EXPLOITING LOCAL DYNAMICS REGULARITY FOR REUSABLE SKILLS IN OFFLINE HIERARCHICAL RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical Reinforcement Learning (HRL) promises to solve long-horizon Reinforcement Learning (RL) tasks more efficiently than non-hierarchical counterparts by discovering and reusing temporally-extended skills. However, obtaining skills that are actually reusable remains an open challenge. Towards this end, we focus on abstractions that exploit the intuition of local dynamics: local transitions in different global contexts require similar kinds of action sequences. By aligning these contexts with the action sequences they require, we are able to learn which skills to reuse and where to reuse them. In principle, this information should benefit many HRL algorithms, where high-level policies have to reason about the low-level skills they use. The resulting algorithm CARL (**C**ontrastive **A**ction-based **R**epresentations for Reusable **L**ocal Control) shows both qualitative clustering of meaningful skills in complex humanoid environments and improved downstream performance on the OGBench benchmark when integrated with HIQL. We visualize additional results and video rollouts on our accompanying website.[1]

1 INTRODUCTION

Hierarchical Reinforcement Learning (HRL) methods offer a framework (Sutton et al., 1999) for abstracting low-level policies or action sequences—commonly referred to as *skills*—so that they can be reused to accomplish complex tasks. However, this promise of reusable skills has often been outweighed by the limitations of existing approaches (Zhang et al., 2020), which include training instability when co-learning skills with the policies that use those skills (Levy et al., 2019; Wang et al., 2025), and failure to leverage reusability. One promising direction involves representing low-level skills as goal-conditioned policies (Park et al., 2023; Hafner et al., 2022; Eysenbach et al., 2019), which decouples low- and high-level policy training. However, representing skills via a goal-conditioned low-level policy loses many of the desirable properties of skills, such as temporal abstraction and consistent reusability. In this work, we formalize the idea of local dynamics—the idea that many local regions of the state space share similar dynamics—to enable reuse of low-level goal-conditioned skills across the state space.

To formalize the idea of local dynamics we define local, finite-horizon MDPs for each state to capture its surrounding states and their dynamics. We then draw on bisimulation as a principled way to define state equivalences based on these MDPs. To derive a practical algorithm from this principle, we leverage the idea of *behavioral similarity*: two state-goal pairs (s_1, g_1) and (s_2, g_2) are considered similar when a policy can use the same skill $(a_1, a_2, a_3, \ldots, a_k)$ to achieve the transition from s_1 to g_1 and from s_2 to g_2. We hypothesize that behavioral similarity will naturally emerge when many states share local dynamics structure, making it a useful heuristic for understanding when skills can be reused. Our method is built upon these insights and leverages a contrastive learning objective to align state-goal pairs with the action sequences that achieve them.

We introduce CARL (**C**ontrastive **A**ction-based **R**epresentations for Reusable **L**ocal Control) to learn representations of local dynamics from offline datasets at a fixed skill horizon. By recognizing the underlying local dynamics structure, CARL identifies where skills can be reused instead of forcing HRL methods to relearn low-level policies from scratch. We demonstrate this by integrating these abstractions into existing HRL algorithms, HIQL and HGCBC (Park et al., 2023) which yields clear performance benefits on the OGBench benchmark.

[1] https://sites.google.com/view/behavior-rep/home

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.

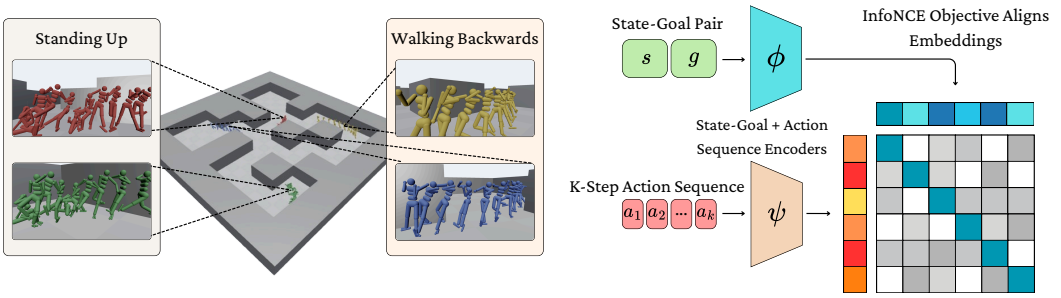


Figure 1: Learning state–goal representations for skill reuse. Humanoid poses left-to-right in each pane reflect progress in time. CARL learns a representation $\phi(s, g)$ that clusters state–goal pairs if they admit the same kinds of k-step action sequences. CARL clusters "walking backward" and "standing up" behaviors for humanoids situated in globally different parts of the maze.

Furthermore, our objective shapes the latent geometry so that state–goal pairs requiring similar local skills are embedded nearby, even when they occur in different regions of the environment. Figure 1 illustrates this in a humanoid maze environment, where CARL groups skills together regardless of where they occur in the maze.

As noted, CARL relies on an offline dataset and fixed horizon to capture local dynamics, which may constrain expressible skills when data coverage is poor or the horizon fails to capture useful structure. We include ablations analyzing the effects of coverage, imbalance, and horizon length. Overall, our work shows that local dynamics structure provides a principled basis for skill extraction and reuse, helping HRL reconnect with the temporal abstractions central to long-horizon decision-making.

2 Related Work

Hierarchical Reinforcement Learning Hierarchical Reinforcement Learning studies how to solve long-horizon Reinforcement Learning (RL) tasks by introducing *temporal abstraction*, which refers to high-level decisions that invoke temporally extended behaviors. One of the first formulations of this idea is the options framework, which models a skill as a policy with an initiation set and termination condition, enabling reuse of temporally extended actions within an SMDP view of control (Sutton et al., 1999). Early HRL work explored a variety of hierarchical decompositions and skill abstractions, including feudal-style manager–worker architectures (Dayan & Hinton, 1992), value-function decomposition methods such as MAXQ (Dietterich, 2000), and policy-constraining formalisms such as hierarchies of abstract machines (HAMs) (Parr & Russell, 1997).

Ideally, we would like to discover these temporal abstractions end-to-end. In practice, however, jointly learning a hierarchy suffers from non-stationarity: as the low-level policy changes, the effective dynamics faced by the high-level policy shift, which often destabilizes joint optimization. Many modern approaches therefore decouple the training of the hierarchy, using subgoals (goals in state space) to train a local goal-conditioned policy while a high-level policy learns to propose useful targets (Nachum et al., 2018; Levy et al., 2019).

More recent methods go further and learn compact representations of subgoals rather than working with raw states. Director (Hafner et al., 2022) compresses latent states of a learned world model into discrete goal codes, and HIQL (Park et al., 2023) learns a representation of state-goal pairs through its value function, which serves as both the high-level action space and the input to the low-level policy. Although these representations have made HRL methods more stable and effective, we believe much of the potential of hierarchy remains untapped. In this work, we explore subgoal representations centered on the reuse of low level behaviors as a path toward realizing it.

