DIRECTION-CONDITIONED POLICIES FOR ONLINE GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Contrastive Reinforcement Learning (CRL) learns representations that estimate goal reachability, yet its policy remains conditioned on raw goals and therefore does not directly exploit the geometry encoded by its critic. We introduce Direction-Conditioned Policies (DCP), a method built around a small modification to CRL: DCP selects previously visited states as waypoints during online training and conditions the policy on their direction and distance in representation space. At deployment, DCP applies the same interface directly to the final goal, requiring neither waypoint selection nor planning. Across nine navigation and manipulation tasks, DCP attains higher final success rates than CRL on seven tasks and spends more time near the goal on seven. Controlled maze experiments further show that DCP captures shortest-path geometry more accurately and that the supplied direction causally influences the actor's behavior. We identify waypoint coverage and ranking as limits to exploration, and show that learned candidate generation improves goal reaching in two controlled mazes.

1 INTRODUCTION

When learning to achieve goals through online interaction, an agent has to discover useful behavior before it has much successful experience to draw on. Contrastive reinforcement learning (CRL; Eysenbach et al., 2022) makes use of the states the agent reaches along the way. Its critic learns to distinguish achieved future goals from other goals, and the policy learns to choose actions that receive a higher critic score. This gives the agent a learning signal even when reaching the requested goal is rare. Recent work has made this approach increasingly effective through stabilization techniques, accelerated simulation, deeper networks, and action-sequence conditioning (Zheng et al., 2023; Bortkiewicz et al., 2025; Wang et al., 2025; Korniak et al., 2026).

However, the policy still receives the goal in raw coordinates. Consider a goal on the opposite side of a maze wall: the coordinate difference points through the wall, even though reaching the goal requires moving away from it first. The critic's representation is learned from trajectories and can capture this route structure. This raises a simple question: can we use that representation to express the goal to the policy? Doing so would give the policy direct access to a geometric relation that the critic already learns. The difficulty is that the representation itself is being learned online. Early in training, a distant goal may lie outside the agent's experience, making its learned direction unreliable. We therefore consider two parts of the problem together: how to express a target to the policy, and which targets to pursue while collecting experience. Previously visited states are natural candidate waypoints (Eysenbach et al., 2019; Ecoffet et al., 2021). Scoring these candidates against the final goal yields intermediate targets whose representations have been trained on collected experience.

We propose Direction-Conditioned Policies (DCP), a method built around a small change to the CRL actor. We encode the current achieved goal and the target using the critic's goal encoder, then supply the normalized displacement and its magnitude to the actor, along with the full physical state. During collection, the target is a scored visited waypoint. During hindsight updates, it is a future achieved state. At deployment, it is the requested final goal. In this way, the same policy input supports all three stages, and the waypoint pool and scorer can be removed at deployment. Figure 1 illustrates these inputs.

Direction-conditioned policies and embedding differences have been used in skill learning and offline goal reaching (Park et al., 2024b;a; Devin et al., 2019). We investigate their use within online CRL,

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.

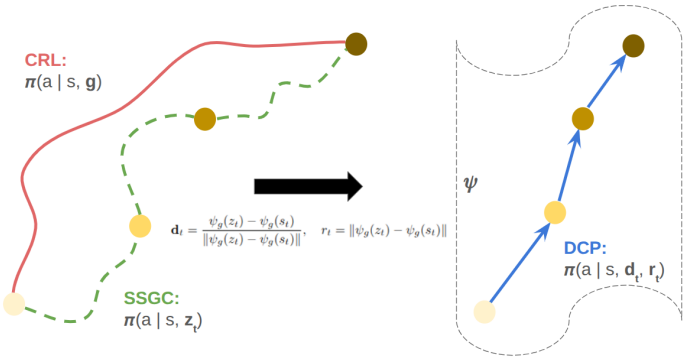


Figure 1: Schematic of the policy inputs. CRL uses the raw goal, SSGC uses a selected waypoint, and DCP uses the waypoint's direction and distance in representation space. At deployment, DCP replaces the waypoint with the final goal. Here $\psi(s)$ abbreviates the encoding of the state's achieved-goal coordinates.

where the policy and representation develop together through the agent's own experience. Our contributions are:

- An online CRL method that uses the critic's goal representation to condition the actor on direction and distance, with deployment requiring only the actor and goal encoder.
- A five-seed comparison across nine tasks, with higher final mean success and time near goal than CRL on seven tasks each.
- Controlled experiments that test route geometry, the actor's use of direction, and waypoint selection, including improvements from learned generation in restrictive mazes.

These results also reveal a remaining difficulty: improving selected waypoints alone is insufficient to address a more disconnected, dynamic loco-manipulation task like AntSoccer.

Section 2 discusses related work, Section 3 describes DCP, Section 4 presents the experiments, and Section 5 concludes. The appendices provide implementation details and additional analyses.

2 Related Work

Contrastive representations for control. CRL derives a goal-reaching objective from contrastive prediction of future states (Eysenbach et al., 2022). JaxGCRL makes large online comparisons practical through accelerated simulation (Bortkiewicz et al., 2025). Depth scaling (Wang et al., 2025) and action-sequence conditioning (Korniak et al., 2026) demonstrate that changes to architecture or critic inputs can substantially alter the capabilities learned with this objective. Single-goal CRL also demonstrates directed exploration before the first successful trial, without intermediate subgoals (Liu et al., 2025). We ask how much of the learned representation can be reused by the policy itself. DCP supplies relative goal-encoder features to the actor while retaining the baseline network depth and single-action policy.

Direction-conditioned skills and goal representations. HILP learns Hilbert representations and directional policies from offline data (Park et al., 2024a); METRA learns metric-aware representations and skills through online unsupervised exploration (Park et al., 2024b). Plan arithmetic uses compositional differences and sums in learned plan representations (Devin et al., 2019). Dual Goal Representations encode goals through temporal distances from other states and improve offline goal reaching (Park et al., 2026). Relative targets also appear in FeUdal Networks, whose worker follows latent directions from a manager (Vezhnevets et al., 2017), HIRO's state-offset subgoals (Nachum et al., 2018), LSD's latent skill directions (Park et al., 2022), and HIQL's latent subgoal actions (Park et al., 2023); learned temporal distances (Hartikainen et al., 2020; Myers et al., 2024) and quasimetrics (Wang et al., 2023; Liu et al., 2023; Myers et al., 2025b) supply the underlying geometry.

We bring learned goal relations to an online contrastive actor–critic, retaining its actor objective and learning the representation from the experience the direction-conditioned policy collects.

Interpolation and waypoint substitution. Contrastive representations can support inference over intermediate states under specific modeling assumptions (Eysenbach et al., 2024). Planning invariance examines when an on-path waypoint and a final goal induce the same action (Myers et al., 2025a). It arises for DCP, which trains on achieved targets but executes toward the final goal, and for flat policies bootstrapped from subgoal-conditioned ones (Zhou & Kao, 2025). We test whether on-path substitution preserves the action and how reliably the scorer selects such waypoints.

3 METHODOLOGY

3.1 CONTRASTIVE GOAL-CONDITIONED LEARNING

Let $s \in \mathcal{S}$ denote the full state, $a \in \mathcal{A}$ an action, and $g \in \mathcal{G}$ a goal. A known projection $q : \mathcal{S} \to \mathcal{G}$ extracts the achieved-goal coordinates, such as the robot's position in navigation or the object's position in manipulation. Success occurs when $q(s)$ lies within the task's goal region. CRL learns a state–action encoder $\phi_\theta(s, a)$ and a goal encoder $\psi_\theta(g)$, both mapping to $\mathbb{R}^k$. We use the JaxGCRL implementation (Bortkiewicz et al., 2025) with negative Euclidean-distance energy,

$$f_\theta(s, a, g) = -\left\|\phi_\theta(s, a) - \psi_\theta(g)\right\|_2 . \tag{1}$$

For a minibatch of state–action pairs and their sampled future achieved goals, write $\ell_{ij} = f_\theta(s_i, a_i, g_j^+)$. Backward InfoNCE distinguishes the matching state–action pair for each goal:

$$\mathcal{L}_{\mathrm{NCE}}(\theta) = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\ell_{ii})}{\sum_{j=1}^{B} \exp(\ell_{ji})}. \tag{2}$$

We retain the baseline's log-sum-exp regularization, replay sampling, and entropy-regularized actor objective. Appendix A gives implementation details.

3.2 A RELATIVE GOAL INPUT FOR THE ACTOR

We want the actor to receive the target relative to the current state in the space learned by the critic. We use the same goal encoder for both positions. For a target $y \in \mathcal{G}$, define

$$v_\theta(s, y) = \psi_\theta(y) - \psi_\theta(q(s)), \qquad r_\theta(s, y) = \left\|v_\theta(s, y)\right\|_2 , \qquad d_\theta(s, y) = \frac{v_\theta(s, y)}{r_\theta(s, y) + \epsilon}, \tag{3}$$

where $\epsilon = 10^{-8}$ prevents division by zero. DCP replaces the raw-goal actor input with $[s, d_\theta(s, y), r_\theta(s, y)]$. The direction specifies where the target lies in the learned space, and the radius gives its distance. We retain the full physical state because the action also depends on velocities and body configuration. For example, moving an object toward the same goal requires a different approach depending on which side of the object the robot occupies.

For nonzero displacement, direction and radius jointly recover the displacement up to numerical stabilization. A waypoint and the final goal can share a direction while differing in distance; the actor receives both quantities.

3.3 COLLECTION, LEARNING, AND DEPLOYMENT

Collecting experience with visited targets. To provide collection targets as the representation develops, we maintain a pool $\mathcal{P}$ of $P = 512$ states drawn from recently collected trajectories. Every 25 environment steps, it samples $N = 32$ candidate indices uniformly with replacement and selects

$$z^* \in \arg\max_{z \in \mathcal{C} \subseteq \mathcal{P}} \psi_\theta(q(z))^\top \psi_\theta(g). \tag{4}$$

The target $y = q(z^*)$ is held until the next refresh, while its direction relative to the current state is recomputed at each action. The inner product ranks candidates using their embedding alignment and magnitude relative to the final goal. This is a simple way to reuse the learned representation for selection. The candidates come from visited states; whether the policy can reach a selected state from its current position is examined in our experiments.

