values appear in Table 3.

AQRL's success gain is not significantly associated with f_n across these tasks (Pearson $r = 0.354$, $p = 0.214$; Spearman $\rho = 0.451$, $p = 0.106$). Some tasks with large fractions of free coordinates show large gains, but FetchSlide has a -1.9-point gain despite $f_n = 88\%$. The coordinate fraction alone does not explain the performance gains; action spaces and other task properties also differ.

Table 2: Completion-search ablation on six tasks. Each entry is test success (%): mean ± sample standard deviation across five training seeds. AQRL/K uses K search updates; $K = 0$ disables search. All other settings match the corresponding main-table runs. Macro rows weight tasks equally; Δ_{32-0} is in percentage points (pp). Bold marks the highest mean among the four budgets. Individual-seed ablation results for these six tasks are in Appendix C.1.

Task	AQRL/0	AQRL/8	AQRL/16	AQRL/32	Δ_{32-0}
AntNavigate	95.68±0.85	96.68±1.18	97.08±2.46	**97.98±1.15**	+2.30
FetchPush	88.64±4.14	90.80±1.12	**91.82±2.92**	91.28±3.56	+2.64
PandaPnP-J	49.30±4.62	48.80±1.48	48.90±4.67	**51.80±3.90**	+2.50
PandaPush-J	88.80±4.19	91.80±2.66	93.90±1.92	**95.10±1.92**	+6.30
PandaSlide	44.10±6.68	48.50±10.76	**56.10±7.77**	53.90±7.48	+9.80
Pusher	38.72±9.32	36.96±6.21	39.88±4.63	**44.14±21.12**	+5.42
Macro success	67.54	68.92	71.28	**72.37**	+4.83
Gain over $K = 0$ (pp)	+0.00	+1.38	+3.74	**+4.83**	–

Table 3: State dimensions and success rate (%) at the main-result checkpoints. d_s and d_g are the state and goal dimensions; $f_n = (d_s - d_g)/d_s$ is the fraction of free coordinates. Δ is AQRL's gain over QRL in percentage points. Bold marks the higher success mean.

Task	d_s	d_g	f_n (%)	QRL	AQRL (Ours)	Δ (pp)
AntMaze-U	107	2	98.1	0.0	**42.4**	+42.4
AntNavigate	29	2	93.1	4.9	**98.0**	+93.1
FetchPnP	25	3	88.0	13.3	**72.3**	+59.0
FetchPush	25	3	88.0	16.8	**91.3**	+74.5
FetchSlide	25	3	88.0	**26.3**	24.4	−1.9
Maze2D-L	4	2	50.0	65.8	**73.2**	+7.4
PandaFlip	20	4	80.0	18.4	**35.0**	+16.6
PandaPnP	19	3	84.2	37.4	**53.5**	+16.1
PandaPnP-J	37	3	91.9	5.3	**51.8**	+46.5
PandaPush-J	32	3	90.6	11.8	**95.1**	+83.3
PandaSlide	18	3	83.3	28.0	**53.9**	+25.9
PointMaze-M	4	2	50.0	86.2	**92.6**	+6.4
Pusher	20	3	85.0	24.8	**44.1**	+19.3
Reacher-H	6	2	66.7	4.1	**98.5**	+94.4

6 Conclusion

AQRL combines SSE with ANC. SSE encodes specified and free coordinates separately, enabling the actor to use only specified-coordinate latents as goal inputs in training and testing, resolving the goal-input mismatch. The actor also receives the full current-state latent. ANC searches over free-coordinate latents for a completion with lower estimated reaching cost to compute the actor loss during training. At test time, the actor outputs actions without completion search. Experiments on fourteen tasks show improvements in success rates and sample efficiency. The six-task ablation shows that ANC provides additional gains in task success.

Limitations. Our experiments use vector states with task-provided splits into specified and free coordinates. Such splits are difficult to define for images, limiting applicability to image observations. ANC adds computation during actor training through candidate projections, IQE evaluations, and target gradients. Searched latents may not correspond to feasible states, so their estimated costs may not be achievable. More search steps can also reduce success: PandaSlide and FetchPush peak at $K = 16$ in our six-task ablation. Lower critic values generally indicate that goals are easier to reach, but do not necessarily lead to higher overall success rates.

Future Work. Future work could use observed states to guide ANC toward feasible completions and adapt the search budget to balance sample efficiency and computation. Comparing estimated costs with actual reaching costs could help assess when ANC provides reliable training targets. Results from deeper residual CRL networks (Wang et al., 2025) also motivate studying how AQRL scales with network depth, model size, and task complexity.

AI USE STATEMENT

Generative AI tools assisted with refining research hypotheses, literature review, method implementation, analysis of experimental records, interpretation of results, and manuscript preparation, including figures, translation, and language editing. The authors reviewed the AI-assisted materials and take full responsibility for the accuracy and integrity of the paper and its accompanying artifacts.

REPRODUCIBILITY STATEMENT

The appendices document the algorithm implementation, experimental settings, baseline sources, and evaluation protocol. Results across training seeds are also provided to support reproducibility and assessment of variability.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2858e880333b3cd64f8192f13ddcca2f-Abstract-Conference.html.

Caleb Chuck, Fan Feng, Carl Qi, Chang Shi, Siddhant Agarwal, Amy Zhang, and Scott Niekum. Null counterfactual factor interactions for goal-conditioned reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36848567d39a5128e671ad04a6075374-Abstract-Conference.html.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012. URL https://arxiv.org/abs/1702.03118.

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 15220–15231, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-Learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620, 2022. doi: 10.52202/068431-2580. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/e7663e974c4ee7a2b475a4775201ce1f-Abstract-Conference.html.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. URL https://arxiv.org/abs/2004.07219.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. In *NeurIPS Workshop on Robot Learning: Self-Supervised and Lifelong Learning*, 2021. URL https://www.robot-learning.ml/2021/files/A3.pdf.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rALA0Xo6yNJ.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning*, volume 80, pp. 1861–1870, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lmhaVtvr.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/148510031349642de5ca0c544f31b2ef-Abstract.html.

Kaiqiang Ke, Qian Lin, Zongkai Liu, Shenghong He, and Chao Yu. Conservative offline goal-conditioned implicit V-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29591–29607. PMLR, 2025. URL https://proceedings.mlr.press/v267/ke25a.html.

Bo Liu, Yihao Feng, Qiang Liu, and Peter Stone. Metric residual network for sample efficient goal-conditioned reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8799–8806, 2023. doi: 10.1609/aaai.v37i7.26058.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37076–37096. PMLR, 2024. URL https://proceedings.mlr.press/v235/myers24a.html.

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0661. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1c5956164472d6d8123d574aa75cd063-Abstract-Conference.html.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ecd92623ac899357312aaa8915853699-Abstract-Conference.html.

Seohong Park, Deepinder Mann, and Sergey Levine. Dual goal representations. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f8cd7eb180deb5c6ec876de2b442fa72-Abstract-Conference.html.

Silviu Pitis, Harris Chan, Kiarash Jamali, and Jimmy Ba. An inductive bias for distances: Neural nets that respect the triangle inequality. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJeiDpVFPr.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018. URL https://arxiv.org/abs/1802.09464.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320. PMLR, 2015. URL https://proceedings.mlr.press/v37/schaul15.html.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. doi: 10.1109/IROS.2012.6386109. URL https://roboti.us/lab/papers/TodorovIROS12.pdf.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. doi: 10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzcinski, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5268. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e74ee34cc0f2d0780f34ee77d8fba25b-Abstract-Conference.html.

Tongzhou Wang and Phillip Isola. Improved representation of asymmetrical distances with interval quasimetric embeddings. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022a. URL https://openreview.net/forum?id=KRiST_rzkGl. Proceedings Track.

Tongzhou Wang and Phillip Isola. On the learning and learnability of quasimetrics. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=y0VvIg25yk.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 36411–36430. PMLR, 2023. URL https://proceedings.mlr.press/v202/wang23al.html.

Bill Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Scaling goal-conditioned reinforcement learning with multistep quasimetric distances. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/eddc0fab5a42f8d8de6eb5566cd9f1d3-Abstract-Conference.html.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=0akLDTFR9x.

APPENDIX CONTENTS

A AQRL implementation . . . 13
A.1 Network architecture and configuration . . . 13
A.2 Training loop and replay sampling . . . 14
A.3 Distance-model objectives . . . 15
A.4 Action distribution and entropy adaptation . . . 15
B Experimental setup and evaluation . . . 16
B.1 Environments and success conditions . . . 16
B.2 Baseline implementations and goal inputs . . . 17
B.3 Evaluation and checkpoint selection . . . 18
C Supplementary results . . . 19
C.1 Six-task search-budget ablation . . . 19
C.2 Complete learning curves and AUC . . . 19
C.3 Training stability and replay coverage . . . 22
C.4 Individual-seed validation and test results . . . 22
C.5 Baseline reproduction checks . . . 32

A AQRL IMPLEMENTATION

A.1 NETWORK ARCHITECTURE AND CONFIGURATION

The actor, encoders, dynamics increment, and projector use an input projection, one four-layer residual block, and an output layer, as shown in Figure 5. Each hidden transformation applies LayerNorm (Ba et al., 2016) and SiLU (Elfwing et al., 2018); the block uses an identity shortcut (He et al., 2016), following the layout of Wang et al. (2025). The actor, dynamics, and projector have hidden width 1024. Encoder branch widths depend on the task's coordinate split; Table 4 gives the resolved AntNavigate example, with 57 specified-coordinate and 455 free-coordinate latent dimensions. The dynamics predicts an increment, $\hat{z}^+ = z + f_\phi(z, a)$. The projector outputs 2048 coordinates for an IQE head with 64 components. Table 5 collects the main training settings.