Abstractions via Behavioral Equivalences and Invariances In this section, we discuss methods that seek to generalize by learning behavior-preserving abstractions, where distinct states (or state-goal contexts) are mapped to similar representations when they are interchangeable (Agarwal et al., 2021; Hansen-Estruch et al., 2022; Islam et al., 2023; Ajay et al., 2021; Park et al., 2025b). A common theme is to define an equivalence relation that preserves global decision-relevant structure (e.g., via

value-, policy-, or model-based notions of similarity) and then learn representations that collapse equivalent situations while discarding irrelevant variation. This perspective is closely related to the notion of bisimulation, which characterizes when two states can be treated as equivalent without materially changing long-horizon outcomes (Li et al., 2006; Ferns & Precup, 2014; Rudolph et al., 2024; Hansen-Estruch et al., 2022; Castro, 2020).

Our work fits into this line of research, but fundamentally targets a complementary notion of equivalence grounded in *local reuse*. Rather than requiring states to match in long-horizon value or reward structure, we ask when different state-goal pairs admit similar local goal-reaching behaviors, which allows us to cluster exactly the state-goal pairs that are the same from the perspective of short-horizon control, regardless of the long-horizon effects.

Contrastive Representations for Control Contrastive learning extracts representations by aligning positive pairs and separating negative pairs, typically using objectives such as InfoNCE (van den Oord et al., 2018; Radford et al., 2021). In RL, contrastive methods have been applied to train task-relevant structure and enable downstream generalization (Eysenbach et al., 2022; Agarwal et al., 2021; Laskin et al., 2020). In particular, Contrastive RL (Eysenbach et al., 2022) connects contrastive objectives to goal-conditioned RL by training representations in which reaching a goal corresponds to matching future states under a learned similarity metric. Our method leverages contrastive learning, but uses it to induce a different latent structure.

3 Preliminaries

3.1 Offline Goal-Conditioned RL

We study temporal abstraction in HRL in the offline goal-conditioned RL (OGCRL) setting. We model the environment as a Markov decision process $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$ is a reward function, and $\gamma \in (0, 1)$ is a discount factor. We consider the case where $\mathcal{G} \subseteq \mathcal{S}$ and when our transition dynamics are deterministic[2].The objective is to learn a goal-conditioned policy $\pi : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$ that maximizes the expected return over a time horizon H: $\max_\pi J(\pi) = \max_\pi \mathbb{E}_{g\sim\rho,\, \zeta\sim\rho^\pi}\big[\sum_{t=0}^{H} \gamma^t r(s_t, g)\big]$. We define a trajectory $\zeta = (s_0, a_0, s_1, a_1, \ldots, s_H)$ to be a sequence of states and actions with a fixed horizon H. Trajectories are sampled according to $\rho^\pi(\zeta) = \mu(s_0) \prod_{t=0}^{H-1} \pi(a_t \mid s_t, g)\, p(s_{t+1} \mid s_t, a_t)$, where μ is the initial state distribution and ρ is the goal distribution. We write $p(s' \mid s, \mathbf{a}_k) = \sum_{s_1,\ldots,s_{k-1}\in\mathcal{S}} \prod_{i=0}^{k-1} p(s_{i+1} \mid s_i, a_i)$, with $s_0 = s$ and $s_k = s'$, to denote the k-step transition function, where $\mathbf{a}_k = (a_0, \ldots, a_{k-1})$ is a k-step action sequence starting at state s and a_i is its i-th action. Throughout, we use calligraphic letters for spaces and lowercase letters for values and functions.

3.2 Local Dynamics

We now formalize the local dynamics structure of a state in terms of its transition dynamics over a fixed horizon k. This formalization lets us define an equivalence relation on states that captures when short-horizon behaviors can be reused across them.

Definition 3.1 (k-ball)**.** For any $k \in \mathbb{N}$ and $s \in \mathcal{S}$, the *k-ball* of s, denoted $\mathcal{B}_k(s)$, is the set of states reachable from s within k steps. Formally, $s' \in \mathcal{B}_k(s)$ if there exist $n \in \{0, 1, \ldots, k\}$ and an action sequence $\mathbf{a}_n$ such that $p(s' \mid s, \mathbf{a}_n) > 0$.

Definition 3.2 (k-ball MDP)**.** The *k-ball MDP* rooted at s is the finite-horizon MDP

$$\mathcal{M}_s^{(k)} = \big(\mathcal{B}_k(s),\ \mathcal{A},\ p,\ r,\ k\big),$$

whose action space $\mathcal{A}$ is shared with $\mathcal{M}$, and whose transition function p and reward r are the restrictions of those of $\mathcal{M}$ to $\mathcal{B}_k(s)$.

We now use the k-ball MDP to formalize a notion of state similarity grounded in local dynamics. Prior work adapting bisimulation to RL (Ferns et al., 2004; Zhang et al., 2021; Rudolph et al., 2024)

[2]Our analysis is limited to the deterministic dynamics case. We leave it to future work to generalize this to the stochastic dynamics setting

measures similarity between MDPs through sequences of rewards or single-step controllability metrics under identical action sequences. Our formalism instead considers dynamics directly, capturing whether two MDPs are interchangeable for goal-reaching behavior.

Definition 3.3 (Dynamics Bisimilarity)**.** Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MDPs with state spaces $\mathcal{S}_1$ and $\mathcal{S}_2$, a shared action space $\mathcal{A}$, and transition functions p_1 and p_2. We say that $\mathcal{M}_1$ and $\mathcal{M}_2$ are *dynamics-bisimilar* if there exists a total relation $B \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ such that for every $(x, x') \in B$ and all $a \in \mathcal{A}$,

$$\forall\, y \in \mathcal{S}_1,\ \exists\, y' \in \mathcal{S}_2 \text{ (and vice versa)}: \quad p_1(y \mid x, a) = p_2(y' \mid x', a),$$
$$\forall\, C \in \mathcal{S}_1/B: \quad \bar{p}_1(C \mid x, a) = \bar{p}_2(B^{-1}(C) \mid x', a),$$

where $\mathcal{S}_1/B$ is the partition of $\mathcal{S}_1$ induced by B (the set of equivalence classes C of B-related states in $\mathcal{S}_1$), $B^{-1}(C) = \{y' \in \mathcal{S}_2 : \exists\, y \in C,\ (y, y') \in B\}$ is the corresponding subset of $\mathcal{S}_2$, and $\bar{p}(C \mid s, a) = \sum_{s' \in C} p(s' \mid s, a)$ extends p to subsets.

We say that two states s and s' share local dynamics if their corresponding k-ball MDPs, $\mathcal{M}_s^{(k)}$ and $\mathcal{M}_{s'}^{(k)}$, are *dynamics-bisimilar*.

4 Learning Representations for Skill Reuse

Although our state equivalence relates local dynamics, directly estimating it for skill reuse is challenging: it requires constructing the k-ball MDP around each state and comparing transition structure under all relevant actions, which is especially hard in continuous, high-dimensional domains. We therefore seek an approximation that preserves the key ideas of our formalism while remaining simple to implement.