Table 1: Targets used for collection, actor updates, and deployment. Each stage constructs the same direction-and-distance input from its target.

Stage	Target y	Policy input
Online collection	Scored visited waypoint $q(z^*)$	$[s, d_\theta(s,y), r_\theta(s,y)]$
Actor update	Achieved future state $q(s_{t+\Delta})$	$[s, d_\theta(s,y), r_\theta(s,y)]$
Deployment	Requested final goal g	$[s, d_\theta(s,g), r_\theta(s,g)]$

Learning from achieved future states. For learning, each replayed trajectory supplies examples of goals that were actually achieved. We sample a future state and use $y = q(s_{t+\Delta})$ as the actor's target. With $c_\theta(s,y) = [d_\theta(s,y), r_\theta(s,y)]$, the actor minimizes

$$\mathcal{L}_\pi(\eta) = \mathbb{E}\left[\alpha \log \pi_\eta(a \mid s, c_\theta(s,y)) - f_\theta(s,a,y)\right], \quad a \sim \pi_\eta(\cdot \mid s, c_\theta(s,y)). \tag{5}$$

The actor loss updates η with the encoder parameters held fixed. The encoders learn through the contrastive objective. As the policy changes the trajectories it collects, it also changes the data from which the representation learns. The entropy coefficient is learned as in the baseline (Haarnoja et al., 2018).

Direct execution at deployment. At evaluation, we encode the requested final goal and use it directly as y in Equation 3. The actor then executes toward this goal at every step. This removes the pool and scorer, leaving the goal encoder and actor. Table 1 summarizes the three stages. Section 4.5 measures how waypoint-to-goal substitution affects the actor's output.

4 Experiments

4.1 Does DCP improve online goal reaching?

Figure 2 shows all nine tasks. The Ant and Humanoid navigation tasks require moving the robot to a target position around walls. AntPush additionally requires moving an obstructing block to reach the target position. AntSoccer requires moving a ball to the target. The three Pusher tasks require a robot arm to move a cylinder and differ in target placement. Appendix B gives their goal distributions, success conditions, and training budgets.

We evaluate DCP, CRL, and an ablation over CRL, which we term Self-Subgoal Conditioning (SSGC), on nine Brax-based navigation and manipulation tasks (Freeman et al., 2021; Bortkiewicz et al., 2025). SSGC uses the same waypoint selection as DCP and supplies raw waypoint coordinates to the actor. All methods share task-specific interaction budgets, critic objectives, and principal optimization settings. DCP's larger input increases actor parameters by 11–21%, entirely in the first layer. Hidden widths, depths, action heads, and critic parameter counts are unchanged; the total actor–critic increase is 4.15–6.40% (Appendix A.3).

Each method is trained with five seeds. Episodes last 1,000 steps. AntPush, PusherHard, and Humanoid U-Maze use 100M training steps; the other tasks use 50M. We report *success rate*, the percentage of evaluation episodes entering the goal region at least once, and *time near goal*, the mean number of steps spent inside it. We use both because a policy can reach the goal and then move away from it. Final values refer to the last checkpoint. Figure 3 gives the learning curves; Appendix B specifies task settings and Appendix G reports intermediate checkpoints.

DCP improves final mean success and time near goal over CRL on seven tasks each (Table 2). The largest absolute success improvements occur on PusherEasy ($+51.0$ percentage points), PusherHard Far ($+34.9$), and PusherHard ($+18.9$). DCP also improves success on all four navigation tasks, although Humanoid U-Maze remains difficult and variable across seeds. Compared with SSGC, DCP has higher final mean success on seven tasks.

Looking at both metrics makes the task-level differences clearer. On Ant U-Maze, DCP reaches the goal in more episodes but spends fewer steps there than CRL. On AntPush, its final success is lower while its time near goal is higher; SSGC has the largest final means on both metrics. AntSoccer is a failure case for DCP relative to CRL on both measures.

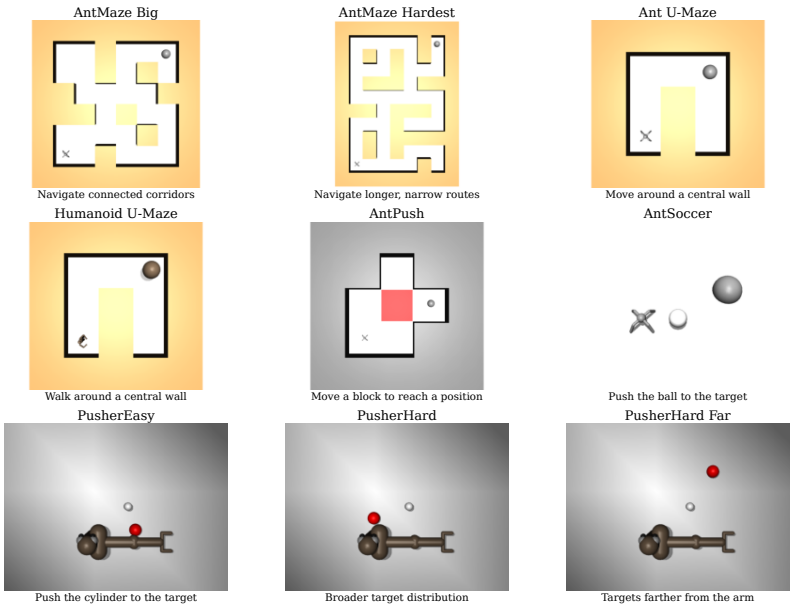


Figure 2: The nine benchmark tasks, rendered from their simulation models. The three Pusher tasks share the same arm and cylinder; their target distributions differ. Eight tasks follow JaxGCRL, while PusherHard Far changes PusherHard's target distribution.

Table 2: Final evaluation across nine tasks, five seeds per method. Entries are means with reported 95% confidence-interval half-widths. Bold marks the largest mean in each task and metric. Time near goal is measured in steps out of 1,000.

	Success rate (%)			Time near goal		
Task	CRL	SSGC	DCP	CRL	SSGC	DCP
AntMaze Big	18.3±6.5	23.3±3.0	**29.9**±2.9	89.2±28.0	99.8±9.0	**115.3**±14.6
AntMaze Hardest	12.2±5.6	14.2±8.0	**16.1**±2.3	48.1±23.7	**66.0**±41.4	63.2±13.3
Ant U-Maze	40.8±10.2	49.8±4.2	**55.0**±13.3	**261.3**±77.2	252.7±52.1	238.1±76.2
Humanoid U-Maze	0.0±0.0	0.0±0.0	**8.4**±9.6	0.0±0.0	0.0±0.0	**3.6**±4.5
AntPush	45.6±24.1	**51.4**±15.5	42.7±12.6	143.4±105.6	**199.1**±93.3	181.7±95.6
PusherEasy	4.4±4.1	34.1±38.5	**55.4**±9.6	26.2±8.4	180.7±183.4	**297.8**±90.9
PusherHard	28.8±32.4	27.8±30.7	**47.7**±5.8	125.7±136.7	170.5±186.0	**280.9**±28.5
PusherHard Far	12.5±24.0	9.0±15.1	**47.4**±5.7	48.4±80.4	49.0±82.3	**260.7**±66.8
AntSoccer	**22.6**±3.4	14.1±3.6	12.9±4.3	**76.4**±7.7	48.2±31.9	42.4±22.1

4.2 What do scoring and direction conditioning contribute?

DCP changes both the collection target and the actor input. To understand their individual effects, we compare all four combinations: CRL, direct-goal direction conditioning, SSGC, and DCP. The actor receives either raw coordinates or direction and distance, and collection uses either the final goal or a scored waypoint. We run each combination with five seeds. All variants train for 10M steps in an open room and a U-maze. Within each actor family, scoring comparisons match architecture and random-number handling; direct-goal DCP retains the scoring computation but ignores its selected target. Table 3 reports mean evaluation success over the training run, summarizing learning over the full training budget.

Direction conditioning without waypoint selection has the largest mean in both settings. Scoring reduces open-room performance in every seed for both actor inputs; adding scoring to the direction-conditioned policy changes mean success by -0.105, with paired 95% interval $[-0.141, -0.068]$. In the U-maze, the corresponding difference is -0.032 with interval $[-0.125, 0.061]$. Here, adding waypoint selection can slow learning. Actor parameters increase from 68,612 to 84,740 (23.5%); the

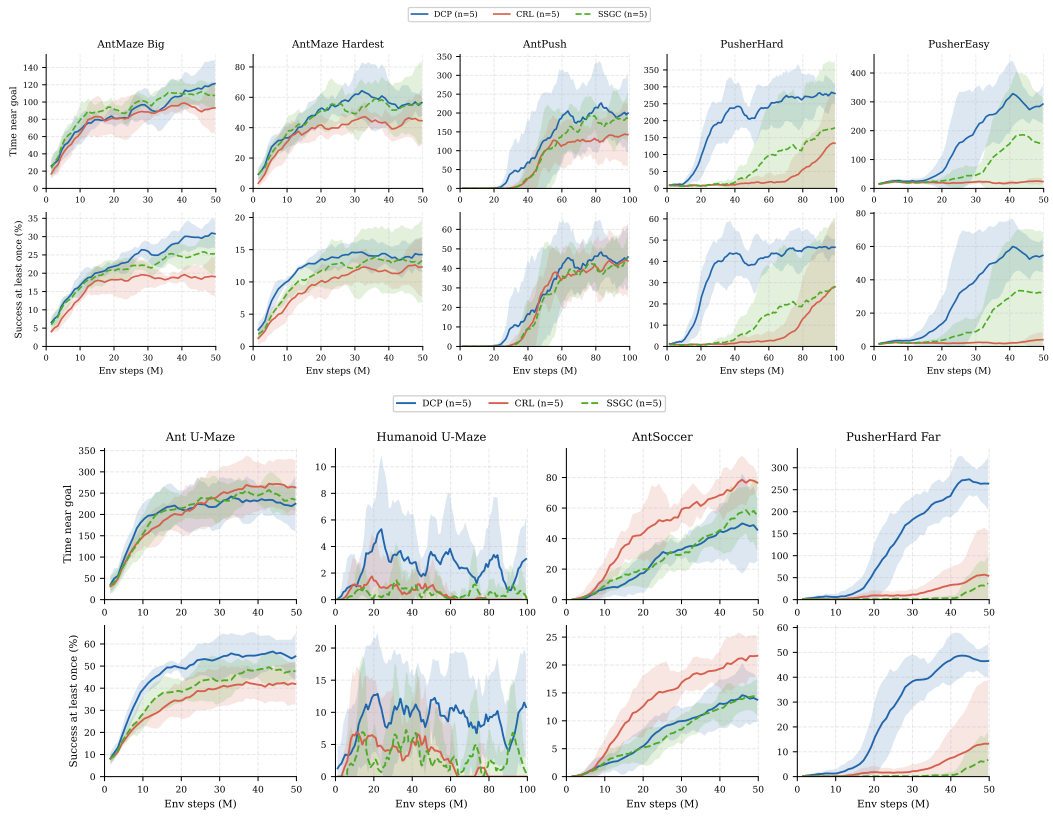


Figure 3: Online learning across all nine tasks. Each task shows time near goal (upper panel) and success at least once (lower panel). Lines are means over five seeds; shading denotes 95% Student-t confidence intervals. Budgets are matched across methods within each task. The comparison includes AntSoccer, where CRL outperforms DCP, and Ant U-Maze, where the final two metrics favor different methods.