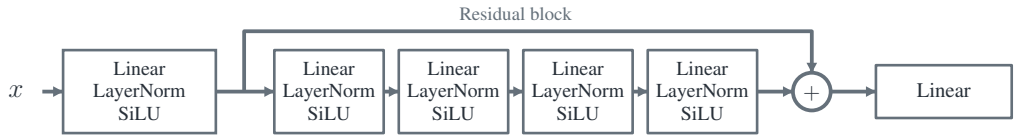


Figure 5: Residual MLP used within the AQRL modules. The input projection and four transformations inside the residual block each use Linear, LayerNorm, and SiLU; the output layer is linear. Module dimensions for AntNavigate are given in Table 4.

Table 4: AQRL network dimensions on AntNavigate. Each MLP uses one four-layer residual block, preceded by an input projection and followed by an output layer, as shown in Figure 5. The actor's 16 outputs parameterize eight action means and scales.

Module	Input	Hidden width	Output
Specified-coordinate encoder	2	355	57
Free-coordinate encoder	27	968	455
Actor	569	1024	16
Dynamics increment	520	1024	512
Quasimetric projector	512	1024	2048

Table 5: Main AQRL configuration. Task-specific evaluation counts are in Table 8. Parameter counts exclude frozen target networks. The search ablation varies only the number of completion updates ($K = 0, 8, 16, 32$); see Appendix C.1.

Setting	AQRL (Ours)
Trainable parameters	21.66M–21.83M
MLP architecture	Residual; LayerNorm; SiLU
State encoding	Split encoders; 512 latent dimensions
Critics/training	One/joint
Branch normalization	None
Completion direction	Minimize distance
Completion updates/step size	32/1.0
Completion optimizer	SGD; retain best candidate
Dynamics loss	IQE+MSE
Weights (w, w_q, w_m)	$(0.1, 1, 1)$
Actor/critic learning rate	$3 \times 10^{-5}/10^{-4}$
Batch size	256
Interaction budget	200k–500k
Training seeds	1000–1004
Validation/test	Table 8
Validation interval	20k or 50k
Checkpoint/actions	Validation-selected/deterministic

A.2 Training loop and replay sampling

Algorithm 2 specifies the environment collection, model updates, and adaptive weights around the actor update in Algorithm 1. Actor-training pairs give equal weight to targets from other replay transitions and later states in the same episode. The actor update uses the freshly updated encoders and distance/dynamics models. The definitions of the distance losses and entropy adaptation follow below; the dynamics and actor losses are defined in Eqs. 7 and 6.

Algorithm 2 AQRL training loop

Require: Task coordinate split (I_g, I_n); interaction budget T; collection/update schedule; search steps K and step size η

Initialize actor π_ψ, encoders and distance model θ, dynamics ϕ, replay $\mathcal{R}$, and weights $\lambda = \mathrm{softplus}(\ell)$, $\alpha = e^{\xi}$
while fewer than T environment interactions have been collected **do**
    Collect a scheduled batch of complete episodes into $\mathcal{R}$: uniform random actions for initial prefill, then $\pi_\psi(\cdot \mid E_\theta(s), e_{g,\theta}(g_{\mathrm{task}}))$
    for each scheduled optimization update **do**
        Sample a transition minibatch and form actor-training pairs (s, y) from replay
        Take descent steps in (θ, ϕ) and an ascent step in ℓ on $\mathcal{L}_{\mathrm{local}} + \mathcal{L}_{\mathrm{global}} + \mathcal{L}_{\mathrm{dyn}}$
        Update the actor using Algorithm 1 with the updated models, replay pairs (s, y), and search settings (K, η)
        Take an ascent step in ξ on $\mathcal{L}_{\mathrm{entropy}}$, using the entropy estimate from the same actor update
    end for
end while
return π_ψ and encoders $e_{g,\theta}, e_{n,\theta}$

A.3 Distance-model objectives

The critic objectives in Eqs. 8–9 follow Wang et al. (2023). The main configuration uses unit cost $c = 1$, $\varepsilon = 0.25$, $\beta = 0.1$, and $b = 15$. We parameterize the positive multiplier as $\lambda = \mathrm{softplus}(\ell)$, initialize it to 0.01, and take gradient ascent steps in ℓ on the local loss while minimizing the joint critic and dynamics objective over the model parameters (Algorithm 2).

A.4 Action distribution and entropy adaptation

The actor outputs a mean μ and an unconstrained scale parameter ρ, with $\sigma = \mathrm{softplus}(\rho) + 10^{-4}$. It samples $u \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$ and sends $a = b_a + c_a \odot \tanh(u)$ to the environment, where b_a and $c_a > 0$ shift and scale each action coordinate to its allowed range. Applying the change-of-variables formula for this action distribution (Haarnoja et al., 2018) gives the conditional differential entropy

$$\mathcal{H}(a \mid s, g) = \mathcal{H}(u \mid s, g) + \mathbb{E}_u \left[\sum_i \log(1 - \tanh^2(u_i)) \right] + \sum_i \log c_{a,i}. \tag{10}$$

We compute Gaussian entropy analytically. The second term accounts for $\tanh$ compression and is estimated with 100 reparameterized samples. The last term accounts for action scaling.

The actor's entropy term is $\mathcal{L}_{\mathrm{entropy}} = \alpha(\mathcal{H}_{\mathrm{target}} - \widehat{\mathcal{H}}(a \mid s, g))$. We update the positive weight α in the opposite direction to the actor: it increases when estimated action entropy is below the target and decreases when entropy is above it. For normalized m-dimensional actions in $[-1, 1]^m$, the default entropy target is set to $-m$.

B EXPERIMENTAL SETUP AND EVALUATION

B.1 ENVIRONMENTS AND SUCCESS CONDITIONS

Figure 6 covers the fourteen tasks. The non-Panda simulations use MuJoCo (Todorov et al., 2012); panda-gym uses PyBullet (Gallouédec et al., 2021). The task names identify the actual wrappers used in the experiments, including later environment versions and goal interfaces.

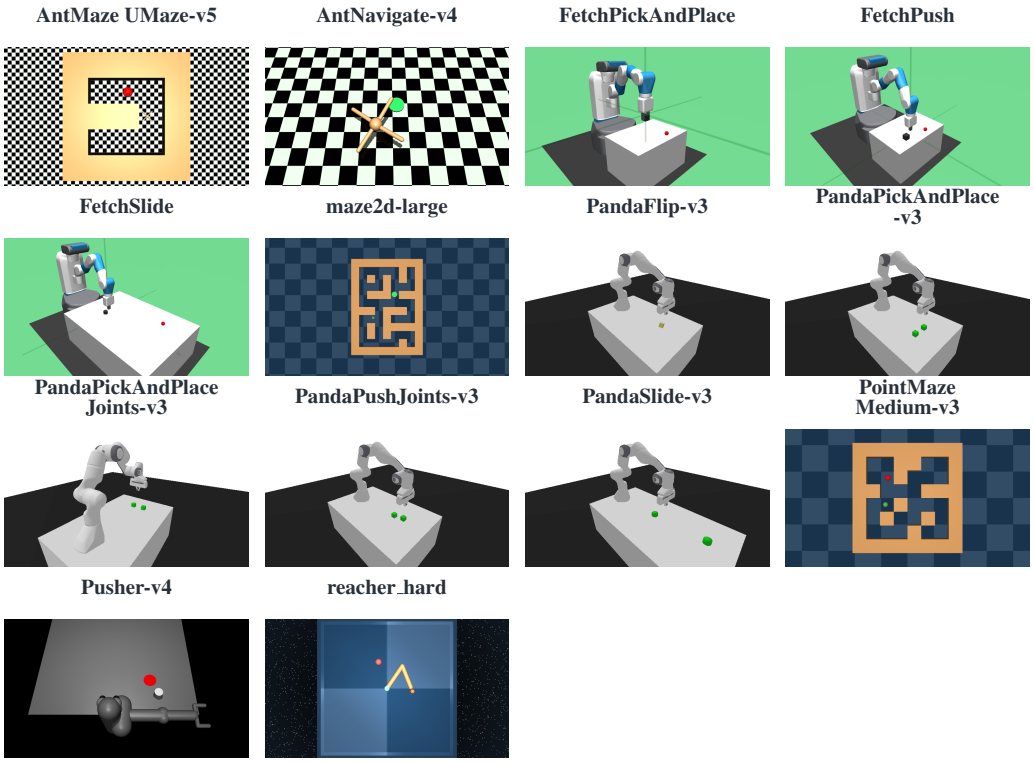


Figure 6: Representative simulator views of all fourteen tasks, in the same alphabetical order as Figure 7 and the result tables. FetchPickAndPlace shows a scripted grasp-and-lift frame; the other panels show reset states. PandaPickAndPlace and its joint-control variant share a scene, shown from different cameras; their action spaces differ. Goal markers are native except for rendering-only green markers for AntNavigate and Maze2D.

Task coordinates. Table 6 specifies the task coordinates and success conditions for all fourteen environments. These conditions are identical across methods, even when their actor goal inputs include additional coordinates. The Panda joint-control variants retain their corresponding object-position goals; PandaFlip instead specifies object orientation.

AntNavigate's 29-dimensional state contains 15 generalized positions and 14 generalized velocities, but its goal constrains only global x, y position. The remaining 27 coordinates describe torso height and orientation, joint positions, and velocities. The artificially constructed goal used here has a zero orientation quaternion, which does not represent a valid rotation. AntNavigate and Pusher use fixed versions of Gym Ant-v3 and Pusher-v2 with the corresponding goal interfaces. We interact with Maze2D online rather than train on its static offline dataset.

Table 6: Task goals and success conditions shared by all methods. Indices I_g are zero-based in the state supplied to the algorithms. δ_p is Euclidean position error in environment units. For PandaFlip, $\delta_o = 1 - \langle u, u_g \rangle^2$ is panda-gym's orientation error for achieved and desired unit quaternions u, u_g. Reacher uses its native binary success reward, with a combined target–fingertip radius of 0.025.