Our core insight is that offline datasets with near-expert behavior can reveal local dynamics structure without directly modeling dynamics-bisimulation relations (Definition 3.3). Intuitively, when two states have dynamics-bisimilar k-ball MDPs, the same k-step action sequence accomplishes equivalent transitions from both. A near-expert data-collecting policy[3]. will exploit this, leaving repeated action-sequence structure as a footprint in the dataset. We call this footprint *behavioral similarity*:

Definition 4.1 (Behavioral similarity)**.** Let $p_{\mathcal{D}}(\mathbf{a} \mid s, g)$ be the k-step inverse dynamics of the dataset $\mathcal{D}$, i.e., the distribution over action sequences $\mathbf{a} \in \mathcal{A}^k$ given start state s and goal g. State-goal pairs (s_1, g_1) and (s_2, g_2) are *behaviorally similar*, written $(s_1, g_1) \approx (s_2, g_2)$, if and only if $p_{\mathcal{D}}(\cdot \mid s_1, g_1) = p_{\mathcal{D}}(\cdot \mid s_2, g_2)$.

Let (s, s') be a pair of states. Let $\mathcal{D}$ be a dataset that covers the k-balls of s and s' following an expert policy. If we have an optimal behavioral similarity encoder $\phi : \mathcal{S} \times \mathcal{G} \to \mathbb{R}^n$ over this dataset it will encode $\phi(s, g) = \phi(s', g')$ if they share the same optimal action sequences. From

Algorithm 1: CARL

Input: Offline dataset $\mathcal{D}$, horizon k, temperature τ;
Initialize: Encoders $\phi(s, g)$ and $\psi(\mathbf{a})$;
Initialize: Goal-conditioned learner $\pi(\cdot \mid \phi(s, g))$;
while *not converged* **do**
  Sample batch $b \sim \mathcal{D}$ of k-step segments $(s_t, \mathbf{a}_k, s_{t+k})$;
  Train encoders: $\mathcal{L}_{\text{InfoNCE}}(b; \phi, \psi)$; `// Eq. (1)`
  Train policy: $\pi(\cdot \mid \phi(s, g))$;

Algorithm 2: HIQL+CARL (changes in blue)

Input: dataset $\mathcal{D}$, horizon k, temperature τ;
Initialize: value $V_{\theta_V}(s, \phi(s, g))$, policies $\pi^h_{\theta_h}(\cdot \mid s, g)$, $\pi^\ell_{\theta_\ell}(\cdot \mid s, \phi(s, s'))$;
Initialize: encoders $\phi_{\theta_\phi}(s, s')$, $\psi_{\theta_\psi}(\mathbf{a})$;
while *not converged* **do**
  Sample minibatch $(s_t, a_{t:t+k-1}, s_{t+k}, g) \sim \mathcal{D}$, horizons $m \sim \text{Unif}\{1, \dots, k\}$;
  Train encoders:
    $\mathcal{L}_{\text{InfoNCE}}(\{s^i_t, s^i_{t+m}, a^i_{t:t+k-1}\}_i; \phi_{\theta_\phi}, \psi_{\theta_\psi})$;
  Update value: $V_{\theta_V}(s_t, \phi_{\theta_\phi}(s_t, g))$;
  $z^\star_t \leftarrow \phi_{\theta_\phi}(s_t, s_{t+k})$; `// embed subgoal`
  Train high-level: fit $\pi^h_{\theta_h}(z^\star_t \mid s_t, g)$ (AWR);
  Train low-level: fit $\pi^\ell_{\theta_\ell}(a_t \mid s_t, z^\star_t)$ (AWR);

[3]We assume that the expert policy follows all optimal strategies between a state and goal with equal likelihood. Otherwise it might prefer different paths between different state-goal pairs that are still dynamically similar in the intuitive sense.

this, we can recover a bisimulation metric by sampling reachable goals and checking whether $\phi(s, g) = \phi(s', g)$. Behavioral similarity is therefore a relaxation of the stricter bisimulation, but one that still captures the essential properties that we require for skill reuse.

This motivates our method CARL (**C**ontrastive **A**ction-based **R**epresentations for Reusable **L**ocal Control), which learns to relate skills with the global contexts in which they are used. Using batches of size N from a diverse offline dataset, we learn representations $\phi(s, g)$ of state-goal pairs and $\psi(\mathbf{a})$ of action sequences by minimizing

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\big(\langle \phi(s^i, g^i), \psi(\mathbf{a}^i)\rangle/\tau\big)}{\sum_{j=1}^{N} \exp(\langle \phi(s^i, g^i), \psi(\mathbf{a}^j)\rangle/\tau)}, \tag{1}$$

where (s^i, g^i) is the i-th state-goal pair in the batch and $\mathbf{a}^i$ is the action sequence that reaches g^i from s^i within k steps. Each state-goal pair and its action sequence form a positive pair, while the other action sequences in the batch, $\mathbf{a}^j$ for $j \neq i$, serve as in-batch negatives. Minimizing the loss pulls each state-goal pair toward shared action sqeuences. Algorithm 1 summarizes the training procedure.

4.1 InfoNCE Recovers Behavioral Similarity

At the optimum of the symmetric InfoNCE objective, the score function $f(s, g, \mathbf{a}) = \phi(s, g)^\top \psi(\mathbf{a})$ recovers the pointwise mutual information between the state-goal pair (s, g) and the action $\mathbf{a}$, up to a global additive constant (Uesaka et al., 2025)

$$\phi^\star(s, g)^\top \psi^\star(\mathbf{a}) = \log \frac{p_\mathcal{D}(\mathbf{a} \mid s, g)}{p_\mathcal{D}(\mathbf{a})} + c,$$

where c does not depend on s, g, or $\mathbf{a}$. On the right-hand side, (s, g) enters only through the conditional $p_\mathcal{D}(\mathbf{a} \mid s, g)$. The marginal $p_\mathcal{D}(\mathbf{a})$ and the constant c are the same for every state-goal pair. So two pairs receive identical scores for every action exactly if they exhibit behavioral similarity. This gives us

$$\phi^\star(s, g) = \phi^\star(s', g') \iff p_\mathcal{D}(\cdot \mid s, g) = p_\mathcal{D}(\cdot \mid s', g').$$

As a result, ϕ is therefore a sufficient statistic for behavioral similarity.

4.2 Integration with Hierarchical Offline RL

Because CARL embeds state-goal pairs solved by similar action sequences nearby, regardless of where they occur in the state space, its representation is well suited as the input to the low-level policy in subgoal-based HRL, where a policy can then reuse the same skill across contexts rather than relearning it in each region.

Following this intuition, we integrate CARL into HIQL (Park et al., 2023), a recent and competitive hierarchical offline RL algorithm, by co-training its subgoal representation with CARL's objective. The representation then forms the inputs of the low-level policy and value function and the output space of the high-level policy, as highlighted in blue in Algorithm 2. Appendices B and C describe our hyperparameter changes and experiments analyzing these design choices.