Table 3: Separating collection targets from actor inputs. Mean success over 20 evaluation checkpoints, averaged over five seeds; $\pm$ denotes a 95% Student-t confidence-interval half-width across seeds. Higher is better.

Method	Collection target	Open room	U-maze
CRL	Final goal	0.956 ± 0.037	0.846 ± 0.153
Direct direction conditioning	Final goal	0.974 ± 0.010	0.885 ± 0.059
SSGC	Scored waypoint	0.866 ± 0.036	0.851 ± 0.040
DCP	Scored waypoint	0.869 ± 0.033	0.854 ± 0.066

benchmark actors have larger state inputs and action heads, making the same added input weights a smaller fraction of their parameters. We next examine the learned geometry and the actor's use of direction.

4.3 Does the representation capture obstacle-respecting routes?

Our motivation relies on the representation learning something useful about the routes an agent can take. We examine this directly in a planar point-mass environment where we can compute shortest paths around the walls. States include position and velocity, and actions are continuous forces. Appendix C describes all controlled environments and shows their layouts. A weighted occupancy graph gives exact shortest paths on a fixed spatial discretization, with diagonal edges allowed only when collision-free. We evaluate 100,128 directed state pairs and separately inspect 2,336 pairs

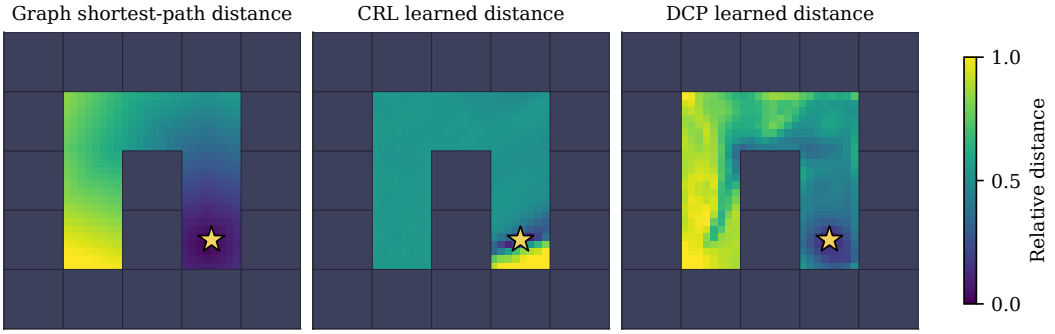


Figure 4: Distance to the starred goal in the controlled U-maze: graph shortest-path distance (left), and learned representation distances for CRL and DCP after 10M steps (middle, right; seed 42). Each panel is scaled by its own 98th-percentile distance. The color scale shows relative distance within each panel.

Table 4: Representation geometry after 10M online steps. Means $\pm$ sample standard deviations over five independently trained encoders. Distances are computed from the goal encoder; local descent follows the negative spatial gradient of the learned distance.

U-maze measure	CRL	DCP
Spearman correlation with graph distance	0.471 ± 0.031	0.756 ± 0.041
Spearman correlation with Euclidean distance	0.456 ± 0.033	0.690 ± 0.034
Graph-correct local descent (%)	64.7 ± 10.1	70.9 ± 4.5

where straight-to-goal motion conflicts with a graph-optimal first move. The graph provides a spatial reference for evaluating the learned representations.

In a five-seed representation study, DCP's goal-embedding distances correlate more strongly with U-maze graph distances than CRL's (Table 4). Figure 4 compares the learned fields with graph distance for one seed; Appendix F shows the individual-seed results and AntMaze examples. On the pairs where moving directly toward the goal conflicts with the shortest route, the negative spatial gradient of the learned distance points toward a graph-optimal first move on 70.9% of pairs for DCP and 64.7% for CRL, averaged across seeds. DCP's open-room graph and Euclidean correlations differ by less than 0.001 in mean, while its U-maze graph correlation exceeds its Euclidean correlation by 0.065. These measurements show that the representation captures the obstacle's effect on route length. Each method learns its representation from its own trajectories. We also find that better distance ordering alone is insufficient to explain control performance: mean U-maze success over training is similar for DCP and CRL (0.847 and 0.846), and CRL performs better in the open room. We therefore next examine the actor's response to the learned direction.

4.4 Does the actor use the supplied direction?

If the actor uses the supplied direction to choose its actions, changing that direction should change how it reaches the goal. We test this with a fixed trained policy, keeping the encoder, distance input, initial conditions, and evaluation procedure unchanged. We first use a differential-drive U-maze with persistent velocity, forward thrust, and yaw torque; the robot cannot command arbitrary lateral force. Across three policies, 42 directed start–goal pairs, and four initial headings, the learned direction yields 45.8% success. A reversed direction reduces success to 0.2%, and wrong-goal, orthogonal, or zero directions also substantially reduce success (Figure 5). **These interventions show that the learned policies use the supplied direction to reach the goal.**

Manipulation adds another difficulty: the robot has to make the right contact before it can move the object toward the goal. We test this in a puck-pushing task, where the goal encoder receives puck coordinates and the actor receives the full robot and puck state. DCP succeeds on 35.9% of matched cases versus 27.9% for CRL, and makes contact from the correct side on 55.1% versus 43.7%.

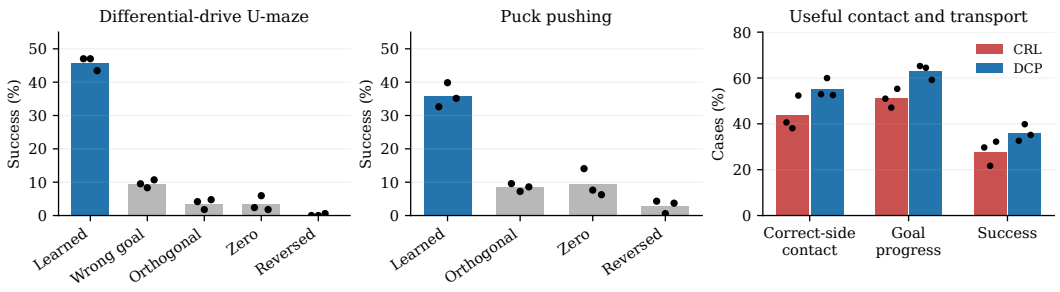


Figure 5: Changing only the direction supplied to a trained actor reduces success in navigation and puck pushing (left, middle). In matched puck-pushing cases, DCP also makes more useful contacts and goal progress than CRL (right). Bars show means and dots show the three training seeds. The puck task uses 128 quartets per seed, varying the robot's side of approach while holding puck and goal fixed within each quartet.

Reversing its direction reduces success to 2.9%, even though contact remains frequent. Direction therefore helps the actor turn contact into goal-directed object motion. The full-state input supplies the robot–puck configuration needed to choose an approach.

4.5 When does a waypoint preserve the final-goal action?

The policy learns from achieved targets during training and receives the final goal at deployment. For this change to work, a useful intermediate target should lead to an action consistent with the final goal. We test this on three trained point-mass DCP policies, evaluating 20,000 start–goal pairs per policy with two alternative targets: an exact midpoint on a graph-shortest path, and an off-path control matched in distance from the source. Holding the actor and physical state fixed, we compare each target's action with the final-goal action. On-path substitution yields mean action cosine 0.789, versus -0.072 for the off-path control; positive alignment with a graph-optimal first move occurs on 95.2% versus 41.9% of cases. The on-path target performs better on these measures across all seeds. Here, the graph provides the midpoint, and substituting it changes both the direction and the distance. We find that the learned policy largely preserves its action when the waypoint lies on the route to the goal. The next question is whether our scorer actually finds such waypoints. Appendix C relates latent triangle slack to direction agreement and includes the radius difference in the actor-output bound.

4.6 How well does the scorer select useful waypoints?

A useful waypoint has to be present among the candidates and then chosen by the scorer. We examine both steps using five trained U-maze encoders and 20,000 matched start–goal pairs and candidate sets per encoder, sampled from the occupancy graph. An exact forward on-path candidate is available in 77.7% of draws. Uniform selection chooses one in 8.3%, while the inner-product scorer selects one in 12.9–18.8%. The scorer also reduces median normalized path slack from 0.761 to 0.197–0.441 and changes average normalized progress from negative to positive. We therefore see a clear improvement over random selection, along with a substantial gap between having a useful candidate and choosing it. Because candidates are sampled from the graph, this experiment tests the scoring rule separately from the states collected during training.

A separate three-seed nested-candidate test increases availability from 77.9% with 32 draws to 94.3% with 128 draws, while useful selection rises by only 2.4–3.9 percentage points. This gap identifies ranking as a limitation even when useful candidates are available. The pool introduces a further constraint. If a fraction p of its states is useful, N independent draws contain at least one useful state with probability $1-(1-p)^N$. When $p=0$, increasing the number of draws cannot help. Alternative ranking rules improve the fixed-encoder test (Appendix C), although their effect on online learning remains to be established.

Table 5: Changing the candidate source in two controlled mazes. Mean success over training, averaged across five seeds. ± denotes sample standard deviation across seeds. Projection maps each proposal to the nearest stored state.