Task	Goal coordinates	I_g	Success condition
AntMaze UMaze-v5	Agent position (XY)	0, 1	$\delta_p \leq 0.45$
AntNavigate-v4	Agent position (XY)	0, 1	$\delta_p < 0.5$
FetchPickAndPlace	Object position (XYZ)	3, 4, 5	$\delta_p < 0.05$
FetchPush	Object position (XYZ)	3, 4, 5	$\delta_p < 0.05$
FetchSlide	Object position (XYZ)	3, 4, 5	$\delta_p < 0.05$
maze2d-large	Agent position (XY)	0, 1	$\delta_p < 0.5$
PandaFlip-v3	Object quaternion	0, 1, 2, 3	$\delta_o < 0.2$
PandaPickAndPlace-v3	Object position (XYZ)	0, 1, 2	$\delta_p < 0.05$
PandaPickAndPlaceJoints-v3	Object position (XYZ)	0, 1, 2	$\delta_p < 0.05$
PandaPushJoints-v3	Object position (XYZ)	0, 1, 2	$\delta_p < 0.05$
PandaSlide-v3	Object position (XYZ)	0, 1, 2	$\delta_p < 0.05$
PointMaze Medium-v3	Agent position (XY)	0, 1	$\delta_p \leq 0.45$
Pusher-v4	Object position (XYZ)	0, 1, 2	$\delta_p < 0.05$
reacher_hard	Fingertip position (XY)	0, 1	$\delta_p \leq 0.025$

B.2 Baseline implementations and goal inputs

QRL (Wang et al., 2023), TD-InfoNCE (Zheng et al., 2024), CRL (Eysenbach et al., 2022), GCSL (Ghosh et al., 2021), and C-Learning (Eysenbach et al., 2021) are ported from publicly released implementations associated with their original papers. For each baseline, we use its online-learning variant. We preserve the core objectives, representation heads, and update rules, adjusting MLP widths and some depths to match parameter counts. Appendix C.5 reports additional implementation checks on selected source tasks.

Table 7: Baseline implementation sources. Links identify the public online-learning implementations; commit IDs specify the versions used.

Algorithm	Reference	Official implementation
QRL	Wang et al. (2023)	quasimetric-learning/quasimetric-rl
TD-InfoNCE	Zheng et al. (2024)	chongyi-zheng/td_infonce 18f4e7e5
CRL	Eysenbach et al. (2022)	Google Research: contrastive_rl ec7c3d34
GCSL	Ghosh et al. (2021)	dibyaghosh/gcsl cfae5609
C-Learning	Eysenbach et al. (2021)	Google Research: c_learning ec7c3d34

Capacity and training. All methods have 21.4M–22.1M trainable parameters; AQRL has 21.66M–21.83M, depending on the task. Counts exclude frozen target networks. TD-InfoNCE uses four hidden layers of 1192 units. CRL and C-Learning use four hidden layers of 1544 units. GCSL uses four hidden layers of 2680 units except on AntNavigate, where its recorded joint-action model uses widths $(1808, 1808, 1800, 1800)$. QRL uses four hidden layers of 1184 units, one critic, and joint training.

TD-InfoNCE keeps twin normalized 16-dimensional representations, discount 0.99, actor/critic learning rates $5 \times 10^{-5}/3 \times 10^{-4}$, and target-network update rate 0.005. CRL keeps binary NCE over all pairs and unnormalized 64-dimensional representations scored by a dot product. Its entropy coefficient remains fixed at zero. GCSL discretizes each action dimension into three choices. When the joint action space exceeds 100 choices, it uses the original method's factorized representation, except on AntNavigate, which uses the joint representation. C-Learning uses pure-TD recursive classification, with odds clipped at 20.

The non-QRL baselines use batch size 256 and one learning update per environment step. Each method keeps its own rules for initial data collection and exploration. Algorithm settings remain fixed for all fourteen tasks.

Actor goal inputs. We retain each baseline method's goal inputs, even when they contain more than the success coordinates. During testing, QRL uses artificially constructed complete goal states, following its released implementation. We place the task goal values in their corresponding coordinates and set the remaining coordinates to zero. For Fetch, this construction also copies the desired object position into the gripper-position coordinates $(0, 1, 2)$; success depends only on the object coordinates $(3, 4, 5)$. CRL uses these complete goal vectors on all tasks except AntMaze UMaze, where it uses the XY goal. TD-InfoNCE, C-Learning, and GCSL use the task goal coordinates on non-Fetch tasks. On Fetch, TD-InfoNCE and C-Learning use the complete 25-dimensional goal vector, while GCSL uses its first six coordinates (gripper and object positions). AQRL uses specified-coordinate latents as goal inputs throughout the suite. All methods retain the complete current-state information. These goal inputs describe the baseline evaluated here. QRL also discusses an extension for goals defined as sets of states in its Appendix A; we do not use that extension (Wang et al., 2023).

B.3 Evaluation and checkpoint selection

The main comparison uses 200k–500k interactions per task and five training seeds per method and task. All methods use the same success criteria, interaction budgets, training seeds, checkpoint schedule, and evaluation code. We select each checkpoint by validation performance and test it once with deterministic actions on separate episode seeds.

Table 8: Per-task evaluation schedule for the main comparison. Each setting uses five seeds. Val. and Test give evaluation episode counts. Validation and test use disjoint episode-seed ranges.

Task	Budget	Interval	Val.	Test	Horizon
AntMaze UMaze-v5	400k	20k	100	100	700
AntNavigate-v4	500k	50k	500	1000	1000
FetchPickAndPlace	200k	20k	1000	1000	50
FetchPush	200k	20k	1000	1000	50
FetchSlide	500k	50k	500	1000	50
maze2d-large	200k	20k	1000	1000	800
PandaFlip-v3	400k	20k	200	200	50
PandaPickAndPlace-v3	400k	20k	200	200	50
PandaPickAndPlaceJoints-v3	400k	20k	200	200	50
PandaPushJoints-v3	400k	20k	200	200	50
PandaSlide-v3	400k	20k	200	200	50
PointMaze Medium-v3	200k	20k	100	100	600
Pusher-v4	500k	50k	500	1000	100
reacher_hard	200k	20k	1000	1000	1000

Checkpoint selection prioritizes validation success, then lower mean hitting time, higher mean episode return, and later training step. For this selection criterion, unsuccessful validation episodes receive a hitting time of $H + 1$, where H is the episode horizon. Within each task, all methods use the same validation episode seeds.

C SUPPLEMENTARY RESULTS

C.1 SIX-TASK SEARCH-BUDGET ABLATION

We use the main-table configuration (Table 5), varying only the search budget $K \in \{0, 8, 16, 32\}$. Each setting uses the same five training seeds (1000–1004). At $K = 0$, the sampled target latent is used without search updates.

Budgets and evaluation schedules match Table 8; checkpoint selection follows Appendix B.3. Each reported value is test success at the validation-selected checkpoint, not the maximum validation success or the final training-time validation value. We average the five training seeds and use their sample standard deviation ($\mathrm{ddof} = 1$). Macro averages first average seeds within each task, then weight tasks equally. The $K = 32$ results are the same runs as Table 1.

Table 9 lists the 120 runs for the six ablation tasks.

Table 9: Individual-seed test success (%) for the six search-ablation tasks. Each row compares the same training seed at four search budgets. Means and sample SD appear in Table 2.

Task	Seed	$K = 0$	$K = 8$	$K = 16$	$K = 32$
AntNavigate	1000	94.90	97.30	97.00	96.10
	1001	96.50	97.10	92.90	98.00
	1002	95.10	94.70	98.40	99.20
	1003	96.70	96.60	99.10	98.20
	1004	95.20	97.70	98.00	98.40
FetchPush	1000	90.00	90.30	93.30	95.80
	1001	83.30	90.00	87.90	87.40
	1002	85.80	89.70	89.70	93.30
	1003	93.90	92.00	95.00	91.90
	1004	90.20	92.00	93.20	88.00
PandaPnP-J	1000	42.50	48.50	49.00	49.50
	1001	46.50	46.50	51.50	56.50
	1002	52.50	49.00	41.00	49.50
	1003	53.00	50.50	53.00	55.50
	1004	52.00	49.50	50.00	48.00
PandaPush-J	1000	82.50	94.00	97.00	96.00
	1001	88.00	92.00	92.50	96.50
	1002	90.50	87.50	94.50	92.00
	1003	94.00	94.00	93.00	96.50
	1004	89.00	91.50	92.50	94.50
PandaSlide	1000	48.00	57.50	55.00	51.00
	1001	54.00	55.00	65.50	56.50
	1002	39.50	31.00	45.00	50.50
	1003	40.50	53.50	61.00	65.50
	1004	38.50	45.50	54.00	46.00
Pusher	1000	50.50	45.60	42.00	40.80
	1001	39.70	30.60	35.00	77.60
	1002	33.90	32.40	46.50	48.30
	1003	43.50	40.90	36.20	31.90
	1004	26.00	35.30	39.70	22.10

C.2 COMPLETE LEARNING CURVES AND AUC

Figure 7 extends the four-task selection in Figure 3 to all fourteen tasks. Table 10 gives every method–task AUC with seed variability.