We apply the same changes to **H**ierarchical **G**oal-**C**onditioned **B**ehavior **C**loning (HGCBC), the baseline used by Park et al. (2023), to assess the generality of CARL's benefits. Section 5 shows that HIQL+CARL achieves win rates of 20/22 on state-based and 13/14 on image-based tasks against HIQL, and HGCBC+CARL achieves 21/21 on state-based tasks against HGCBC.

5 Experiments

Our experiments analyze both the embedding structure produced by CARL and its effect on downstream performance when integrated into an existing HRL method. The following sections seek to address the questions below.

- Q1. Does CARL's representation structure enable skills to be reused?
- Q2. How does CARL benefit HRL methods in downstream tasks?
- Q3. What are the important components for capturing behavioral similarity?
- Q4. How does CARL organize different skills in its representation structure?

5.1 Skill Reuse in Toy Environments

(Q1) Skill Reuse. To test how CARL enables skill reuse, we design a toy environment consisting of five identical grid-world rooms (Figure 2). These rooms differ only in their global (x, y) coordinates, which form the proprioceptive state for the agent.

We test zero-shot generalization in two settings. First, a goal-conditioned policy trained in a single room must solve the same task in held-out rooms. We compare HIQL to HIQL augmented with a CARL encoder trained on all rooms. Since CARL aliases state-goal pairs reachable by the same action sequences, the policy should transfer through the shared representation. Our experiments confirm this: HIQL+CARL solves all held-out rooms, while HIQL fails to generalize. Second, we scale to 20 identical grid-world rooms, training on 4 and testing on the remaining 16.

HIQL+CARL generalizes to 12 unseen rooms versus 8 for HIQL alone, further demonstrating that explicitly modeling skill reusability improves transfer.

5.2 Evaluating Performance Benefits for HRL

(Q2) HRL Integration. Here, we examine how CARL benefits HRL by benchmarking on OGBench tasks. We analyze results for HIQL+CARL and HGCBC+CARL, as described in Section 4.2.

Experimental setup We evaluate on two task categories from OGBench. The first is the locomotion suite, where various embodiments (point-mass, ant, humanoid) have to navigate through a maze. These tasks test learning high-level planning and low-level locomotion skills from offline data.

We use the navigate datasets for these tasks, collected with a noisy expert policy that wanders the maze by achieving randomly sampled goals. This strategy yields broad coverage with repeating low-level structure, providing opportunities for skill reuse and leveraging CARL's core capability.

The second category is the manipulation suite, where a 6-DoF UR5e arm performs tasks like stacking cubes, arranging tabletop scenes, or solving puzzles, testing object manipulation and combinatorial generalization. We test on the play datasets, collected by open-loop planners with temporally correlated noise. These planners rely on repeated behavior primitives, which we expect CARL to take advantage of.

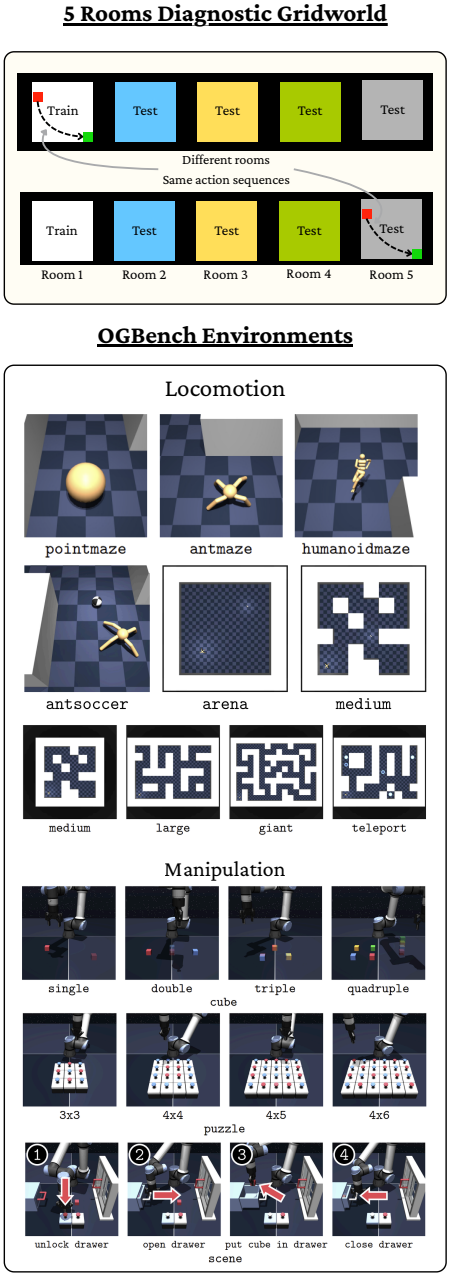


Figure 2: Examples of environments used to benchmark CARL, including a diagnostic grid world as well as environments from the OGBench suite.

These environments feature continuous control and span a range of complexity, from the low-dimensional point maze to the high-dimensional humanoid, allowing us to assess how CARL scales with state and action space complexity. The robotics domains introduce additional challenges through multi-entity interactions. Together, these environments test CARL across a diverse range of control difficulties. We defer to the OGBench paper for additional environment and dataset specifications (Park et al., 2025a).

Table 2: Success rates (%) on state-based OGBench tasks, reported as mean ± 95% CI over 8 seeds. **(a) HRL Comparisons** contrasts HIQL and HGCBC with their CARL-augmented variants, where we **bold** values within 95% of the better method in each pair. **(b) CARL Ablations** compares variants of CARL combined with HIQL, where we **bold** values within 95% of the best ablation. The final row reports per-algorithm win rates excluding ties, with the highest **bolded**. HIQL results are taken from Park et al. (2025a), with their reported standard deviations converted to 95% confidence intervals.