Task	Visited pool	Generator	Generator + projection
Two-route maze	0.457 ± 0.301	0.613 ± 0.202	0.738 ± 0.128
Rare-gateway maze	0.271 ± 0.150	0.520 ± 0.113	0.204 ± 0.291

4.7 Can a learned candidate source improve exploration?

The finite pool limits selection to states the agent has already visited. To address this, we replace pool samples with candidates from an online four-component mixture model trained on trajectory midpoints (Jurgenson et al., 2020). We keep the actor, critic, scorer, candidate count, and 10M-step budget fixed. The generator conditions on the current achieved-goal coordinates and the final goal. We also test whether restricting generated proposals to observed states helps: a third variant projects each proposal to its nearest stored state before scoring. The nine-task comparison uses the default visited-state pool.

The generator increases mean success over training in both tasks (Table 5), exceeding the pool in four of five paired seeds in the two-route maze and all five in the rare-gateway maze. Projection improves the mean in the two-route setting but removes the benefit at the rare gateway, where it can move proposals to the wrong side of the passage. The generator extends candidate coverage beyond the visited-state pool. Appendix D describes its training and the projection step.

4.8 Does seeding the pool with successful states help?

A more direct way to improve candidate availability is to add states from successful trajectories. We use this intervention to test whether supplying useful states is enough to improve control, while keeping replay unchanged. Each task uses three target seeds and matched libraries of successful-path states and states preceding meaningful object motion. On PusherHard, reserving 128 of the 512 slots for successful-path states increases mean success over training from 32.39% with the matched pre-motion library to 38.17%, winning all three paired seeds. However, adding 128 states while preserving all 512 online slots and their expected exposure gives no consistent benefit: successful-path seeding scores 36.26%, compared with 38.09% for the matched control. On AntSoccer, the scorer selects successful-path states on 30.91% of refreshes, versus 4.59% for pre-motion states, and waypoint-to-goal direction agreement improves in every seed. **Ball motion and success do not improve.** Analysis of completed training runs also finds sparse ball motion and little change from the initial representation's distance ordering (Appendix E). In this case, the scorer recognizes the added states, but the policy still fails to turn them into useful ball motion.

5 Conclusion

We asked whether a CRL policy could make better use of the geometry learned by its critic. DCP supplies the actor with the direction and distance to a target in representation space, using visited waypoints during collection and the final goal at deployment. Across nine tasks, DCP improves final mean success and time near goal over CRL on seven tasks each.

Our analyses show that the representation captures route structure and that the actor uses the supplied direction in navigation and manipulation. They also clarify where the difficulty lies. Waypoint selection can slow learning when direct goal conditioning already works, and better candidates are useful only if the learned policy can act on them. Learned generation improves exploration in two controlled mazes; adding successful states to the pool leaves AntSoccer unresolved. Improving the connection between candidate selection and the behavior the policy can execute is therefore an important next step.

Our experiments use simulated state observations. Actor sizes and collected trajectories differ across methods, motivating parameter-matched and shared-data comparisons. Further work could extend DCP to visual observations and combine it with deeper or action-chunked CRL.

REPRODUCIBILITY STATEMENT

The DCP implementation and reproduction scripts are available at https://anonymous.4open.science/r/dcp-supplement-anon-CF41/. Appendix A specifies the objectives, policy inputs, and collection procedure. Appendix B gives task definitions, budgets, and experimental settings. The final task comparisons use five seeds; additional experiments state their seed counts separately. Figure 3 and Appendix G provide learning curves and checkpoint values. Appendix C gives the proof relating latent triangle slack to direction agreement.

ETHICS STATEMENT

This work studies goal-conditioned learning in simulated environments with state observations. No human-subject data are used. Physical deployment requires safety constraints and hardware validation, particularly for contact-rich tasks.

AI USE STATEMENT

Generative AI tools were used to assist with research code, analysis of experimental results, literature retrieval, and drafting and revising the manuscript. They were also used in discussing mathematical claims and proofs. The research ideas originated with the authors. The authors are responsible for the accuracy of the final text, mathematical statements, citations, code, and reported results.

REFERENCES

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://arxiv.org/abs/2408.11052.

Coline Devin, Daniel Geng, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Plan arithmetic: Compositional plan vectors for multi-task control. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1910.14033.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems, 2021. URL https://arxiv.org/abs/1901.10995.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5c48ff18e0a47baaf81d8b8ea51eec92-Paper.pdf.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e7663e974c4ee7a2b475a4775201ce1f-Paper-Conference.pdf.

Benjamin Eysenbach, Vivek Myers, Ruslan Salakhutdinov, and Sergey Levine. Inference via interpolation: Contrastive representations provably enable planning and inference. In *Advances in Neural Information Processing Systems*, volume 37, pp. 58901–58928, 2024. doi: 10.52202/079017-1878. URL https://arxiv.org/abs/2403.04082.

Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax—a differentiable physics engine for large scale rigid body simulation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/d1f491a404d6854880943e5c3cd9ca25-Paper-round1.pdf.

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 35th International Conference on Machine Learning*, volume 80, pp. 1861–1870, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

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.

Tom Jurgenson, Or Avner, Edward Groshev, and Aviv Tamar. Sub-goal trees a framework for goal-based reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5020–5030. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/jurgenson20a.html.

Michal Korniak, Kamil Dybek, Benjamin Eysenbach, Marco Bagatella, and Michał Bortkiewicz. Three steps at a time: Learning representations from action sequences in contrastive RL. *arXiv preprint arXiv:2608.30640*, 2026. URL https://arxiv.org/abs/2608.30640.

Bo Liu, Yihao Feng, Qiang Liu, and Peter Stone. Metric residual networks for sample efficient goal-conditioned reinforcement learning. In *Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, Februray 2023.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c34ddd05eb089991f06f3c5dc36836e0-Paper-Conference.pdf.

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, 2024. URL https://proceedings.mlr.press/v235/myers24a.html.

Vivek Myers, Catherine Ji, and Benjamin Eysenbach. Horizon generalization in reinforcement learning. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2501.02709.

Vivek Myers, {Bill Chunyuan} Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In *39th Conference on Neural Information Processing Systems, NeurIPS 2025*, Advances in Neural Information Processing Systems, pp. 22292–22317. Neural information processing systems foundation, 2025b. Publisher Copyright: © 2025 Neural information processing systems foundation. All rights reserved.; 39th Conference on Neural Information Processing Systems, NeurIPS 2025 ; Conference date: 02-12-2025 Through 07-12-2025.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 3307–3317, Red Hook, NY, USA, 2018. Curran Associates Inc.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BGvt0ghNgA.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34866–34891. Curran Associates, Inc., 2023. doi: 10.52202/075280-1515. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6d7c4a0727e089ed6cdd3151cbe8d8ba-Paper-Conference.pdf.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39737–39761, 2024a. URL https://proceedings.mlr.press/v235/park24g.html.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations*, 2024b. URL https://arxiv.org/abs/2310.08887.

Seohong Park, Deepinder Mann, and Sergey Levine. Dual goal representations. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aMKFTidLSM.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning, 2017. URL https://arxiv.org/abs/1703.01161.

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, pp. 157643–157670, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/e74ee34cc0f2d0780f34ee77d8fba25b-Paper-Conference.pdf.

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, 2023. URL https://proceedings.mlr.press/v202/wang23al.html.

Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive RL: Techniques for robotic goal reaching from offline data. In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023. URL https://openreview.net/forum?id=IAX12qUjYk.

John Luoyu Zhou and Jonathan Kao. Flattening hierarchies with policy bootstrapping. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KaD2Dw8Ahz.

A Implementation and Objectives

A.1 Default algorithm

Algorithm 1 gives the DCP procedure used in the nine-task comparison. We use the visited-state pool and inner-product scorer throughout that comparison; the generator, demonstration-state libraries, and alternative scoring rules are studied in separate experiments. The policy outputs a tanh-squashed Gaussian distribution; evaluation uses the deterministic policy. The actor input concatenates the full state, a 64-dimensional direction, and one radius.

Algorithm 1 Direction-Conditioned Policies

Initialize actor π_η, encoders $(\phi_\theta, \psi_\theta)$, replay, and state pool.
while training budget remains **do**
  Observe full state s and requested final goal g.
  if waypoint refresh is due **then**
    Sample 32 candidate indices with replacement from the 512-state pool.
    Select z^* maximizing $\psi_\theta(q(z))^\top \psi_\theta(g)$.
    Set collection target $y \leftarrow q(z^*)$.
  end if
  Recompute (d, r) from (s, y) using Equation 3.
  Sample and execute $a \sim \pi_\eta(\cdot \mid s, d, r)$; store the transition.
  Refresh the pool from the collected trajectory batch at the collection boundary.
  Sample replay sequences and relabel transitions with future achieved goals.
  Update encoders with backward InfoNCE and the baseline regularizer.
  Update actor and entropy coefficient, holding encoder parameters fixed.
end while
At deployment: use $y = g$ at each step, without the pool or scorer.

Numerical details. The implemented energy is $-\sqrt{\|\phi(s,a) - \psi(g)\|_2^2 + 10^{-6}}$. The goal displacement is divided by $r + 10^{-8}$; a zero displacement therefore produces a zero direction. In addition to Equation 2, the critic uses

$$\mathcal{L}_{\text{reg}} = 0.1 \frac{1}{B} \sum_i \left[\log \sum_j \exp(\ell_{ij} + 10^{-6}) \right]^2 . \tag{6}$$

We differentiate the actor loss through the critic's action input while holding both encoders fixed. The temperature update uses the baseline entropy target.

Future-goal relabeling. Within a sampled sequence, a future index $j > i$ receives weight proportional to γ^{j-i}, restricted to the same trajectory identifier. A small diagonal weight handles positions with no eligible future sample. The selected achieved-goal coordinates provide the critic positive and actor target.

Pool maintenance. The default pool is refreshed from evenly spaced indices of the flattened latest collection batch. Its initial entries are zero-initialized before collected observations replace them. Candidate indices are shared across parallel environments, but candidates are ranked separately for each requested goal. The waypoint is refreshed every 25 steps according to the implementation's environment-step schedule. Encoder evaluations of the current achieved goal and selected target produce an updated direction on every action.