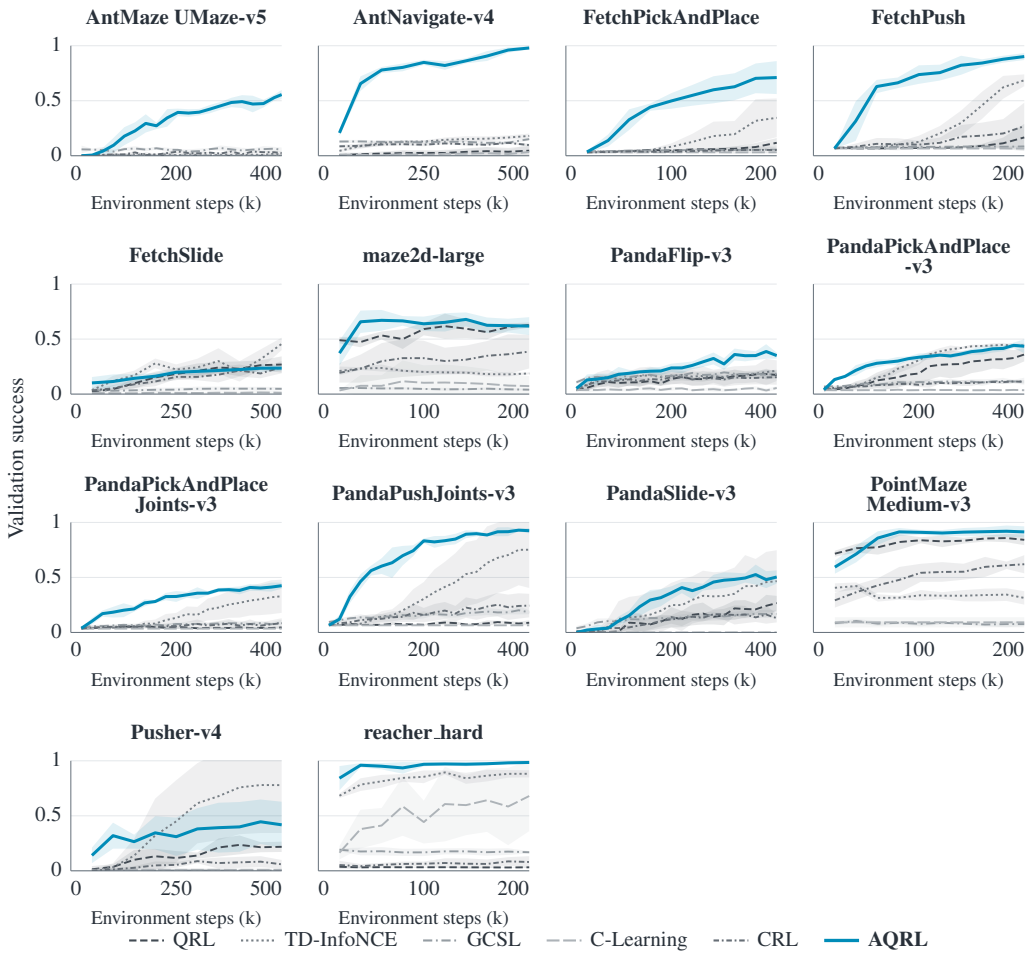


Figure 7: Validation success across fourteen tasks, ordered alphabetically. Lines show the mean and bands $\pm$ one sample standard deviation over five training seeds. Checkpoints are linearly aligned within the seeds' common observed interval, without smoothing or extrapolation. All panels use the full $[0, 1]$ success range; bands extending beyond it are clipped for display. Per-task normalized AUC is reported in Table 10.

Table 10: Normalized validation-success AUC (%) by task over 20%–100% of its interaction budget. Means ± sample SD over five training seeds. Bold marks the highest mean. Task names are abbreviated as in Table 1.

Task	QRL	TD-InfoNCE	GCSL	C-Learning	CRL	AQRL (Ours)
AntMaze-U	0.0±0.0	0.9±0.4	5.6±0.4	0.0±0.0	2.4±0.3	**38.0±3.7**
AntNavigate	3.0±2.7	14.1±1.2	12.6±0.5	1.5±3.4	10.4±0.3	**85.0±2.1**
FetchPnP	6.0±1.2	14.8±8.1	5.0±0.4	3.1±0.2	4.4±2.1	**52.1±6.8**
FetchPush	8.4±1.7	28.3±6.0	8.1±0.4	6.6±0.1	14.5±7.5	**74.2±2.9**
FetchSlide	19.2±4.6	**25.3±3.1**	4.4±1.7	1.1±0.9	15.5±3.3	19.5±1.7
Maze2D-L	57.0±5.2	20.2±0.7	4.7±0.9	9.3±13.6	32.3±14.0	**64.9±3.7**
PandaFlip	13.9±0.9	16.7±7.2	17.5±1.0	4.4±0.6	14.0±5.4	**27.0±2.9**
PandaPnP	23.2±8.4	32.9±1.5	10.7±2.0	3.6±0.0	9.6±1.1	**34.8±1.8**
PandaPnP-J	4.5±0.5	18.1±7.8	7.5±1.3	3.5±0.0	6.4±2.7	**33.5±1.4**
PandaPush-J	7.9±0.7	41.7±19.8	16.5±2.2	6.5±0.0	18.6±7.2	**79.4±2.5**
PandaSlide	14.1±5.0	27.9±20.9	14.1±2.6	0.0±0.0	11.8±3.6	**36.6±2.5**
PointMaze-M	82.6±2.9	33.7±3.7	8.2±0.7	9.2±4.6	52.8±6.5	**89.2±2.0**
Pusher	15.9±4.5	**51.8±31.0**	0.3±0.5	0.9±1.3	6.1±2.5	36.3±16.8
Reacher-H	3.2±0.3	85.3±1.8	17.3±2.8	55.0±23.9	6.6±2.5	**96.5±1.8**

C.3 Training stability and replay coverage

Table 11 complements the AUC results with peak-to-final validation changes and the fraction of successful transitions in replay.

Table 11: Training diagnostics across fourteen tasks (70 runs per method). End/peak averages the final-to-peak validation success ratio over runs with nonzero peaks; Peak–end averages the peak minus final success over all runs. Successful replay is the mean fraction of stored transitions whose next state satisfies the episode goal, measured at training end. Bold marks column maxima, except for Peak–end, where it marks the minimum.

Method	End/peak (%)	Peak–end (pp)	Successful replay (%)
QRL	89.1	2.1	8.0
TD-InfoNCE	86.0	2.6	14.3
GCSL	83.1	1.8	2.4
C-Learning	87.2	**0.8**	4.3
CRL	81.5	2.9	5.1
AQRL (Ours)	**94.6**	3.1	**18.2**

C.4 Individual-seed validation and test results

Table 12 reports the individual runs underlying Tables 1 and 10, with validation peaks, selected checkpoints, and end-of-training validation results. Checkpoints follow the selection rule in Appendix B.3. Tasks and methods follow the main table's order; seed labels identify training runs.

Table 12: Validation and test results for all individual training seeds. Peak val., End val., and Test report Success rate (%); Val. AUC is also scaled to percent. Peak val. is the highest validation success, attained at the selected checkpoint in every run. Step gives its environment-interaction count, with Budget expressing the fraction of the run's interaction budget as a percentage. End val. is validation success at training completion. Val. AUC uses the normalized 20%–100% interval from Table 10. Test is success at the selected checkpoint, as in Table 1; Successes gives successful/total test episodes. Each group specifies the validation episode count n_{val} per checkpoint. Bold marks the highest success and AUC across methods within each task and training seed, with all ties determined from values before rounding.

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
AntMaze UMaze-v5 QRL ($n_{\text{val}} = 100$)							
1000	0.0	385700	96.4	0.0	0.0	0.0	0/100
1001	0.0	345100	86.3	0.0	0.0	0.0	0/100
1002	0.0	40600	10.2	0.0	0.0	0.0	0/100
1003	0.0	263900	66.0	0.0	0.0	0.0	0/100
1004	0.0	365400	91.3	0.0	0.0	0.0	0/100
AntMaze UMaze-v5 TD-InfoNCE ($n_{\text{val}} = 100$)							
1000	3.0	223300	55.8	1.0	0.5	1.0	1/100
1001	4.0	182700	45.7	2.0	1.5	1.0	1/100
1002	4.0	142100	35.5	0.0	0.9	1.0	1/100
1003	2.0	385700	96.4	0.0	0.5	0.0	0/100
1004	2.0	182700	45.7	1.0	1.1	1.0	1/100
AntMaze UMaze-v5 GCSL ($n_{\text{val}} = 100$)							
1000	9.0	284200	71.0	5.0	5.1	0.0	0/100
1001	8.0	142100	35.5	4.0	5.4	2.0	2/100
1002	10.0	20300	5.1	8.0	5.6	5.0	5/100
1003	10.0	365400	91.3	3.0	5.9	8.0	8/100
1004	10.0	304500	76.1	7.0	6.1	8.0	8/100