	(a) HRL Comparisons				(b) CARL Ablations			
Task	HIQL+CARL	HIQL	HGCBC+CARL	HGCBC	CARL	Single-Action CARL	Multi-Action Prediction	Single-Action Prediction
pointmaze-medium	**84.4 ± 3.1**	79 ± 4.3	**13.6 ± 1.8**	0.0 ± 0.0	**84.4 ± 3.1**	**86.6 ± 3.8**	67.6 ± 4.0	74.1 ± 5.4
pointmaze-large	**75.4 ± 3.1**	58 ± 4.3	**19.5 ± 3.8**	0.4 ± 0.3	**75.4 ± 3.1**	**74.7 ± 8.0**	58.1 ± 9.5	62.6 ± 7.1
pointmaze-giant	**64.5 ± 15.8**	46 ± 7.6	**0.0 ± 0.0**	**0.0 ± 0.0**	**64.5 ± 15.8**	53.6 ± 13.2	22.7 ± 7.3	18.2 ± 9.2
pointmaze-teleport	**31.4 ± 14.0**	18 ± 3.3	**29.7 ± 1.8**	16.7 ± 1.2	**31.4 ± 14.0**	25.0 ± 8.0	18.0 ± 4.7	19.4 ± 6.4
antmaze-medium	**97.9 ± 0.9**	**96 ± 0.9**	**73.1 ± 1.5**	59.8 ± 1.6	**97.9 ± 0.9**	**95.5 ± 2.4**	**96.9 ± 1.7**	**96.1 ± 1.2**
antmaze-large	**91.9 ± 2.4**	**91 ± 1.7**	**64.9 ± 0.9**	59.3 ± 1.9	**91.9 ± 2.4**	86.4 ± 6.1	**91.5 ± 3.5**	**91.5 ± 1.2**
antmaze-giant	**75.2 ± 3.8**	65 ± 4.3	**18.4 ± 1.3**	8.5 ± 0.8	**75.2 ± 3.8**	47.1 ± 7.3	70.0 ± 4.3	**72.3 ± 3.5**
antmaze-teleport	**41.4 ± 2.1**	**42 ± 2.6**	**38.5 ± 1.3**	**36.9 ± 1.9**	41.4 ± 2.1	42.5 ± 4.5	**46.6 ± 2.4**	**47.0 ± 4.5**
humanoidmaze-medium	**90.5 ± 2.4**	**89 ± 1.7**	**44.4 ± 1.3**	32.4 ± 0.9	**90.5 ± 2.4**	**88.9 ± 4.5**	**86.5 ± 3.1**	**87.1 ± 2.8**
humanoidmaze-large	**58.3 ± 3.8**	49 ± 3.3	**31.0 ± 0.8**	22.6 ± 0.8	**58.3 ± 3.8**	51.1 ± 3.3	45.9 ± 3.3	50.0 ± 4.0
humanoidmaze-giant	**27.2 ± 3.1**	12 ± 3.3	**20.0 ± 1.1**	9.5 ± 1.3	**27.2 ± 3.1**	18.1 ± 3.8	17.0 ± 5.2	16.3 ± 3.1
antsoccer-medium	**12.8 ± 2.4**	**13 ± 1.7**	**7.0 ± 0.4**	5.9 ± 0.7	**12.8 ± 2.4**	5.6 ± 1.7	10.9 ± 2.8	12.1 ± 2.6
antsoccer-arena	**63.7 ± 3.1**	58 ± 1.7	**24.9 ± 1.6**	12.5 ± 0.8	**63.7 ± 3.1**	56.0 ± 7.3	56.9 ± 4.0	**63.0 ± 5.2**
cube-single	**32.8 ± 3.1**	15 ± 2.6	**6.9 ± 0.7**	4.4 ± 0.8	**32.8 ± 3.1**	30.5 ± 6.9	25.9 ± 4.3	22.4 ± 3.5
cube-double	**23.4 ± 4.5**	6 ± 1.7	**1.3 ± 0.3**	1.1 ± 0.3	**23.4 ± 4.5**	18.9 ± 4.0	5.4 ± 2.6	5.4 ± 1.2
cube-triple	**15.2 ± 3.5**	3 ± 0.9	**2.1 ± 0.6**	0.4 ± 0.3	15.2 ± 3.5	**24.4 ± 4.0**	6.4 ± 2.1	9.0 ± 4.0
cube-quadruple	**0.1 ± 0.2**	0 ± 0.0	**0.1 ± 0.1**	0.0 ± 0.0	0.1 ± 0.2	**1.8 ± 1.7**	0.4 ± 0.7	0.3 ± 0.5
puzzle-3x3	**45.5 ± 7.3**	12 ± 1.7	**6.0 ± 0.9**	4.1 ± 0.3	**45.5 ± 7.3**	15.0 ± 4.3	27.1 ± 5.2	28.1 ± 3.8
puzzle-4x4	**35.2 ± 7.1**	7 ± 1.7	**5.9 ± 0.9**	0.3 ± 0.2	**35.2 ± 7.1**	22.5 ± 4.0	14.2 ± 3.1	13.7 ± 2.4
puzzle-4x5	**4.2 ± 2.4**	**4 ± 0.9**	**1.6 ± 0.6**	0.4 ± 0.2	4.2 ± 2.4	**5.0 ± 1.4**	3.0 ± 1.9	3.1 ± 0.9
puzzle-4x6	**3.6 ± 2.1**	3 ± 0.9	**0.6 ± 0.3**	0.3 ± 0.2	3.6 ± 2.1	**3.9 ± 1.7**	2.4 ± 1.4	2.3 ± 1.7
scene	**70.5 ± 5.2**	38 ± 2.6	**15.8 ± 1.0**	8.5 ± 1.1	**70.5 ± 5.2**	**69.8 ± 6.4**	55.7 ± 5.7	64.3 ± 3.1
win-rate	**20/22**	2/22	**21/21**	0/21	**16/22**	5/22	0/22	1/22

OGBench State-Based Performance We show the effect of CARL on HRL algorithms in Table 2a by comparing HIQL and HGCBC with their augmented variants on OGBench. Notably, HIQL+CARL improves over HIQL by at least 10 percentage points on giant maze variants, 17 on smaller cube environments, 28 on smaller puzzle environments, and an impressive 30 on the scene task. HGCBC+CARL tends to outperform HGCBC by around 10 percentage points on most navigation environments. This provides strong evidence that CARL's representation benefits HRL in downstream tasks.

CARL's gains are more limited on antmaze-teleport, antsoccer-medium, and the harder cube and puzzle tasks. The underlying causes remain unclear, but we hypothesize that these environments pose distinct challenges: stochastic dynamics (portals in antmaze-teleport), long horizons with entity-centric generalization (antsoccer-medium, cube), and combinatorial generalization over large configuration spaces (puzzle). Addressing these will likely require advances in the representation structure, such as representations that are invariant to entity permutations or that explicitly support compositional reasoning at the level of the high-level policy.

In robotics domains, CARL improves HIQL substantially more than it improves HGCBC. We attribute this to the difference in how each algorithm extracts its low-level policy: HIQL uses AWR, which upweights high-advantage actions, while HGCBC relies on unweighted behavioral cloning.

Table 1: HIQL+CARL vs. HIQL on visual environments. Mean success rates (%) ± 95% CI over 4 seeds. Values within 95% of the best per row are **bolded**; for win rates, ties are excluded and the highest is **bolded**. HIQL results are taken from Park et al. (2025a), with their reported standard deviations converted to 95% confidence intervals.