A.2 Training settings

Table 6 lists the shared optimization settings and DCP's waypoint settings. Humanoid U-Maze uses fewer parallel environments and a smaller minibatch; comparisons within that task retain the same settings across methods.

Table 6: Main settings. Humanoid U-Maze uses the stated smaller parallelism and batch size. Replay sizes retain the implementation's buffer configuration units.

Setting	Value
Actor / critic hidden width	256
Hidden layers	2
Representation dimension	64
Actor, critic, and temperature learning rates	3×10^{-4}
Discount	0.99
Minibatch size	256 (Humanoid: 128)
Parallel environments	512 (Humanoid: 128)
Episode length	1,000
Collection unroll length	62
Minimum / maximum replay size	1,000 / 10,000
Contrastive loss / energy	Backward InfoNCE / negative L2 norm
Log-sum-exp penalty coefficient	0.1
Pool size / candidates / refresh steps	512 / 32 / 25

We train DCP, CRL, and SSGC with the same within-task simulation settings and backward InfoNCE configuration. All baseline results come from these matched runs.

A.3 Parameter counts

We count saved actor and critic parameters for all five seeds of each benchmark task. Counts agree across seeds. DCP replaces the two or three raw goal coordinates with 64 direction components and one distance. With first-layer width 256, this adds $256(65 - \dim(g))$ weights: 16,128 for two-dimensional goals and 15,872 for three-dimensional goals. All remaining actor layers and both critic encoders have identical parameter counts. The totals in Table 7 include the actor and both encoders; the learned entropy coefficient adds one scalar to each method.

Table 7: Saved parameter counts for the nine benchmark tasks. Grouped tasks have identical architectures and counts. Percentage increases are relative to CRL.

Tasks	CRL actor	DCP actor	Actor increase	Total increase
AntMaze Big / Hardest / U-Maze	78,096	94,224	20.65%	6.37%
Humanoid U-Maze	144,162	160,034	11.01%	4.15%
AntPush / AntSoccer	78,608	94,736	20.52%	6.35%
PusherEasy / Hard / Hard Far	75,534	91,406	21.01%	6.40%

The controlled point-mass actor has a smaller physical-state input and fewer action outputs, so the added input weights account for a larger relative increase (23.5%). Across the benchmark, that increase is 11–21% for the actor and 4.15–6.40% for the complete actor–critic. The capacity difference is thus confined to the input layer. A parameter-matched raw-goal actor would provide a further control for its contribution to performance.

Computational cost. We measure training throughput over three seeds using the median steps per second in the second half of each run. This reduces the effect of compilation and logging on the comparison. DCP runs at 95.3% of CRL's throughput at the median across tasks, with task ratios ranging from 92.4% to 97.6%. A 512-entry float32 state pool occupies 0.523 MiB at the largest state size (268). The pool and scoring are removed at deployment; goal-encoder evaluation is still required.

B Tasks and Evaluation Protocols

Table 8 specifies each task's goal coordinates, success radius, and training budget. These determine what counts as reaching the goal: AntPush measures the robot's position, whereas AntSoccer measures the ball's position. All three methods receive the same interaction budget for a given task.

Table 8: The nine evaluation tasks. Training steps are matched across methods within each task.

Task	Goal coordinates	Radius	Training steps
AntMaze Big	Ant position	0.5	50M
AntMaze Hardest	Ant position	0.5	50M
Ant U-Maze	Ant position	0.5	50M
Humanoid U-Maze	Humanoid position	0.5	100M
AntPush	Ant position (movable obstacle)	0.5	100M
AntSoccer	Ball XY	0.5	50M
PusherEasy	Cylinder position	0.1	50M
PusherHard	Cylinder position	0.1	100M
PusherHard Far	Cylinder position	0.1	50M

Navigation. AntMaze Big and AntMaze Hardest place an eight-actuator quadruped in different corridor layouts. Ant U-Maze uses the same robot in a U-shaped corridor, where reaching the opposite arm requires moving around the central wall. Humanoid U-Maze uses a 17-actuator humanoid in that layout. Actions control the robot's joints. The actor observes the physical state, including positions and velocities. Ant navigation uses horizontal root position as the achieved goal; Humanoid uses three-dimensional root position, with target height fixed by the environment. Success requires a goal distance below 0.5.

Object interaction. AntPush requires the Ant to reach a target position in a passage obstructed by a movable block. Success is measured from the Ant's position. AntSoccer uses the same type of quadruped to move a ball; the achieved goal is the ball's horizontal position, and success requires ball-to-target distance below 0.5. AntSoccer is called `ant_ball` in the environment implementation; these are two names for the same task.

Robot-arm manipulation. The Pusher tasks use a seven-joint arm to push a cylinder. The achieved goal is the cylinder's three-dimensional position, and success requires distance below 0.1. The variants differ only in their target distribution. Let (u, v) denote the two planar goal-joint offsets, in their stored order. PusherEasy samples (u, v) uniformly from $[-0.55, -0.25] \times [-0.2, 0.2]$; PusherHard uses $[-0.65, 0.35] \times [-0.55, 0.45]$; PusherHard Far uses $[0, 0.35] \times [-0.55, 0.45]$. The first joint translates along world y and the second along world x, relative to the goal body's reference position. All variants initially sample cylinder offsets from $[-0.3, 0] \times [-0.2, 0.2]$, with the same adjustment when the cylinder and goal are too close. Figure 10 provides additional views of target placement.

For an episode of length $T = 1000$, let I_t indicate membership in the goal region. Success is $\mathbf{1}\{\sum_t I_t > 0\}$, averaged over episodes and expressed as a percentage. Time near goal is $\sum_t I_t$, averaged over episodes. Where we report mean success over training in controlled studies, the statistic is the arithmetic mean of the logged evaluation success rates across the fixed evaluation schedule, first computed per seed. Comparisons within each study use the same schedule.

Table 9 distinguishes the main five-seed comparison from the smaller intervention and transfer studies. Each seed corresponds to a separately trained policy or encoder; multiple probes of one policy share that training run. The five-seed studies use seeds 1, 42, 123, 256, and 1234; three-seed interventions use 1, 42, and 123. Uncertainty is computed across independent training seeds. The benchmark and four-way comparison use Student-t intervals with four degrees of freedom: $\bar{x} \pm 2.77645\, s/\sqrt{5}$, where s is sample standard deviation across seeds. Geometry and candidate-source tables instead report sample standard deviations.

C Controlled Geometry and Selection Tests

C.1 Controlled environments

Figure 6 shows the six controlled environments. These supplement the nine benchmark tasks; their results are reported separately. All use 1,000-step episodes. The point-mass maze studies train for

Table 9: Number of seeds and evaluations in each experiment. The geometry experiment and four-way comparison use separately trained policies.

Experiment	Training and evaluation
Nine-task comparison	Five seeds per method and task; 50M or 100M steps.
Four-way collection/input comparison	Five seeds, 10M steps, 20 evaluations, separately in each layout.
Representation geometry	Five encoders per method and layout, trained for 10M steps.
Candidate-source comparison	Five seeds per candidate source and maze, 10M steps.
Waypoint substitution	Three policies; 20,000 paired probes per policy.
Differential-drive intervention	Three policies; 42 directed pairs and four headings per policy.
Puck-pushing intervention	Three policies; 128 matched quartets per policy.
Demonstration-state seeding	Three target seeds; successful states added to the candidate pool.
Frozen-CRL transfer	One behavioral seed; 50M source pretraining plus 25M adaptation.

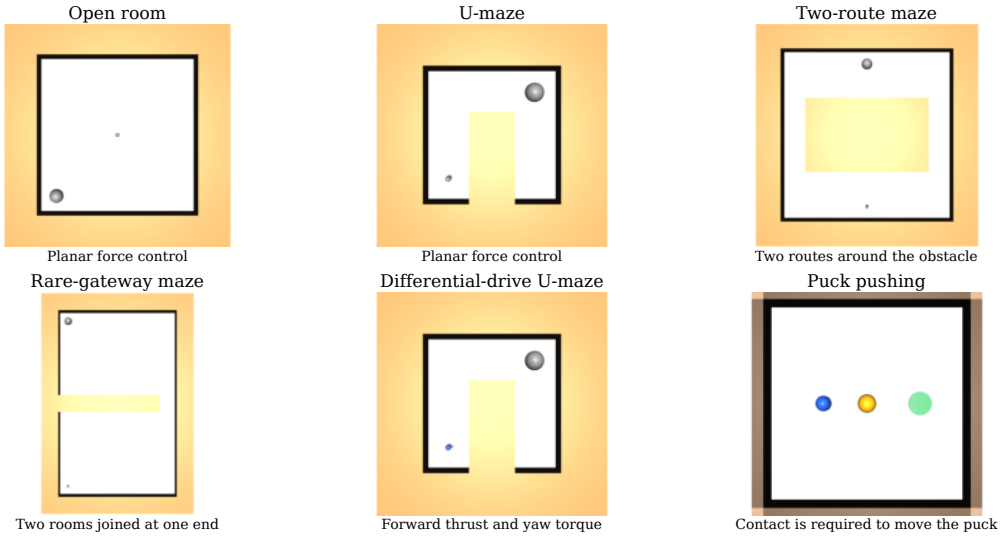


Figure 6: The controlled environments, rendered from their simulation models with floor textures hidden. The four point-mass environments differ in wall layout and start/goal distributions. The differential-drive robot adds heading-dependent motion. In puck pushing, the blue agent must contact the yellow puck and move it to the green target. Images show illustrative task configurations.

10M interactions and evaluate at 20 checkpoints. We use the same dynamics and training settings across methods within each layout.

Point-mass mazes. The state is (x, y, v_x, v_y), the action is a two-dimensional force, and the goal is a position. Forces change velocity, which persists between steps. Wall cells have side length 4, and success requires position error below 0.5. The open room has boundary walls only and starts near its center. The U-maze starts in one arm, with goals sampled from allowed free cells. The two-route maze places start and goal on opposite sides of a central block, allowing a route around either end. The rare-gateway maze has two rooms separated by a wall, with a single passage near one end; the start and goal lie across the divider, away from that passage. The generator experiments use the latter two layouts.