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			AntMaze UMaze-v5 C-Learning ($n_{\text{val}} = 100$)				
1000	0.0	400000	100.0	0.0	0.0	0.0	0/100
1001	0.0	400000	100.0	0.0	0.0	0.0	0/100
1002	0.0	400000	100.0	0.0	0.0	0.0	0/100
1003	0.0	400000	100.0	0.0	0.0	0.0	0/100
1004	0.0	40600	10.2	0.0	0.0	0.0	0/100
			AntMaze UMaze-v5 CRL ($n_{\text{val}} = 100$)				
1000	7.0	304500	76.1	3.0	2.2	0.0	0/100
1001	5.0	400000	100.0	5.0	2.2	3.0	3/100
1002	5.0	304500	76.1	2.0	2.4	2.0	2/100
1003	8.0	203000	50.7	1.0	2.8	2.0	2/100
1004	6.0	345100	86.3	1.0	2.5	2.0	2/100
			AntMaze UMaze-v5 AQRL (Ours) ($n_{\text{val}} = 100$)				
1000	**54.0**	385700	96.4	**51.0**	**32.2**	**35.0**	**35/100**
1001	**59.0**	400000	100.0	**59.0**	**42.1**	**41.0**	**41/100**
1002	**58.0**	400000	100.0	**58.0**	**37.5**	**46.0**	**46/100**
1003	**55.0**	385700	96.4	**51.0**	**40.1**	**45.0**	**45/100**
1004	**59.0**	400000	100.0	**59.0**	**38.1**	**45.0**	**45/100**
			AntNavigate-v4 QRL ($n_{\text{val}} = 500$)				
1000	7.4	500000	100.0	7.4	5.3	9.5	95/1000
1001	7.8	400000	80.0	7.6	4.6	7.3	73/1000
1002	8.0	500000	100.0	8.0	4.9	7.8	78/1000
1003	0.0	100000	20.0	0.0	0.0	0.0	0/1000
1004	0.0	150000	30.0	0.0	0.0	0.0	0/1000
			AntNavigate-v4 TD-InfoNCE ($n_{\text{val}} = 500$)				
1000	18.2	450000	90.0	17.8	13.6	17.8	178/1000
1001	17.8	450000	90.0	16.0	13.0	18.0	180/1000
1002	20.0	500000	100.0	20.0	15.9	17.9	179/1000
1003	16.6	300000	60.0	15.8	13.2	15.3	153/1000
1004	20.4	500000	100.0	20.4	14.6	20.5	205/1000
			AntNavigate-v4 GCSL ($n_{\text{val}} = 500$)				
1000	16.6	500000	100.0	16.6	12.1	11.7	117/1000
1001	15.6	500000	100.0	15.6	12.9	13.8	138/1000
1002	15.0	500000	100.0	15.0	13.3	10.6	106/1000
1003	15.6	500000	100.0	15.6	12.5	13.2	132/1000
1004	14.2	100000	20.0	12.8	12.1	13.4	134/1000
			AntNavigate-v4 C-Learning ($n_{\text{val}} = 500$)				
1000	17.2	500000	100.0	17.2	7.6	21.3	213/1000
1001	0.0	500000	100.0	0.0	0.0	0.0	0/1000
1002	0.0	500000	100.0	0.0	0.0	0.0	0/1000
1003	0.0	500000	100.0	0.0	0.0	0.0	0/1000
1004	0.0	500000	100.0	0.0	0.0	0.0	0/1000
			AntNavigate-v4 CRL ($n_{\text{val}} = 500$)				
1000	12.6	450000	90.0	11.6	10.7	10.3	103/1000
1001	12.0	250000	50.0	9.0	10.6	10.5	105/1000
1002	12.2	450000	90.0	9.6	10.6	13.6	136/1000
1003	11.8	450000	90.0	8.0	10.2	13.7	137/1000
1004	11.6	350000	70.0	10.2	10.1	11.4	114/1000
			AntNavigate-v4 AQRL (Ours) ($n_{\text{val}} = 500$)				
1000	**95.6**	500000	100.0	**95.6**	**84.6**	**96.1**	**961/1000**
1001	**97.8**	500000	100.0	**97.8**	**81.6**	**98.0**	**980/1000**
1002	**99.2**	500000	100.0	**99.2**	**85.7**	**99.2**	**992/1000**
1003	**98.4**	500000	100.0	**98.4**	**87.2**	**98.2**	**982/1000**
1004	**98.8**	500000	100.0	**98.8**	**85.9**	**98.4**	**984/1000**

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			FetchPickAndPlace QRL ($n_{\text{val}} = 1000$)				
1000	12.5	200000	100.0	12.5	6.4	14.3	143/1000
1001	17.8	200000	100.0	17.8	7.6	19.0	190/1000
1002	6.1	80000	40.0	5.9	4.5	7.8	78/1000
1003	12.2	200000	100.0	12.2	6.2	14.7	147/1000
1004	10.5	200000	100.0	10.5	5.1	10.7	107/1000
			FetchPickAndPlace TD-InfoNCE ($n_{\text{val}} = 1000$)				
1000	53.6	180000	90.0	48.2	24.1	53.2	532/1000
1001	57.1	200000	100.0	57.1	22.9	57.9	579/1000
1002	16.2	200000	100.0	16.2	6.8	16.3	163/1000
1003	20.6	200000	100.0	20.6	8.6	19.6	196/1000
1004	30.3	200000	100.0	30.3	11.7	32.0	320/1000
			FetchPickAndPlace GCSL ($n_{\text{val}} = 1000$)				
1000	7.3	200000	100.0	7.3	5.5	7.1	71/1000
1001	5.6	200000	100.0	5.6	4.8	6.0	60/1000
1002	4.9	200000	100.0	4.9	4.6	6.5	65/1000
1003	6.0	160000	80.0	5.5	5.2	7.8	78/1000
1004	5.7	180000	90.0	5.0	4.8	6.4	64/1000
			FetchPickAndPlace C-Learning ($n_{\text{val}} = 1000$)				
1000	3.3	200000	100.0	3.3	3.1	4.2	42/1000
1001	3.1	200000	100.0	3.1	3.1	4.3	43/1000
1002	3.0	200000	100.0	3.0	3.0	4.4	44/1000
1003	3.4	200000	100.0	3.4	3.4	4.3	43/1000
1004	3.0	200000	100.0	3.0	3.0	4.2	42/1000
			FetchPickAndPlace CRL ($n_{\text{val}} = 1000$)				
1000	3.0	200000	100.0	3.0	3.0	4.2	42/1000
1001	3.0	200000	100.0	3.0	3.0	4.2	42/1000
1002	8.3	180000	90.0	8.0	5.1	9.4	94/1000
1003	10.5	160000	80.0	9.0	7.8	12.1	121/1000
1004	3.0	200000	100.0	3.0	3.0	4.2	42/1000
			FetchPickAndPlace AQRL (Ours) ($n_{\text{val}} = 1000$)				
1000	**71.1**	180000	90.0	**64.6**	**48.1**	**70.0**	**700/1000**
1001	**90.8**	200000	100.0	**90.8**	**63.6**	**91.2**	**912/1000**
1002	**83.6**	180000	90.0	**83.1**	**53.0**	**85.1**	**851/1000**
1003	**57.3**	200000	100.0	**57.3**	**47.0**	**55.4**	**554/1000**
1004	**60.6**	160000	80.0	**59.5**	**49.1**	**59.7**	**597/1000**
			FetchPush QRL ($n_{\text{val}} = 1000$)				
1000	36.7	200000	100.0	36.7	11.2	37.5	375/1000
1001	7.3	180000	90.0	7.0	6.8	7.4	74/1000
1002	20.4	200000	100.0	20.4	8.9	19.3	193/1000
1003	9.0	180000	90.0	8.9	7.7	9.9	99/1000
1004	10.3	200000	100.0	10.3	7.6	9.9	99/1000
			FetchPush TD-InfoNCE ($n_{\text{val}} = 1000$)				
1000	70.4	200000	100.0	70.4	24.1	67.5	675/1000
1001	59.3	200000	100.0	59.3	23.9	55.6	556/1000
1002	80.5	180000	90.0	69.5	38.4	78.6	786/1000
1003	70.8	200000	100.0	70.8	26.9	71.0	710/1000
1004	73.0	200000	100.0	73.0	28.2	72.2	722/1000
			FetchPush GCSL ($n_{\text{val}} = 1000$)				
1000	9.2	180000	90.0	8.8	8.6	10.0	100/1000
1001	9.0	200000	100.0	9.0	8.5	9.7	97/1000
1002	8.5	160000	80.0	8.1	7.9	8.4	84/1000
1003	8.6	120000	60.0	8.0	7.9	8.0	80/1000
1004	8.4	200000	100.0	8.4	7.8	8.8	88/1000

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			FetchPush C-Learning ($n_{\text{val}} = 1000$)				
1000	6.7	200000	100.0	6.7	6.7	7.1	71/1000
1001	6.5	200000	100.0	6.5	6.5	7.0	70/1000
1002	6.4	200000	100.0	6.4	6.4	7.1	71/1000
1003	6.6	200000	100.0	6.6	6.6	7.2	72/1000
1004	6.7	200000	100.0	6.7	6.7	7.3	73/1000
			FetchPush CRL ($n_{\text{val}} = 1000$)				
1000	42.4	200000	100.0	42.4	17.8	41.6	416/1000
1001	31.2	160000	80.0	24.8	19.5	31.6	316/1000
1002	54.2	200000	100.0	54.2	22.4	58.0	580/1000
1003	6.5	200000	100.0	6.5	6.5	7.0	70/1000
1004	6.5	200000	100.0	6.5	6.5	7.0	70/1000
			FetchPush AQRL (Ours) ($n_{\text{val}} = 1000$)				
1000	**94.2**	200000	100.0	**94.2**	**77.0**	**95.8**	**958/1000**
1001	**89.7**	200000	100.0	**89.7**	**77.4**	**87.4**	**874/1000**
1002	**92.6**	200000	100.0	**92.6**	**72.3**	**93.3**	**933/1000**
1003	**91.9**	180000	90.0	**87.1**	**73.6**	**91.9**	**919/1000**
1004	**87.2**	200000	100.0	**87.2**	**70.7**	**88.0**	**880/1000**
			FetchSlide QRL ($n_{\text{val}} = 500$)				
1000	33.0	500000	100.0	33.0	22.1	29.4	294/1000
1001	26.0	350000	70.0	24.2	18.8	21.1	211/1000
1002	35.4	500000	100.0	35.4	**25.2**	35.0	350/1000
1003	26.8	450000	90.0	24.8	15.8	24.1	241/1000
1004	21.6	450000	90.0	17.0	13.9	21.9	219/1000
			FetchSlide TD-InfoNCE ($n_{\text{val}} = 500$)				
1000	**37.4**	500000	100.0	**37.4**	**22.7**	**35.7**	**357/1000**
1001	**52.8**	500000	100.0	**52.8**	**23.9**	**53.2**	**532/1000**
1002	**43.6**	500000	100.0	**43.6**	24.2	**44.3**	**443/1000**
1003	**48.0**	500000	100.0	**48.0**	**25.1**	**44.3**	**443/1000**
1004	**48.4**	350000	70.0	**46.4**	**30.7**	**48.0**	**480/1000**
			FetchSlide GCSL ($n_{\text{val}} = 500$)				
1000	6.0	400000	80.0	5.6	4.6	5.3	53/1000
1001	4.6	450000	90.0	3.6	3.4	3.3	33/1000
1002	6.4	50000	10.0	4.6	4.6	3.6	36/1000
1003	8.0	300000	60.0	7.0	6.8	8.5	85/1000
1004	3.2	500000	100.0	3.2	2.4	2.3	23/1000
			FetchSlide C-Learning ($n_{\text{val}} = 500$)				
1000	0.6	500000	100.0	0.6	0.6	0.3	3/1000
1001	4.6	450000	90.0	3.2	2.7	4.4	44/1000
1002	0.6	500000	100.0	0.6	0.6	0.3	3/1000
1003	1.2	500000	100.0	1.2	1.2	1.6	16/1000
1004	0.6	500000	100.0	0.6	0.6	0.3	3/1000
			FetchSlide CRL ($n_{\text{val}} = 500$)				
1000	30.0	500000	100.0	30.0	15.4	28.6	286/1000
1001	30.2	400000	80.0	25.4	17.0	26.6	266/1000
1002	29.2	500000	100.0	29.2	19.8	31.1	311/1000
1003	20.4	400000	80.0	16.8	14.4	18.4	184/1000
1004	21.0	500000	100.0	21.0	11.0	21.2	212/1000
			FetchSlide AQRL (Ours) ($n_{\text{val}} = 500$)				
1000	24.4	450000	90.0	20.6	18.7	23.7	237/1000
1001	27.8	450000	90.0	26.8	20.9	26.1	261/1000
1002	24.2	450000	90.0	22.2	21.6	24.3	243/1000
1003	23.4	500000	100.0	23.4	17.9	23.8	238/1000
1004	25.0	500000	100.0	25.0	18.3	24.1	241/1000