Task	HIQL+CARL	HIQL
visual-antmaze-medium	**97.3 ± 2.7**	**93 ± 6.4**
visual-antmaze-large	**85.5 ± 5.6**	53 ± 14.3
visual-antmaze-giant	**43.0 ± 6.2**	6 ± 6.4
visual-antmaze-teleport	**45.3 ± 2.7**	37 ± 3.2
visual-humanoidmaze-medium	**2.3 ± 2.4**	0 ± 0.0
visual-humanoidmaze-large	**0.0 ± 0.0**	**0 ± 0.0**
visual-humanoidmaze-giant	**0.0 ± 0.0**	**0 ± 0.0**
visual-cube-single	87.8 ± 8.1	**89 ± 0.0**
visual-cube-double	**41.8 ± 9.4**	39 ± 3.2
visual-cube-triple	**24.8 ± 2.7**	21 ± 0.0
visual-cube-quadruple	**14.8 ± 4.5**	14 ± 1.6
visual-puzzle-3x3	**75.5 ± 4.6**	73 ± 12.7
visual-puzzle-4x4	**83.5 ± 7.0**	60 ± 65.2
visual-puzzle-4x5	**18.0 ± 3.7**	13 ± 14.3
visual-puzzle-4x6	**15.8 ± 8.0**	9 ± 9.5
visual-scene	**54.3 ± 2.1**	49 ± 6.4
win-rate	**13/14**	1/14

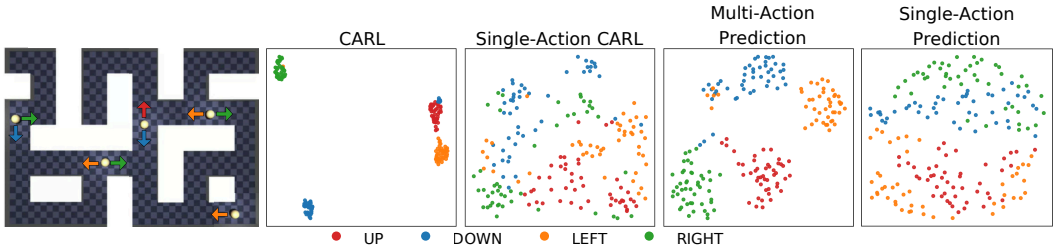


Figure 3: UMAP visualization of learned embeddings colored by behavioral mode. When embeddings are trained using CARL, state-goal pairs that require similar action sequences cluster tightly, even when the raw states differ substantially. Other ablations show more overlap and looser clusters.

Combining our method (to discover where a skill can be reused) with AWR (to discover which skills are optimal) allows us to reuse skills that are more likely to achieve the global task, and thus results in better task performance. While it remains unclear why robotics tasks in particular benefit disproportionately, the pattern suggests a complementary relationship between extracting high-quality skills and knowing where to use them.

OGBench Pixel-Based Performance Visual observations represent a higher-dimensional and more realistic setting for real-world control, making them an important test of CARL's applicability beyond compact state representations. Thus, we benchmark our HIQL+CARL variant on OGBench's visual suite without further hyperparameter tuning, and present results in Table 1. We find significant gains on longer-horizon antmaze tasks (over 30 percentage points on large and giant). Robotics domains also see improvements of at least 5 percentage points on scene and the larger puzzle tasks, with gains of over 20 percentage points on puzzle-4x4. However, CARL does not always improve performance. Visual humanoid control is still too difficult even with CARL's representation, and cube tasks also benefit less than other robotics domains.

5.3 Ablating CARL's Design for Capturing Behavioral Similarity

(Q3) Component Ablation. In this section, we analyze how different parts of our representation learning approach affect its ability to capture behavioral similarity. Our full method optimizes $\mathcal{L}_{\text{InfoNCE}}$ (Eq. (1)) with the action sequence $\mathbf{a}$ as the contrastive target. The three ablations modify this along one or both of two axes. *Single-Action CARL* replaces $\mathbf{a}$ with its first action a_0, keeping the contrastive objective. *Multi-Action Prediction* keeps $\mathbf{a}$ but replaces the contrastive loss with direct regression, $\|\xi(\phi(s,g)) - \mathbf{a}\|_2^2$, where ξ is a network head that takes in representations of state-goal pairs produced by ϕ. *Single-Action Prediction* applies both changes.

To compare these ablations, we use UMAP (McInnes et al., 2018) to visualize how well each method organizes its latent structure by behavioral similarity. More concretely, we construct state-goal pairs corresponding to motion in cardinal directions for the pointmaze environment from OGBench. We use pointmaze because its simple dynamics make it easy to reason about the ideal clustering structure: pairs should cluster by the direction of motion they induce.

Figure 3 shows that only CARL's combination of action-sequence modeling and a contrastive objective produces tight, disjoint clusters suitable for downstream skill reuse. The ablations yield scattered, overlapping clusters, with the exception of Multi-Action Prediction, whose clusters are looser but still disjoint enough to capture skill reuse meaningfully. This suggests that modeling action sequences is important for capturing behavioral similarity and supporting skill reuse.

We also evaluate downstream performance on state-based OGBench environments in Table 2b to contextualize these differences in representation geometry. CARL generally outperforms the other variants, emphasizing the importance of both action-sequence modeling and a contrastive objective. Single-Action CARL comes closest in performance, while the prediction methods perform comparably to each other. This suggests that the contrastive objective is an effective way to translate behavioral similarity into useful downstream representations.

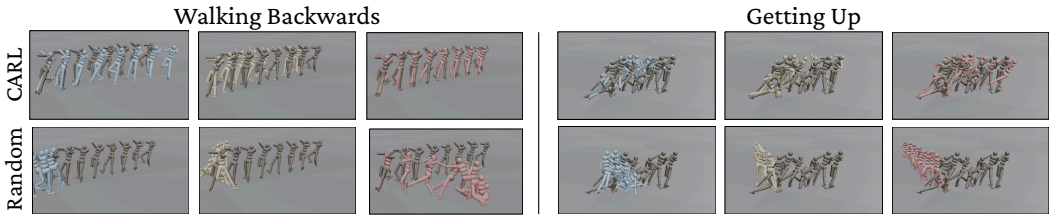


Figure 4: Visualizations of nearest-neighbor trajectories under CARL and a random encoder for two reference skills (walking backwards and standing up); although these neighbors originate from across the maze, we shift them to share the reference's starting position to highlight their behavioral similarity. The gray trajectory is the reference; blue, yellow, and red are its nearest neighbors.

5.4 Visualizing CARL's Representation Structure

(Q4) Skill Structure. We have observed how CARL's geometry tightly clusters state-goal pairs by behavioral similarity. In this section, we examine whether this result extends to higher-dimensional and less structured settings by studying our representations in the humanoidmaze environment.

To probe CARL's clustering structure, we select nearest neighbors of two reference state-goal pairs, one corresponding to walking backwards and one to standing up, and visualize their rollouts. As shown in Figure 4, the nearest-neighbor trajectories (blue, yellow, and red) are behaviorally coherent with the gray reference trajectory, whereas the random encoder's neighbors are not. Together with Figure 3, this demonstrates that CARL's representation organizes less structured, complex skills by their behavioral similarity, even in scaled-up environments.

6 Limitations

Skill extraction horizon. In this work, the notion of a skill depends on the choice of extraction horizon k. We analyze CARL's sensitivity to k in Appendix C.4 and find that the optimal value is environment-dependent. A more principled approach would remove this dependence, discovering relevant local dynamics structure at various horizons to solve a given task.