Differential-drive U-maze. The physical state is $(x, y, \theta, v_x, v_y, \dot{\theta})$; the two actions apply forward thrust along the current heading and yaw torque. The robot cannot directly command sideways force, and velocity persists. The goal is horizontal position, with success radius 0.5. For the direction test,

each of three learned policies is evaluated on 42 directed start–goal pairs and four initial headings per pair, holding these cases fixed across interventions.

Puck pushing. A planar force-controlled robot pushes a separate, unactuated puck. The actor observes both positions and velocities, an eight-dimensional physical state; the goal encoder receives only puck position. Success requires puck-to-target distance below 0.55. Contact transfers the robot's motion to the puck. The matched evaluation uses 128 groups of four cases per seed. Within a group, the puck and target are fixed while the robot approaches from four different sides. Groups vary target direction and distance, absolute position, and initial robot–puck separation. This tests whether the actor combines its direction input with the physical arrangement needed for useful contact.

C.2 Graph reference and interpretation

The point-mass state is (x, y, v_x, v_y) and the action is a two-dimensional force. The U-maze is represented by a weighted occupancy graph with cardinal and collision-free diagonal connections. For the global-distance comparison, the graph supplies 100,128 directed pairs of distinct positions. Graph distances measure spatial route lengths. Local-direction probes compare the negative spatial gradient of $\|\psi(q(s)) - \psi(g)\|$ with graph-optimal outgoing moves. The actor input is defined in Equation 3.

In the open room, graph distance and Euclidean distance largely agree. The layouts have different initial-state and goal distributions; comparisons of scoring rules and actor inputs are performed within each layout.

C.3 Comparing waypoint selection rules

We want to know whether the scoring rule chooses useful candidates when they are available. We therefore sample start, goal, and candidate positions from the occupancy graph, hold the encoders fixed, and give each rule the same 20,000 draws of 32 candidates.

With graph distance D, a candidate is counted as forward on-path when

$$D(s, z) + D(z, g) - D(s, g) = 0, \qquad D(z, g) < D(s, g), \tag{7}$$

up to the numerical tolerance. Relative slack divides the left-hand slack by $D(s, g)$; relative progress is $[D(s, g) - D(z, g)]/D(s, g)$. Both require distinct source and goal. The default graph uses eight subdivisions and 448 nodes. Additional resolutions test sensitivity to discretization.

We compare two alternative scoring rules on the same candidate sets without retraining the encoders. One maximizes the cosine between $\psi(z) - \psi(s)$ and $\psi(g) - \psi(s)$. The other minimizes $\|\psi(z) - \psi(s)\| + \|\psi(g) - \psi(z)\|$. Across five trained encoders, the default inner-product scorer selects a forward on-path candidate in 12.9–18.8% of draws; cosine ranking achieves 24.7–27.8%, and minimizing the sum of learned distances achieves 28.6–33.0%. Uniform selection achieves 8.3%. Both alternatives improve waypoint selection in this test. Whether they improve online learning remains untested: changing the selection rule would also change the experience collected and the representation learned.

C.4 A geometric statement about the direction input

When a waypoint and goal lie in similar directions from the current embedding, they provide similar direction inputs to the actor. We can quantify this relation using the slack in the triangle formed by their embeddings. We use exact normalization and nonzero displacements for the statement below. Let $x = \psi(q(s))$, $w = \psi(q(z))$, and $v = \psi(g)$. Define

$$a = \|w - x\|, \quad b = \|v - w\|, \quad c = \|v - x\|, \quad h = a + b - c, \quad u_z = \frac{w - x}{a}, \quad u_g = \frac{v - x}{c}. \tag{8}$$

Proposition 1 (Latent slack and direction agreement)**.** *For $a, c > 0$, the triangle slack $h \geq 0$ satisfies*

$$\|u_z - u_g\|^2 = \frac{h(b + c - a)}{ac} \leq \frac{2h}{a}. \tag{9}$$

Proof. The cosine law gives $b^2 = a^2 + c^2 - 2ac\langle u_z, u_g\rangle$. Consequently,

$$ac\,\|u_z - u_g\|^2 = b^2 - (a-c)^2 \tag{10}$$
$$= (a+b-c)(b+c-a) = h(b+c-a). \tag{11}$$

The triangle inequality $b \le a + c$ implies $b + c - a \le 2c$, proving the bound. □

This identity describes the geometry of the Euclidean embedding. If a deterministic actor is separately Lipschitz in direction and radius, with constants L_d, L_r at a fixed state, then

$$\|\pi(s, u_z, a) - \pi(s, u_g, c)\| \le L_d\sqrt{2h/a} + L_r|a - c|. \tag{12}$$

The distance input matters here. Even when $h = 0$, the waypoint and goal can be at different distances from the current embedding. For example, $x = 0$, $w = 1$, and $v = 2$ give identical directions and different radii. Equation 12 accounts for both changes.

The distance $\|\psi(q(s)) - \psi(g)\|$ is symmetric in its two goal-coordinate arguments and may vanish between distinct coordinates. It is therefore a pseudometric on those coordinates. Relating it to optimal-control cost requires additional assumptions, as in metric constructions for contrastive successor features and quasimetric value learning (Myers et al., 2024; Wang et al., 2023).

D Candidate Generation and Demonstration Seeding

D.1 Online candidate generator

The generator is a conditional mixture-density network with four diagonal-Gaussian components and two hidden layers of width 128. Its inputs are the current achieved-goal coordinates and a desired endpoint. Training samples an ordered triple of indices $i < k < j$ from a replay sequence, with $k = \lfloor (i + j)/2 \rfloor$, and fits the midpoint coordinates by negative log likelihood. Triples crossing trajectory identifiers are masked out. The learning rate is 3×10^{-4} and the generator warm-up is 1M interaction steps. Component scales are constrained between 0.05 and 2, with initial scale and offset scale 0.5. Candidate count stays at 32.

The unprojected variant uses sampled goal-coordinate proposals as the waypoint targets. The projected variant maps proposals to the nearest pool state's goal coordinates, then applies the same scorer. Projection returns a stored target but can move a proposal across an obstacle. All three variants retain the same deployment policy input. The default DCP configuration uses the visited-state pool.

D.2 Demonstration-state libraries

We construct each task's libraries from nine successful DCP episodes, three per source seed. The successful-path library contains 128 states chosen around object displacement. For the control, we take the same number of states from those episodes before meaningful object motion. This lets us test the value of the states supplied to the pool while keeping replay and contrastive training unchanged.

In the reserved-slot comparison, each library occupies 128 of the 512 slots, leaving 384 online slots. The augmented comparison instead preserves all 512 online slots and adds 128 library slots. It draws 40 candidates so that the expected number of online draws remains 32, approximately preserving each online state's exposure. The reserved-slot experiment tests the value of library content under constrained capacity. The augmented experiment tests the benefit of adding those states to the full online pool.

Table 10: PusherHard demonstration-state experiments, three target seeds. Entries are mean evaluation success over training (%). Paired wins compare successful-path against pre-motion seeding. DCP without seeding scores 35.76% in these experiments.

Pool intervention	Pre-motion	Successful path	Paired wins
Reserve 128 of 512 slots	32.39	38.17	3/3
Add 128 to 512 online slots	38.09	36.26	0/3

Table 10 shows that useful demonstration states help when they replace part of a capacity-limited pool: successful-path seeding exceeds the matched pre-motion control in all three seeds. Adding these states to the complete online pool gives no corresponding benefit. The comparison therefore depends on both the content of the added states and which online candidates they replace.

For AntSoccer, successful-path seeding improves mean waypoint/final-goal direction cosine from 0.596 to 0.655 and latent slack from 1.096 to 1.000, without improving ball motion or success. Improving candidate availability and selection is insufficient to resolve this task.

E Additional Analyses

E.1 Learning early in training and maintaining success

The direction input is learned online, so we also examine whether an initially inaccurate representation delays learning and whether success is maintained once it begins. In a three-seed analysis of the nine tasks, DCP crosses 2% success earlier than CRL on seven tasks and ties on two. At 5% and 10%, it is earlier on six, tied on one, and later on two. Its success over the first 25M steps is higher on seven tasks, tied on AntPush, and lower on AntSoccer. These comparisons concern early learning, separately from the final five-seed results.

After first crossing each threshold, we measure the fraction of the next ten evaluations that falls below it. The median fraction at 5% and 10% is 0% for DCP, compared with 10% and 20% for CRL. This measures persistence after initial success.

On AntPush, mean DCP success decreases from 44.8% at 75M steps to 42.7% at 100M steps in the five-seed results. The learning curves show the full course of training, including this late decline.

E.2 AntSoccer representation and activity

To understand what happens on AntSoccer, we examine how much the representation changes and how often the agent moves the ball. Across five seeds, we compare distance ordering on fixed coordinate pairs with its ordering at initialization. On AntSoccer, the mean Spearman-correlation change is only -0.050 at 25M and -0.021 at the final checkpoint; the final changes are larger in magnitude on AntPush and the Pusher tasks. Only 3.8% of retained consecutive AntSoccer transitions move the ball by more than 10^{-3}. AntSoccer combines sparse object movement with little change in distance ordering.

We measured these signals in completed runs. They identify AntSoccer as unusual among the evaluated tasks. Tracking fixed coordinate probes and replay activity would also make them available during training; predicting individual-run outcomes requires further testing.

E.3 Frozen CRL representations

If direction conditioning can use geometry learned by CRL, it should also be possible to train a new actor on a frozen CRL representation. We test this directly. A single-seed AntMaze Big experiment starts from a CRL checkpoint trained for 50M steps and trains a new DCP actor for 25M additional interactions. One variant freezes only the goal encoder and adapts the state–action encoder; another freezes both encoders. A fresh joint DCP receives 25M steps without pretraining.

Table 11: Frozen-representation transfer on AntMaze Big, seed 42. The transferred variants use 50M CRL pretraining steps and 25M actor-training steps; fresh DCP uses 25M steps in total.

Variant	Pretraining	Final success (%)	Mean training success (%)
Fresh joint DCP	0	25.4	16.72
Frozen CRL goal encoder	50M	22.3	11.60
Both CRL encoders frozen	50M	23.0	12.46

Table 11 shows that a new direction-conditioned actor can use a CRL representation even when both encoders are frozen, reaching 23.0% final success. Fresh DCP reaches 25.4% with a smaller total

interaction budget. This establishes reuse in the tested run, while the unequal pretraining budgets preclude a sample-efficiency comparison.