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			maze2d-large QRL ($n_{\text{val}} = 1000$)				
1000	63.4	200000	100.0	63.4	53.0	64.2	642/1000
1001	66.6	180000	90.0	**66.1**	55.8	65.3	653/1000
1002	60.5	200000	100.0	**60.5**	54.7	60.9	609/1000
1003	**80.9**	120000	60.0	63.3	**66.1**	**79.0**	**790/1000**
1004	63.7	120000	60.0	60.8	55.5	59.8	598/1000
			maze2d-large TD-InfoNCE ($n_{\text{val}} = 1000$)				
1000	25.7	60000	30.0	19.6	20.2	24.0	240/1000
1001	24.4	20000	10.0	19.5	21.2	24.5	245/1000
1002	24.4	40000	20.0	18.1	19.6	23.4	234/1000
1003	28.3	60000	30.0	17.9	19.6	27.8	278/1000
1004	24.6	20000	10.0	18.7	20.4	24.9	249/1000
			maze2d-large GCSL ($n_{\text{val}} = 1000$)				
1000	5.7	40000	20.0	2.9	4.2	5.6	56/1000
1001	6.2	20000	10.0	4.3	4.4	7.3	73/1000
1002	5.3	20000	10.0	4.0	4.3	6.4	64/1000
1003	6.9	80000	40.0	4.8	6.2	6.6	66/1000
1004	5.9	40000	20.0	4.1	4.5	6.1	61/1000
			maze2d-large C-Learning ($n_{\text{val}} = 1000$)				
1000	2.8	200000	100.0	2.8	2.8	2.1	21/1000
1001	2.8	200000	100.0	2.8	2.8	2.1	21/1000
1002	3.6	200000	100.0	3.6	3.6	4.6	46/1000
1003	3.6	200000	100.0	3.6	3.6	4.6	46/1000
1004	45.5	80000	40.0	23.6	33.7	45.8	458/1000
			maze2d-large CRL ($n_{\text{val}} = 1000$)				
1000	30.6	160000	80.0	28.0	25.1	30.7	307/1000
1001	63.3	180000	90.0	55.1	50.8	60.6	606/1000
1002	46.7	100000	50.0	45.6	38.7	47.0	470/1000
1003	46.8	200000	100.0	46.8	33.2	43.1	431/1000
1004	17.6	200000	100.0	17.6	13.8	15.3	153/1000
			maze2d-large AQRL (Ours) ($n_{\text{val}} = 1000$)				
1000	**78.5**	140000	70.0	**63.8**	**70.7**	**75.8**	**758/1000**
1001	**70.0**	120000	60.0	61.9	**63.6**	**70.6**	**706/1000**
1002	**82.0**	60000	30.0	48.8	**60.4**	**78.5**	**785/1000**
1003	68.7	40000	20.0	**63.6**	64.4	68.9	689/1000
1004	**71.1**	200000	100.0	**71.1**	**65.3**	**72.0**	**720/1000**
			PandaFlip-v3 QRL ($n_{\text{val}} = 200$)				
1000	20.5	340000	85.0	18.5	14.9	19.0	38/200
1001	26.0	380000	95.0	18.5	14.7	18.5	37/200
1002	21.0	320000	80.0	13.0	13.9	18.5	37/200
1003	20.0	400000	100.0	20.0	12.8	24.0	48/200
1004	17.0	380000	95.0	17.0	13.3	12.0	24/200
			PandaFlip-v3 TD-InfoNCE ($n_{\text{val}} = 200$)				
1000	8.0	280000	70.0	5.0	4.2	5.0	10/200
1001	30.5	380000	95.0	22.5	18.4	23.5	47/200
1002	26.5	400000	100.0	26.5	18.7	22.0	44/200
1003	26.0	260000	65.0	24.0	19.8	25.5	51/200
1004	31.5	380000	95.0	28.5	22.4	21.0	42/200
			PandaFlip-v3 GCSL ($n_{\text{val}} = 200$)				
1000	23.5	380000	95.0	20.0	16.9	13.0	26/200
1001	23.0	240000	60.0	17.0	17.2	21.5	43/200
1002	22.5	340000	85.0	21.5	19.1	18.0	36/200
1003	24.0	320000	80.0	20.0	17.8	22.5	45/200
1004	23.0	280000	70.0	12.5	16.6	15.0	30/200

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			PandaFlip-v3 C-Learning ($n_{\text{val}} = 200$)				
1000	8.0	280000	70.0	5.0	4.2	5.0	10/200
1001	7.0	200000	50.0	6.5	4.7	4.0	8/200
1002	5.0	80000	20.0	3.5	3.5	3.5	7/200
1003	8.5	160000	40.0	6.0	4.7	6.0	12/200
1004	7.5	360000	90.0	7.0	4.9	5.5	11/200
			PandaFlip-v3 CRL ($n_{\text{val}} = 200$)				
1000	21.5	200000	50.0	16.0	17.0	16.5	33/200
1001	21.0	120000	30.0	16.5	16.1	14.0	28/200
1002	22.0	320000	80.0	16.0	16.3	15.5	31/200
1003	23.5	400000	100.0	23.5	16.3	16.5	33/200
1004	5.5	220000	55.0	4.5	4.3	4.5	9/200
			PandaFlip-v3 AQRL (Ours) ($n_{\text{val}} = 200$)				
1000	**37.0**	280000	70.0	**34.0**	**27.9**	**29.5**	**59/200**
1001	**50.0**	380000	95.0	**39.0**	**31.5**	**45.5**	**91/200**
1002	**37.0**	320000	80.0	**33.0**	**24.7**	**33.5**	**67/200**
1003	**38.0**	320000	80.0	**32.5**	**24.7**	**34.5**	**69/200**
1004	**39.0**	380000	95.0	**36.5**	**26.0**	**32.0**	**64/200**
			PandaPickAndPlace-v3 QRL ($n_{\text{val}} = 200$)				
1000	39.5	400000	100.0	39.5	26.5	42.0	84/200
1001	33.0	400000	100.0	33.0	17.0	29.0	58/200
1002	44.0	400000	100.0	44.0	32.5	41.0	82/200
1003	27.5	400000	100.0	27.5	12.0	29.0	58/200
1004	41.5	260000	65.0	36.0	27.9	46.0	92/200
			PandaPickAndPlace-v3 TD-InfoNCE ($n_{\text{val}} = 200$)				
1000	47.5	360000	90.0	44.5	34.0	52.5	105/200
1001	43.0	360000	90.0	41.0	31.4	47.5	95/200
1002	45.5	340000	85.0	45.0	31.0	52.5	105/200
1003	**45.5**	360000	90.0	**45.5**	**34.3**	**52.0**	**104/200**
1004	**45.5**	400000	100.0	**45.5**	**33.8**	**50.5**	**101/200**
			PandaPickAndPlace-v3 GCSL ($n_{\text{val}} = 200$)				
1000	11.5	380000	95.0	10.0	9.9	9.5	19/200
1001	10.5	260000	65.0	8.0	8.5	13.5	27/200
1002	12.5	360000	90.0	10.5	9.8	12.0	24/200
1003	16.0	400000	100.0	16.0	13.4	14.5	29/200
1004	14.0	360000	90.0	14.0	11.9	10.0	20/200
			PandaPickAndPlace-v3 C-Learning ($n_{\text{val}} = 200$)				
1000	4.5	220000	55.0	3.5	3.6	2.5	5/200
1001	4.5	160000	40.0	3.5	3.7	2.5	5/200
1002	4.0	380000	95.0	3.5	3.6	2.5	5/200
1003	4.5	160000	40.0	3.5	3.7	2.5	5/200
1004	4.0	260000	65.0	3.5	3.6	3.0	6/200
			PandaPickAndPlace-v3 CRL ($n_{\text{val}} = 200$)				
1000	11.5	320000	80.0	10.0	8.4	10.0	20/200
1001	15.5	220000	55.0	10.0	11.1	11.5	23/200
1002	12.0	340000	85.0	11.0	8.7	8.5	17/200
1003	13.5	400000	100.0	13.5	9.7	11.0	22/200
1004	14.0	300000	75.0	11.0	9.9	11.5	23/200
			PandaPickAndPlace-v3 AQRL (Ours) ($n_{\text{val}} = 200$)				
1000	**48.0**	400000	100.0	**48.0**	**37.8**	**57.0**	**114/200**
1001	**45.0**	360000	90.0	**42.5**	**34.1**	**52.0**	**104/200**
1002	**51.0**	400000	100.0	**51.0**	**34.8**	**59.0**	**118/200**
1003	43.5	380000	95.0	34.5	33.7	51.5	103/200
1004	44.5	380000	95.0	41.5	33.6	48.0	96/200