Dataset coverage. Our representation depends on the diversity, coverage, and quality of the offline dataset. Data from a random policy lacks the behaviors needed to learn useful representations, and sparse coverage can cause the collapse of distinct state-goal pairs. Appendix C.5 characterizes this sensitivity and shows that CARL does not amplify it relative to HIQL. Addressing these limitations in an online setting will require intelligent data collection or a more robust objective.

7 Closing Remarks

We introduced CARL, a representation learning approach for goal-conditioned offline RL that explicitly captures reusable short-horizon behaviors. Our central claim is that many RL problems contain *local dynamical structure*, allowing local and global information to be separated. By learning embeddings that cluster state-goal pairs according to the short-horizon behaviors they admit, CARL builds local reusability into the low-level policy of a hierarchy, providing a concrete mechanism for skill reuse that improves existing hierarchical RL frameworks.

Based on CARL, there are several exciting directions for future work. The behavior-centric view suggests a connection between representation learning and skill discovery. Future work could explicitly extract, label, or compose the clusters in CARL's representation structure, extending prior work (Lin et al., 2022) to yield a unified framework for offline skill discovery and hierarchical control.

Finally, CARL currently operates offline, but the behavioral similarity objective could be extended to the online setting. More broadly, addressing the dataset coverage limitations discussed earlier could yield sample-efficiency gains and widen the applicability of hierarchical RL.

AI USE STATEMENT

We used generative AI tools to polish the writing and notation, format our LaTeX source, and search for related work and citations. During our research, we consulted them for coding and for refining ideas. All research ideas, methods, experiments, and conclusions are our own, and we reviewed all AI-assisted code and text. We take full responsibility for the content of this paper.

REFERENCES

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qda7-sVg84.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10069–10076, 2020.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In Stephen J. Hanson, Jack D. Cowan, and C. Lee Giles (eds.), *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann, 1992. URL https://proceedings.neurips.cc/paper_files/paper/1992/file/d14220ee66aeec73c49038385428ec4c-Paper.pdf.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Ben Eysenbach, Russ R. Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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.

Norm Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 210–219. AUAI Press, 2014. URL https://www.auai.org/uai2014/proceedings/individuals/67.pdf.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pp. 162–169. AUAI Press, 2004.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8407–8426. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen-estruch22a.html.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs), 2016. URL https://arxiv.org/abs/1606.08415.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm van Seijen, Remi Tachet des Combes, and John Langford. Agent-controller representations: Principled offline rl with rich exogenous information, 2023. URL https://arxiv.org/abs/2211.00164.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, pp. 531–539, 2006. URL http://rbr.cs.umass.edu/aimath06/proceedings/P21.pdf.

Xingyu Lin, Carl Qi, Yunchu Zhang, Zhiao Huang, Katerina Fragkiadaki, Yunzhu Li, Chuang Gan, and David Held. Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=tyxyBj2w4vw.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction, 2018. URL https://arxiv.org/abs/1802.03426.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34866–34891, 2023.

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

Seohong Park, Deepinder Mann, and Sergey Levine. Dual goal representations, 2025b. URL https://arxiv.org/abs/2510.06714.

Ronald Parr and Stuart J. Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, volume 10, 1997.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Max Rudolph, Caleb Chuck, Kevin Black, Misha Lvovsky, Scott Niekum, and Amy Zhang. Learning action-based representations using invariance. *Reinforcement Learning Journal*, 1:342–365, 2024.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

Tatsuki Tsujimoto. mjswan: MuJoCo simulation on web assembly with neural networks. URL https://github.com/ttktjmt/mjswan.

Toshimitsu Uesaka, Taiji Suzuki, Yuhta Takida, Chieh-Hsin Lai, Naoki Murata, and Yuki Mitsufuji. Weighted point set embedding for multimodal contrastive learning toward optimal similarity metric, 2025. URL https://arxiv.org/abs/2404.19228.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018. URL https://arxiv.org/abs/1807.03748.

Vivienne Huiling Wang, Tinghuai Wang, and Joni Pajarinen. Hierarchical reinforcement learning with uncertainty-guided diffusional subgoals. In *Forty-Second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=1YOYA2zN1j.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21579–21590, 2020.

A Training Details

Symmetric InfoNCE Loss. In practice, we use the normalized dot product as our score function,

$$f_{ij} = \left\langle \frac{\phi(s^i, g^i)}{\|\phi(s^i, g^i)\|_2}, \frac{\psi(\mathbf{a}^j)}{\|\psi(\mathbf{a}^j)\|_2} \right\rangle,$$

for a batch of N positive pairs $\{(s^i, g^i, \mathbf{a}^i)\}_{i=1}^N$. The symmetric InfoNCE loss averages two cross-entropy terms. The first term classifies the correct action sequence for each state-goal pair, and the second classifies the correct state-goal pair for each action sequence:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{2N} \sum_{i=1}^{N} \left[\log \frac{\exp(f_{ii}/\tau)}{\sum_{j=1}^{N} \exp(f_{ij}/\tau)} + \log \frac{\exp(f_{ii}/\tau)}{\sum_{j=1}^{N} \exp(f_{ji}/\tau)} \right].$$

We normalize the embeddings so that the objective cannot be trivially minimized by scaling their norms. The temperature τ then directly controls how sharp the softmax is.

Goal Sampling. Given a k-step triplet $(s_t, \mathbf{a}_k, s_{t+k})$ sampled from our dataset, we construct positive pairs by uniformly sampling goals from $s_{t+1}, \ldots, s_{t+k}$. In other words, we obtain positive pairs that consist of (s_t, s_{t+m}) and $\mathbf{a}_k$, where $1 \leq m \leq k$. This allows us to capture state-goal equivalences for all goals reachable within k steps from a state, aligning more closely with our theoretical formulation. In addition, this allows us to capture richer temporal structure in the environment, since skills at different horizons may become useful for the high-level policy. The benefits of this strategy are further explored in Appendix C.2.

Action Sequence Striding. For humanoidmaze, we found that long action sequences and large action spaces presented computational challenges due to the memory required to store long sequences of high-dimensional actions. As a result, we introduced an action stride hyperparameter, which avoids encoding the entire action sequence. We empirically selected an action stride of 4 for humanoid tasks because we found that this satisfied our computational budget constraints and minimally changed the action sequence. We additionally note that this minimally affects downstream performance, likely because the strided action sequence captures most of the structure in the full action sequence.

Action Sequence Truncation. An edge case arises when constructing state-goal pairs near the end of a trajectory, where the full k-step action sequence and goal are unavailable. In this case, we pair the state with the last available goal in the trajectory and pad the action sequence to length k by repeating the final action.