The frozen encoder parameters remain unchanged at all 26 checkpoints examined during adaptation. Separate implementation checks confirm that 15 CRL checkpoints across AntMaze Big, PusherHard, and AntSoccer can be loaded and kept frozen while the actor is updated. The AntMaze Big experiment demonstrates behavioral transfer for one seed. Its generality across seeds and tasks remains to be tested.

F Additional Visualizations

Variation across training seeds. Figure 7 expands Table 4 into individual-seed comparisons. Each line connects encoders trained with the same seed. The first two panels measure how well learned distances order graph shortest-path distances, over all state pairs and over the longest-distance third. DCP improves both measures in every seed. The third panel measures local descent: whether decreasing learned distance initially points toward a graph-optimal move. Its larger variation shows that improved global distance ordering can coexist with local directional errors.

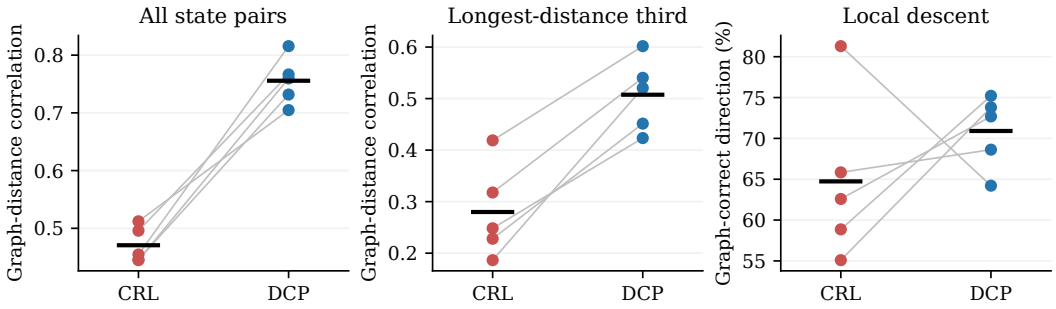


Figure 7: U-maze representation measurements for all five seeds after 10M steps. Connected dots compare CRL and DCP with the same seed; black horizontal marks show means. DCP has higher graph-distance correlation in every seed, both over all pairs and over the longest-distance third. The local-descent comparison shows more variation across seeds.

Distance fields in AntMaze Big. Figures 8 and 9 place learned distances alongside executed trajectories. For a fixed goal, we evaluate the goal encoder over spatial coordinates and plot the resulting distance to the goal embedding. The first example shows successful trajectories for both methods; the second shows DCP reaching the goal while CRL stalls. Comparing the contours with the maze walls helps reveal whether nearby physical positions are separated by learned distance when a wall forces a detour. Colors are scaled within each panel, so the comparison concerns spatial structure. These examples illustrate behaviors that contribute to the aggregate results; the five-seed learning curves report their broader variability.

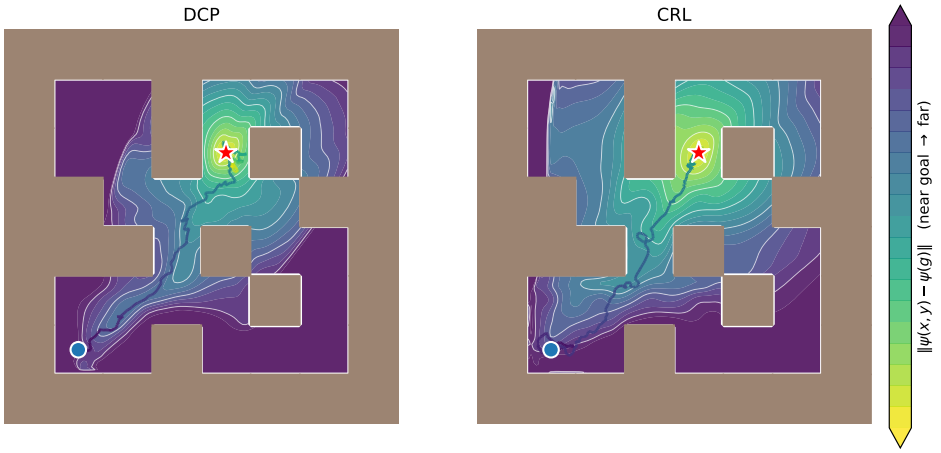


Figure 8: AntMaze Big distance fields and example trajectories for a seed in which both DCP and CRL reach the goal. Contours show $\|\psi(x, y) - \psi(g)\|$; the blue circle and red star mark start and goal. Both fields reflect the corridor layout. Colors are scaled separately for each method.

Pusher target placement. Figure 10 shows how target placement changes the required cylinder motion. The arm must first approach the cylinder from a suitable side and then maintain contact while

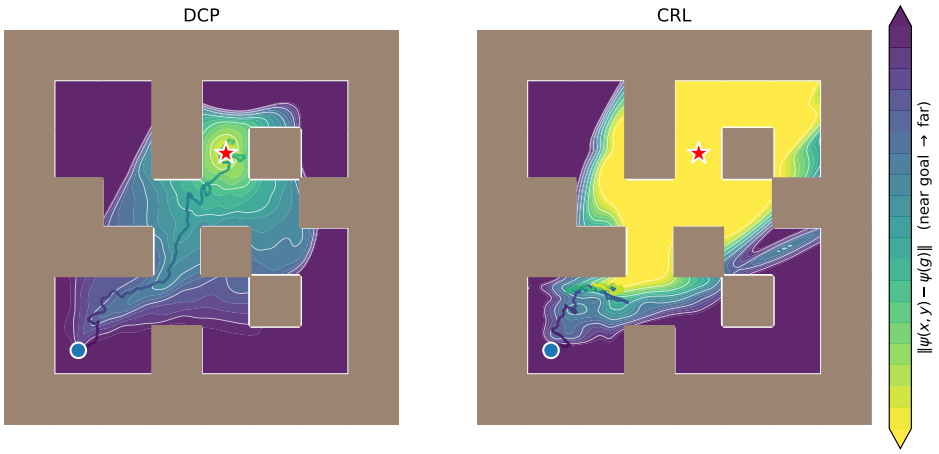
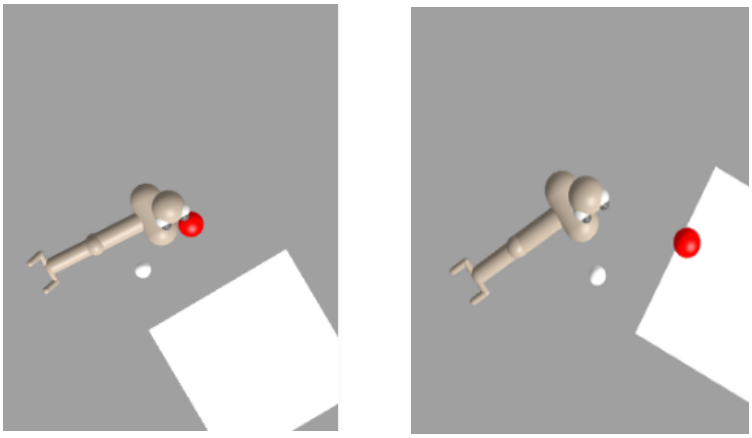


Figure 9: A second AntMaze Big example (seed 1), in which DCP reaches the goal and CRL stalls. The color range in each panel is clipped to the 1st–99th percentiles of distances along that method's trajectory. The field and trajectory illustrate one outcome; the main comparison averages all five seeds.

pushing toward the target. The near-target image is drawn from PusherHard's target distribution; PusherHard Far restricts the target distribution as specified in Appendix B. These views help interpret why reaching the cylinder alone is insufficient for success.



PusherHard: near target PusherHard Far

Figure 10: Additional Pusher views. The white cylinder must be moved to the red target. Target placement changes the required approach and pushing direction. The near-target view is an example from PusherHard.

Comparing waypoint scoring rules. Figure 11 holds the encoder and sampled candidate sets fixed while changing the selection rule. An exact forward on-path waypoint is available in 77.7% of draws. The gap between availability and selection therefore measures missed opportunities within the sampled set. Direction-agreement and latent-slack scoring select on-path candidates more frequently than the default inner product in this test. Evaluating these rules during online training would additionally measure how their choices change the subsequently collected experience.

Generating candidates beyond the visited-state pool. Figure 12 expands Table 5 into paired seed results. Each point averages success across the 20 evaluation checkpoints of a complete training run. The generator improves over the pool in four of five seeds in the two-route maze and all five in the rare-gateway maze. Projecting generated candidates onto visited states improves the mean in the

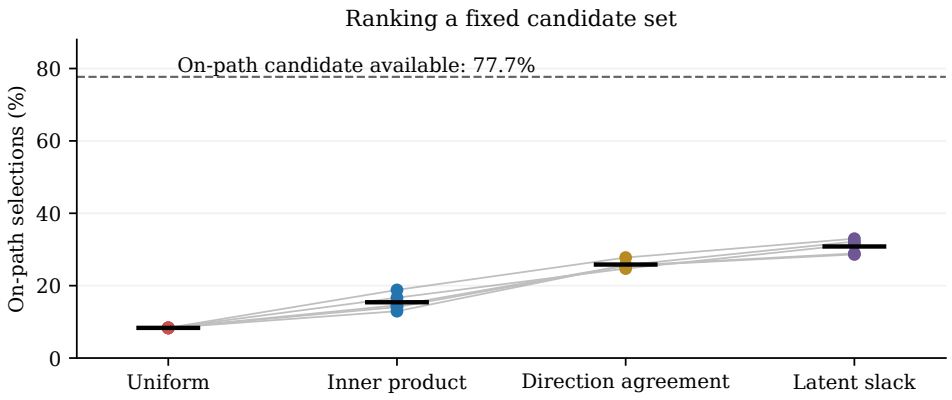


Figure 11: Waypoint selection with five fixed U-maze encoders. Connected points use the same encoder and candidate draws; black marks show means. A forward on-path candidate is available in 77.7% of draws, but the default inner-product rule selects one much less often. Ranking by direction agreement or latent slack improves this fixed-encoder test.

two-route maze but reduces it in the rare-gateway maze. This difference is consistent with the layouts: projection can preserve useful route choices when visited states cover both routes, but can remove proposals beyond a poorly explored gateway.