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			PandaPickAndPlaceJoints-v3 QRL ($n_{\text{val}} = 200$)				
1000	6.5	100000	25.0	3.5	4.3	3.5	7/200
1001	7.5	320000	80.0	5.0	5.1	4.0	8/200
1002	8.0	120000	30.0	3.5	5.1	7.0	14/200
1003	7.0	400000	100.0	7.0	4.0	6.0	12/200
1004	7.0	140000	35.0	3.5	4.2	6.0	12/200
			PandaPickAndPlaceJoints-v3 TD-InfoNCE ($n_{\text{val}} = 200$)				
1000	6.0	400000	100.0	6.0	4.3	3.5	7/200
1001	39.0	400000	100.0	39.0	21.4	46.5	93/200
1002	41.0	400000	100.0	**41.0**	23.5	47.5	95/200
1003	40.5	400000	100.0	40.5	19.3	44.5	89/200
1004	38.5	360000	90.0	38.0	22.0	42.5	85/200
			PandaPickAndPlaceJoints-v3 GCSL ($n_{\text{val}} = 200$)				
1000	8.5	60000	15.0	6.5	7.1	6.0	12/200
1001	7.5	260000	65.0	7.5	6.7	8.0	16/200
1002	12.0	400000	100.0	12.0	8.5	8.5	17/200
1003	8.0	400000	100.0	8.0	6.2	4.5	9/200
1004	12.0	300000	75.0	11.5	9.3	8.5	17/200
			PandaPickAndPlaceJoints-v3 C-Learning ($n_{\text{val}} = 200$)				
1000	3.5	400000	100.0	3.5	3.5	2.5	5/200
1001	3.5	400000	100.0	3.5	3.5	2.5	5/200
1002	3.5	400000	100.0	3.5	3.5	2.5	5/200
1003	3.5	400000	100.0	3.5	3.5	2.5	5/200
1004	3.5	400000	100.0	3.5	3.5	2.5	5/200
			PandaPickAndPlaceJoints-v3 CRL ($n_{\text{val}} = 200$)				
1000	12.5	300000	75.0	11.5	8.9	9.0	18/200
1001	3.5	400000	100.0	3.5	3.5	2.5	5/200
1002	3.5	60000	15.0	3.5	3.5	2.5	5/200
1003	13.0	340000	85.0	12.5	8.9	9.0	18/200
1004	11.5	300000	75.0	11.0	7.2	11.0	22/200
			PandaPickAndPlaceJoints-v3 AQRL (Ours) ($n_{\text{val}} = 200$)				
1000	**42.0**	340000	85.0	**41.0**	**32.0**	**49.5**	**99/200**
1001	**44.0**	400000	100.0	**44.0**	**35.4**	**56.5**	**113/200**
1002	**44.0**	380000	95.0	39.5	**33.3**	**49.5**	**99/200**
1003	**46.5**	400000	100.0	**46.5**	**34.4**	**55.5**	**111/200**
1004	**41.5**	400000	100.0	**41.5**	**32.5**	**48.0**	**96/200**
			PandaPushJoints-v3 QRL ($n_{\text{val}} = 200$)				
1000	11.0	260000	65.0	7.5	8.0	12.5	25/200
1001	13.0	240000	60.0	12.5	8.8	12.5	25/200
1002	9.5	340000	85.0	9.5	7.2	11.5	23/200
1003	8.5	220000	55.0	7.0	7.3	11.0	22/200
1004	13.0	320000	80.0	7.5	8.1	11.5	23/200
			PandaPushJoints-v3 TD-InfoNCE ($n_{\text{val}} = 200$)				
1000	87.5	380000	95.0	87.0	43.8	86.0	172/200
1001	18.0	380000	95.0	13.5	8.6	20.5	41/200
1002	85.5	400000	100.0	85.5	43.0	89.5	179/200
1003	95.0	400000	100.0	95.0	52.8	96.0	192/200
1004	**96.0**	380000	95.0	**95.0**	60.1	**97.0**	**194/200**
			PandaPushJoints-v3 GCSL ($n_{\text{val}} = 200$)				
1000	16.5	380000	95.0	16.5	14.1	17.5	35/200
1001	25.5	380000	95.0	20.5	19.2	24.0	48/200
1002	22.0	300000	75.0	19.0	16.3	17.0	34/200
1003	18.0	280000	70.0	16.5	14.8	17.5	35/200
1004	22.5	320000	80.0	19.0	18.2	23.5	47/200

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			PandaPushJoints-v3 C-Learning ($n_{\text{val}} = 200$)				
1000	6.5	400000	100.0	6.5	6.5	9.5	19/200
1001	6.5	400000	100.0	6.5	6.5	9.5	19/200
1002	6.5	400000	100.0	6.5	6.5	9.5	19/200
1003	6.5	400000	100.0	6.5	6.5	9.5	19/200
1004	6.5	400000	100.0	6.5	6.5	9.5	19/200
			PandaPushJoints-v3 CRL ($n_{\text{val}} = 200$)				
1000	34.5	360000	90.0	26.5	20.5	30.0	60/200
1001	27.5	400000	100.0	27.5	20.0	28.0	56/200
1002	35.5	400000	100.0	35.5	20.5	32.5	65/200
1003	36.0	380000	95.0	26.0	25.7	32.5	65/200
1004	6.5	400000	100.0	6.5	6.5	9.5	19/200
			PandaPushJoints-v3 AQRL (Ours) ($n_{\text{val}} = 200$)				
1000	**94.0**	380000	95.0	**89.0**	**76.7**	**96.0**	**192/200**
1001	**95.0**	340000	85.0	**92.0**	**82.5**	**96.5**	**193/200**
1002	**94.5**	280000	70.0	**93.0**	**77.1**	**92.0**	**184/200**
1003	**96.0**	400000	100.0	**96.0**	**80.8**	**96.5**	**193/200**
1004	94.0	380000	95.0	92.5	**80.1**	94.5	189/200
			PandaSlide-v3 QRL ($n_{\text{val}} = 200$)				
1000	31.5	400000	100.0	31.5	18.2	28.5	57/200
1001	40.0	380000	95.0	35.0	18.5	41.5	83/200
1002	25.5	300000	75.0	24.5	15.5	21.0	42/200
1003	28.0	380000	95.0	24.0	6.6	28.0	56/200
1004	22.5	340000	85.0	18.5	12.0	21.0	42/200
			PandaSlide-v3 TD-InfoNCE ($n_{\text{val}} = 200$)				
1000	49.0	400000	100.0	**49.0**	22.7	**52.5**	**105/200**
1001	**66.5**	380000	95.0	**66.5**	**47.5**	**65.0**	**130/200**
1002	**74.0**	340000	85.0	**73.5**	**50.2**	**71.0**	**142/200**
1003	45.0	380000	95.0	42.5	18.4	48.5	97/200
1004	3.5	360000	90.0	1.5	0.5	5.0	10/200
			PandaSlide-v3 GCSL ($n_{\text{val}} = 200$)				
1000	16.0	360000	90.0	14.5	13.8	19.0	38/200
1001	20.5	400000	100.0	20.5	15.5	24.0	48/200
1002	13.0	400000	100.0	13.0	9.9	15.5	31/200
1003	22.0	360000	90.0	20.5	16.9	23.5	47/200
1004	16.5	400000	100.0	16.5	14.4	19.0	38/200
			PandaSlide-v3 C-Learning ($n_{\text{val}} = 200$)				
1000	0.5	260000	65.0	0.0	0.0	0.5	1/200
1001	0.5	180000	45.0	0.0	0.1	0.0	0/200
1002	0.5	380000	95.0	0.0	0.1	0.0	0/200
1003	0.0	120000	30.0	0.0	0.0	0.0	0/200
1004	1.0	40000	10.0	0.0	0.0	1.0	2/200
			PandaSlide-v3 CRL ($n_{\text{val}} = 200$)				
1000	16.0	240000	60.0	9.0	7.0	16.5	33/200
1001	25.5	320000	80.0	15.5	16.2	21.0	42/200
1002	27.0	380000	95.0	11.0	14.1	31.0	62/200
1003	19.0	280000	70.0	11.0	9.4	18.5	37/200
1004	18.5	400000	100.0	18.5	12.4	24.0	48/200
			PandaSlide-v3 AQRL (Ours) ($n_{\text{val}} = 200$)				
1000	**51.0**	300000	75.0	48.5	**36.6**	51.0	102/200
1001	61.0	400000	100.0	61.0	40.5	56.5	113/200
1002	50.0	360000	90.0	46.0	33.5	50.5	101/200
1003	**64.5**	360000	90.0	**47.5**	**36.2**	**65.5**	**131/200**
1004	**54.0**	280000	70.0	**49.0**	**36.0**	**46.0**	**92/200**

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			PointMaze Medium-v3 QRL ($n_{\text{val}} = 100$)				
1000	92.0	81600	40.7	89.0	87.0	89.0	89/100
1001	88.0	183600	91.6	85.0	79.9	85.0	85/100
1002	87.0	163200	81.4	77.0	80.7	86.0	86/100
1003	86.0	102000	50.9	82.0	81.4	84.0	84/100
1004	89.0	122400	61.1	88.0	83.8	87.0	87/100
			PointMaze Medium-v3 TD-InfoNCE ($n_{\text{val}} = 100$)				
1000	47.0	20400	10.2	30.0	35.1	43.0	43/100
1001	43.0	40800	20.4	41.0	38.3	35.0	35/100
1002	43.0	40800	20.4	31.0	33.9	39.0	39/100
1003	39.0	20400	10.2	24.0	28.2	30.0	30/100
1004	42.0	40800	20.4	31.0	32.7	40.0	40/100
			PointMaze Medium-v3 GCSL ($n_{\text{val}} = 100$)				
1000	12.0	183600	91.6	10.0	8.8	9.0	9/100
1001	11.0	40800	20.4	5.0	7.1	6.0	6/100
1002	13.0	40800	20.4	8.0	8.7	15.0	15/100
1003	10.0	40800	20.4	8.0	8.0	8.0	8/100
1004	11.0	20400	10.2	7.0	8.2	6.0	6/100
			PointMaze Medium-v3 C-Learning ($n_{\text{val}} = 100$)				
1000	9.0	200400	100.0	9.0	9.0	7.0	7/100
1001	9.0	200400	100.0	9.0	9.0	7.0	7/100
1002	9.0	200400	100.0	9.0	9.0	7.0	7/100
1003	3.0	200400	100.0	3.0	3.0	8.0	8/100
1004	16.0	200400	100.0	16.0	16.0	11.0	11/100
			PointMaze Medium-v3 CRL ($n_{\text{val}} = 100$)				
1000	57.0	183600	91.6	53.0	44.8	56.0	56/100
1001	67.0	200400	100.0	67.0	56.7	63.0	63/100
1002	63.0	163200	81.4	53.0	48.8	57.0	57/100
1003	69.0	200400	100.0	69.0	52.5	57.0	57/100
1004	72.0	163200	81.4	68.0	61.5	70.0	70/100
			PointMaze Medium-v3 AQRL (Ours) ($n_{\text{val}} = 100$)				
1000	**95.0**	183600	91.6	**92.0**	**88.2**	**94.0**	**94/100**
1001	**96.0**	183600	91.6	**96.0**	**88.9**	**90.0**	**90/100**
1002	**96.0**	200400	100.0	**96.0**	**92.3**	**97.0**	**97/100**
1003	**94.0**	81600	40.7	**84.0**	**87.1**	**91.0**	**91/100**
1004	**93.0**	163200	81.4	**89.0**	**89.6**	**91.0**	**91/100**
			Pusher-v4 QRL ($n_{\text{val}} = 500$)				
1000	21.4	500000	100.0	21.4	10.3	20.2	202/1000
1001	28.8	400000	80.0	23.2	20.1	25.9	259/1000
1002	28.4	500000	100.0	28.4	17.4	20.7	207/1000
1003	36.2	400000	80.0	20.6	19.8	38.7	387/1000
1004	22.6	450000	90.0	15.4	12.2	18.6	186/1000
			Pusher-v4 TD-InfoNCE ($n_{\text{val}} = 500$)				
1000	**96.4**	500000	100.0	**96.4**	**48.6**	**96.4**	**964/1000**
1001	**100.0**	400000	80.0	**97.6**	**82.5**	**100.0**	**1000/1000**
1002	**98.4**	450000	90.0	**97.8**	**69.0**	**98.8**	**988/1000**
1003	**97.6**	500000	100.0	**97.6**	**57.6**	**98.1**	**981/1000**
1004	6.0	150000	30.0	0.0	1.2	3.2	32/1000
			Pusher-v4 GCSL ($n_{\text{val}} = 500$)				
1000	0.2	300000	60.0	0.0	0.0	0.1	1/1000
1001	0.0	400000	80.0	0.0	0.0	0.0	0/1000
1002	0.0	50000	10.0	0.0	0.0	0.0	0/1000
1003	0.0	200000	40.0	0.0	0.0	0.0	0/1000
1004	1.4	250000	50.0	1.2	1.2	0.5	5/1000

Continued on next page

Table 12: Individual-seed validation and test results (continued).

Seed	Peak val.	Step	Budget (%)	End val.	Val. AUC	Test	Successes
			Pusher-v4 C-Learning ($n_{\text{val}} = 500$)				
1000	0.0	500000	100.0	0.0	0.0	0.0	0/1000
1001	1.6	500000	100.0	1.6	1.6	1.6	16/1000
1002	2.8	500000	100.0	2.8	2.8	1.9	19/1000
1003	0.0	500000	100.0	0.0	0.0	0.0	0/1000
1004	0.0	500000	100.0	0.0	0.0	0.0	0/1000
			Pusher-v4 CRL ($n_{\text{val}} = 500$)				
1000	10.8	300000	60.0	3.2	5.0	8.8	88/1000
1001	17.0	450000	90.0	12.8	10.4	18.8	188/1000
1002	6.8	400000	80.0	5.4	4.2	6.3	63/1000
1003	10.2	300000	60.0	3.2	5.3	9.3	93/1000
1004	9.0	300000	60.0	4.8	5.4	7.2	72/1000
			Pusher-v4 AQRL (Ours) ($n_{\text{val}} = 500$)				
1000	44.0	450000	90.0	26.8	32.7	40.8	408/1000
1001	79.6	400000	80.0	75.8	65.3	77.6	776/1000
1002	45.2	500000	100.0	45.2	32.8	48.3	483/1000
1003	39.8	450000	90.0	38.4	29.6	31.9	319/1000
1004	**24.4**	400000	80.0	**22.8**	**21.4**	**22.1**	**221/1000**
			reacher_hard QRL ($n_{\text{val}} = 1000$)				
1000	4.1	80000	40.0	3.7	3.7	4.5	45/1000
1001	3.6	60000	30.0	2.9	3.0	3.9	39/1000
1002	3.2	180000	90.0	3.1	2.9	3.3	33/1000
1003	4.9	20000	10.0	3.5	3.3	4.9	49/1000
1004	3.2	20000	10.0	2.7	2.9	4.0	40/1000
			reacher_hard TD-InfoNCE ($n_{\text{val}} = 1000$)				
1000	89.5	80000	40.0	87.9	85.1	90.0	900/1000
1001	91.9	180000	90.0	91.7	85.7	90.8	908/1000
1002	91.5	160000	80.0	89.7	87.5	88.3	883/1000
1003	90.9	160000	80.0	82.9	85.9	92.1	921/1000
1004	91.2	120000	60.0	88.7	82.5	91.2	912/1000
			reacher_hard GCSL ($n_{\text{val}} = 1000$)				
1000	20.2	100000	50.0	17.3	17.7	15.9	159/1000
1001	19.4	20000	10.0	15.1	12.9	21.1	211/1000
1002	21.9	60000	30.0	16.7	20.6	22.1	221/1000
1003	21.0	60000	30.0	18.0	17.7	19.6	196/1000
1004	21.8	40000	20.0	17.2	17.4	25.0	250/1000
			reacher_hard C-Learning ($n_{\text{val}} = 1000$)				
1000	92.7	200000	100.0	92.7	69.7	93.9	939/1000
1001	74.0	120000	60.0	67.3	57.9	73.1	731/1000
1002	19.1	20000	10.0	13.3	13.3	20.1	201/1000
1003	81.3	200000	100.0	81.3	62.7	81.4	814/1000
1004	94.2	180000	90.0	85.3	71.3	95.4	954/1000
			reacher_hard CRL ($n_{\text{val}} = 1000$)				
1000	10.1	180000	90.0	3.0	5.1	11.4	114/1000
1001	15.7	200000	100.0	15.7	11.0	15.9	159/1000
1002	8.8	200000	100.0	8.8	5.6	9.3	93/1000
1003	8.1	80000	40.0	2.9	5.1	8.7	87/1000
1004	9.6	200000	100.0	9.6	6.1	10.6	106/1000
			reacher_hard AQRL (Ours) ($n_{\text{val}} = 1000$)				
1000	**98.0**	180000	90.0	**97.7**	**95.8**	**97.7**	**977/1000**
1001	**99.7**	100000	50.0	**99.6**	**98.9**	**100.0**	**1000/1000**
1002	**97.9**	200000	100.0	**97.9**	**94.5**	**97.6**	**976/1000**
1003	**98.5**	180000	90.0	**98.5**	**95.4**	**98.7**	**987/1000**
1004	**99.1**	80000	40.0	**98.7**	**98.0**	**98.4**	**984/1000**

C.5 BASELINE REPRODUCTION CHECKS

We checked the TD-InfoNCE, CRL, GCSL, and C-Learning implementations on selected tasks from their source task suites. Table 13 reports the results of these 20 checks. Success rates vary across tasks, with strong performance on several tasks for each method. These results support the validity of the baseline implementations. They are single-seed implementation checks, not estimates of variability across training seeds or a claim of numerical agreement with every original-paper result.

Table 13: Baseline reproduction checks on selected source tasks. Success rate (%). Final val. and peak val. use 50 episodes per checkpoint; test uses 200 episodes at the validation-selected checkpoint. Each row uses seed 1000. Budgets count environment interactions (M = million).

Method	Task	Budget	Final val.	Peak val.	Test
TD-InfoNCE	FetchPickAndPlace	1M	88.0	94.0	97.0
TD-InfoNCE	FetchPush	1M	92.0	100.0	96.0
TD-InfoNCE	FetchReach	0.5M	82.0	100.0	100.0
TD-InfoNCE	FetchSlide	1M	56.0	66.0	55.5
CRL	AntUMaze	10M	54.0	64.0	50.0
CRL	FetchPush	3M	78.0	92.0	78.0
CRL	FetchReach	0.2M	92.0	100.0	100.0
CRL	PointSmall	0.1M	100.0	100.0	99.5
CRL	SawyerBin	3M	38.0	46.0	37.5
CRL	SawyerPush	3M	92.0	96.0	93.0
GCSL	Claw	1M	76.0	76.0	73.0
GCSL	FourRooms	0.2M	72.0	76.0	78.5
GCSL	LunarLander	0.2M	72.0	82.0	66.5
GCSL	SawyerDoor	1M	100.0	100.0	100.0
GCSL	SawyerPush	1M	76.0	76.0	76.5
C-Learning	Finger	3M	82.0	98.0	95.5
C-Learning	SawyerDrawer	3M	70.0	96.0	96.0
C-Learning	SawyerPush	3M	16.0	38.0	26.5
C-Learning	SawyerReach	3M	100.0	100.0	100.0
C-Learning	SawyerWindow	3M	64.0	94.0	91.0