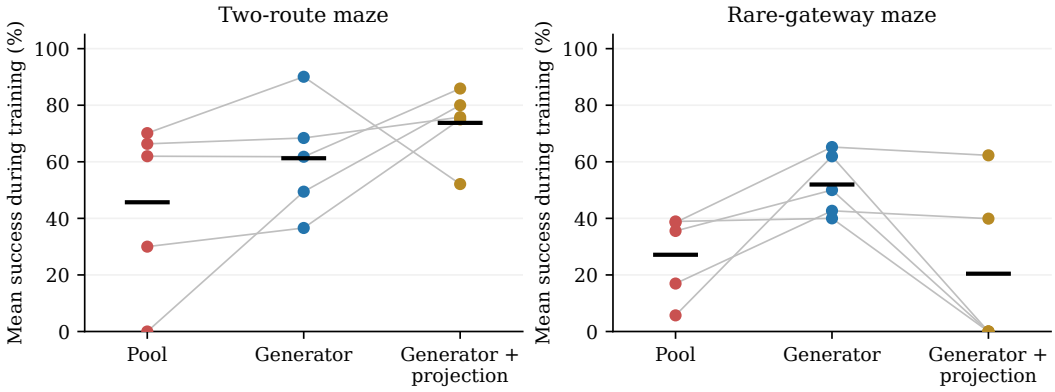


Figure 12: Candidate-source comparison with all five seeds shown. Lines connect runs with the same seed; black marks show means. The generator improves mean success during training in both layouts. Nearest-state projection has different effects in the two mazes and removes the mean benefit in the rare-gateway maze. Each value averages the 20 evaluation checkpoints.

G Complete Benchmark Checkpoints

Figure 3 gives the complete nine-task learning curves. The following tables give the five-seed checkpoint evaluation. Nominal checkpoints are matched to the nearest available logged evaluation. Dashes indicate that a checkpoint lies beyond the task's training budget. Student-t intervals are reported without clipping to the metric's bounds.

Table 12 reports whether the goal region is reached at least once per episode. Reading across a task shows when performance improves and whether it persists. On PusherHard, DCP reaches 33.7% at 25M steps, compared with 0.9% for CRL; CRL later improves to 28.8% at 100M, while DCP reaches 47.7%. AntPush shows a different pattern: DCP's mean falls from 44.8% at 75M to 42.7% at 100M. CRL retains the higher mean at every reported AntSoccer checkpoint. Bold entries identify the largest mean, without implying statistical significance.

Table 13 reports how long the agent stays near the goal. Comparing it with Table 12 separates occasional arrival from sustained proximity. At the final checkpoint, DCP's PusherEasy success of 55.4% accompanies 297.8 steps near the goal, compared with CRL's 4.4% and 26.2 steps. On Humanoid U-Maze, DCP reaches the goal in 8.4% of episodes but averages only 3.6 steps near it, indicating brief arrivals. The intervals describe variation across the five training seeds; the full curves in Figure 3 provide the evaluations between these selected checkpoints.

Table 12: Success at least once (%), mean ± reported 95% confidence-interval half-width over five seeds. Bold marks the largest mean at each task/checkpoint.

Environment	Method	10M	25M	50M	75M	100M
AntMaze Big	DCP	**18.4 ± 5.4**	**23.8 ± 3.3**	**29.9 ± 2.9**	–	–
	CRL	15.1 ± 3.7	17.7 ± 5.3	18.3 ± 6.5	–	–
	SSGC	18.3 ± 1.2	22.0 ± 2.9	23.3 ± 3.0	–	–
AntMaze Hardest	DCP	**9.5 ± 2.5**	**14.5 ± 2.1**	**16.1 ± 2.3**	–	–
	CRL	5.8 ± 2.5	10.5 ± 3.8	12.2 ± 5.6	–	–
	SSGC	7.7 ± 1.6	13.2 ± 4.8	14.2 ± 8.0	–	–
Ant U-Maze	DCP	**40.4 ± 9.5**	**52.9 ± 6.8**	**55.0 ± 13.3**	–	–
	CRL	26.8 ± 4.5	40.7 ± 8.5	40.8 ± 10.2	–	–
	SSGC	28.4 ± 9.6	43.1 ± 8.1	49.8 ± 4.2	–	–
Humanoid U-Maze	DCP	4.6 ± 8.4	6.7 ± 11.4	**13.4 ± 9.5**	**7.7 ± 9.2**	**8.4 ± 9.6**
	CRL	**9.8 ± 11.5**	**7.0 ± 12.0**	3.0 ± 8.5	0.0 ± 0.0	0.0 ± 0.0
	SSGC	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	2.5 ± 6.9	0.0 ± 0.0
AntPush	DCP	0.0 ± 0.0	**1.1 ± 2.8**	**29.1 ± 19.3**	**44.8 ± 8.0**	42.7 ± 12.6
	CRL	0.0 ± 0.0	0.0 ± 0.0	28.9 ± 6.3	38.2 ± 8.8	45.6 ± 24.1
	SSGC	0.0 ± 0.0	0.0 ± 0.0	28.0 ± 32.1	40.8 ± 21.1	**51.4 ± 15.5**
PusherEasy	DCP	**2.8 ± 2.2**	**30.9 ± 35.0**	**55.4 ± 9.6**	–	–
	CRL	1.9 ± 0.9	2.2 ± 1.8	4.4 ± 4.1	–	–
	SSGC	1.9 ± 0.9	7.2 ± 10.8	34.1 ± 38.5	–	–
PusherHard	DCP	**2.0 ± 1.9**	**33.7 ± 14.2**	**37.2 ± 11.7**	**45.6 ± 10.8**	**47.7 ± 5.8**
	CRL	0.3 ± 0.2	0.9 ± 0.6	2.0 ± 2.5	7.3 ± 12.8	28.8 ± 32.4
	SSGC	0.2 ± 0.3	1.2 ± 1.4	8.4 ± 14.8	21.6 ± 26.4	27.8 ± 30.7
PusherHard Far	DCP	**1.1 ± 2.8**	**25.8 ± 17.3**	**47.4 ± 5.7**	–	–
	CRL	0.0 ± 0.0	1.6 ± 1.9	12.5 ± 24.0	–	–
	SSGC	0.0 ± 0.0	0.2 ± 0.3	9.0 ± 15.1	–	–
AntSoccer	DCP	2.6 ± 0.9	8.8 ± 2.9	12.9 ± 4.3	–	–
	CRL	**5.6 ± 3.6**	**14.9 ± 4.8**	**22.6 ± 3.4**	–	–
	SSGC	2.5 ± 1.3	7.0 ± 2.5	14.1 ± 3.6	–	–

Table 13: Time near goal (steps per 1,000-step episode), mean ± reported 95% confidence-interval half-width over five seeds.

Environment	Method	10M	25M	50M	75M	100M
AntMaze Big	DCP	78.7 ± 37.1	**94.6 ± 26.2**	**115.3 ± 14.6**	–	–
	CRL	75.3 ± 12.7	83.2 ± 29.4	89.2 ± 28.0	–	–
	SSGC	**91.9 ± 19.1**	94.6 ± 20.5	99.8 ± 9.0	–	–
AntMaze Hardest	DCP	30.3 ± 12.6	**65.1 ± 12.9**	63.2 ± 13.3	–	–
	CRL	28.4 ± 15.9	39.9 ± 24.0	48.1 ± 23.7	–	–
	SSGC	**32.6 ± 8.1**	59.0 ± 21.0	**66.0 ± 41.4**	–	–
Ant U-Maze	DCP	**195.3 ± 58.0**	223.5 ± 27.9	238.1 ± 76.2	–	–
	CRL	161.9 ± 33.1	**233.2 ± 62.8**	**261.3 ± 77.2**	–	–
	SSGC	157.7 ± 65.3	230.4 ± 38.6	252.7 ± 52.1	–	–
Humanoid U-Maze	DCP	**1.3 ± 2.5**	**4.9 ± 9.5**	**3.4 ± 2.5**	**1.4 ± 1.9**	**3.6 ± 4.5**
	CRL	1.0 ± 1.7	1.4 ± 3.0	0.9 ± 2.5	0.0 ± 0.0	0.0 ± 0.0
	SSGC	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.4 ± 1.1	0.0 ± 0.0
AntPush	DCP	0.0 ± 0.0	**3.8 ± 10.4**	**131.8 ± 113.5**	**190.9 ± 71.6**	181.7 ± 95.6
	CRL	0.0 ± 0.0	0.0 ± 0.0	95.9 ± 42.6	122.7 ± 30.1	143.4 ± 105.6
	SSGC	0.0 ± 0.0	0.0 ± 0.0	103.2 ± 114.9	182.9 ± 83.9	**199.1 ± 93.3**
PusherEasy	DCP	18.2 ± 9.0	**148.2 ± 179.5**	**297.8 ± 90.9**	–	–
	CRL	**18.8 ± 9.3**	20.3 ± 15.1	26.2 ± 8.4	–	–
	SSGC	**18.8 ± 9.3**	40.0 ± 49.3	180.7 ± 183.4	–	–
PusherHard	DCP	**9.3 ± 4.2**	**161.5 ± 84.4**	**185.7 ± 45.4**	**285.3 ± 103.5**	**280.9 ± 28.5**
	CRL	3.1 ± 2.2	9.4 ± 5.5	13.1 ± 9.9	36.2 ± 66.8	125.7 ± 136.7
	SSGC	2.3 ± 2.7	9.4 ± 8.1	54.0 ± 99.4	133.1 ± 161.3	170.5 ± 186.0
PusherHard Far	DCP	**2.5 ± 4.8**	**124.1 ± 88.4**	**260.7 ± 66.8**	–	–
	CRL	0.0 ± 0.0	10.7 ± 15.9	48.4 ± 80.4	–	–
	SSGC	0.0 ± 0.0	1.6 ± 2.7	49.0 ± 82.3	–	–
AntSoccer	DCP	11.2 ± 8.8	33.6 ± 18.7	42.4 ± 22.1	–	–
	CRL	**20.8 ± 10.5**	**46.3 ± 13.8**	**76.4 ± 7.7**	–	–
	SSGC	9.4 ± 6.9	29.8 ± 12.6	48.2 ± 31.9	–	–