Auxiliary Loss Weighting. When combining our method with HIQL and HGCBC, we introduce a hyperparameter λ_{aux}, which weights the policy and value losses against our added objective, $\mathcal{L}_{\text{aux}}$. Specifically, the final loss is calculated as

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{aux}})(\mathcal{L}_{\text{value}} + \mathcal{L}_{\text{high}} + \mathcal{L}_{\text{low}}) + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}.$$

Here, $\mathcal{L}_{\text{low}}$ refers to the low-level policy loss, $\mathcal{L}_{\text{high}}$ refers to the high-level policy loss, and $\mathcal{L}_{\text{value}}$ refers to the value function loss. $\mathcal{L}_{\text{aux}}$ refers to the auxiliary loss applied (which is $\mathcal{L}_{\text{InfoNCE}}$ for CARL). We report the λ_{aux} values we select empirically in Table 3.

B Implementation Details

We base our implementation of CARL on OGBench (Park et al., 2025a). We run our experiments on an internal GPU cluster composed of Quadro RTX 5000 GPUs with around 16 GB of memory. Similar to the times reported by Park et al. (2025a), each run typically takes around 4 hours (state-based tasks) or around 10 hours (pixel-based tasks) when running HIQL+CARL and HGCBC+CARL. We estimate that around 10k GPU-hours were required in total for all experiments, plus additional hours for preliminary experiments.

Table 3: Hyperparameters used for CARL and the other algorithms in this paper. "Prediction" refers to parameters used for both Single-Action and Multi-Action Prediction.

Hyperparameter	**Value**
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2015)
# gradient steps	1,000,000 (states), 500,000 (pixels)
Minibatch size	1024 (states), 256 (pixels)
MLP dimensions	(512, 512, 512)
Representation architecture (pixel-based)	IMPALA CNN (Espeholt et al., 2018)
Image augmentation probability	0.5 (pixel-based manipulation), 0 (others)
AWR temperature	3.0
State-goal MLP dimensions $\phi(s, g)$	(512, 512, 512)
Action sequence MLP dimensions $\psi(\mathbf{a})$	(256, 256)
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Target smoothing coefficient	0.005
Discount factor γ	0.995 ({antmaze, pointmaze}-giant, humanoidmaze), 0.99 (others)
Expectile κ	0.7
Subgoal step	100 (humanoidmaze), 25 (other locomotion), 10 (others)
Representation horizon k	100 (humanoidmaze), 25 (other locomotion), 10 (others)
Action stride	4 (humanoidmaze), 1 (others)
Representation dimension	100
Single-Action CARL auxiliary loss weight λ_{aux}	0.1 (humanoidmaze), 0.5 (other locomotion), 0.1 (others)
CARL auxiliary loss weight λ_{aux}	0.1 (humanoidmaze), 0.3 (other locomotion), 0.7 (others)
Prediction auxiliary loss weight λ_{aux}	0.1 (humanoidmaze), 0.7 (other locomotion), 0.9 (others)
Single-Action CARL temperature τ	0.1 (locomotion), 0.05 (others)
CARL temperature τ	0.1 (locomotion), 0.05 (others)
Evaluation episodes	20

B.1 Hyperparameters

We base our implementation of HIQL+CARL on the implementation of HIQL in the OGBench codebase (Park et al., 2025a). To ensure fair comparisons, we keep most hyperparameters the same when integrating CARL with HIQL and HGCBC. One notable exception is the representation dimension: HIQL and HGCBC originally use representation dimensions of 10, but we change this to 100 in their CARL variants. For a fair comparison, we compare HIQL+CARL to HIQL with a larger representation dimension in Table 6. We find that HIQL's performance does not change significantly overall, with effects varying by environment: increasing the dimension causes regressions on pointmaze tasks while substantially improving performance on puzzle-3x3. Therefore, we use the original hyperparameters from Park et al. (2023) in our main benchmark.

Our auxiliary loss weights λ_{aux} and temperatures τ were tuned for all algorithms through hyperparameter sweeps of equal effort.

For a full description of the hyperparameters used in our training, refer to Table 3.

B.2 Humanoid Visualization Details

For qualitative humanoid evaluations, we use a custom tool to visualize rollouts of trajectories associated with state-goal pairs in our dataset. We used Muwanx (Tsujimoto)[4], a project combining MuJoCo, WebAssembly, Three.js, and ONNX to bring live MuJoCo simulation to web browsers. Building on this tool, we added various UI features and visualizations suited to our experiments.

[4]Muwanx GitHub: https://github.com/ttktjmt/muwanx

To perform the evaluation, we first find all of the nearest neighbors for a reference state-goal pair. We embed every k-step segment in our dataset and bin the segments into sequential groups of 500. From these bins, we select the best matches to the reference state-goal pair, and then select the top 30. Because the state-goal pairs in the humanoidmaze dataset span a variety of global (x, y) locations, we center them to visualize relative differences in joint behavior.

In Figure 4, we inspect nearest neighbors for different encoders by comparing k-step trajectories to a reference behavior. These snapshots show that CARL clusters behaviors better than a random encoder. We provide videos on our website for a better viewing experience: https://sites.google.com/view/behavior-rep/home.

C Additional Experiments

C.1 Pre-training vs. Co-training in HIQL

Table 4: Success rates (%) comparing co-training (jointly trained representations) against pre-training (frozen representations). Results are reported as mean ± 95% confidence interval; values within 95% of the best per row are **bolded**.

Task	Co-train	Pre-train
pointmaze-medium	**84.4 ± 3.1**	75.0 ± 6.7
pointmaze-large	**75.4 ± 3.1**	67.0 ± 6.6
pointmaze-giant	**64.5 ± 15.8**	48.5 ± 7.6
pointmaze-teleport	**31.4 ± 14.0**	18.0 ± 5.2
antmaze-medium	**97.9 ± 0.9**	**97.2 ± 2.3**
antmaze-large	**91.9 ± 2.4**	**92.8 ± 2.4**
antmaze-giant	**75.2 ± 3.8**	68.2 ± 5.0
antmaze-teleport	41.4 ± 2.1	**50.5 ± 2.0**
humanoidmaze-medium	**90.5 ± 2.4**	66.8 ± 4.0
humanoidmaze-large	**58.3 ± 3.8**	17.2 ± 1.4
humanoidmaze-giant	**27.2 ± 3.1**	4.0 ± 2.0
antsoccer-medium	**12.8 ± 2.4**	10.8 ± 4.2
antsoccer-arena	**63.7 ± 3.1**	58.5 ± 4.7
cube-single	**32.8 ± 3.1**	20.5 ± 6.0
cube-double	**23.4 ± 4.5**	8.2 ± 3.1
scene	**70.5 ± 5.2**	49.5 ± 7.2

We explored two ways of integrating CARL into HIQL: pre-training, where the subgoal representation is learned with CARL and then fine-tuned with low-level policy gradients from HIQL, and co-training, where CARL's objective is optimized jointly with HIQL's value and policy losses. Pre-training keeps CARL isolated from HIQL's value learning—the encoder shapes the low-level policy's input but does not interact with the value function. Co-training instead shares the subgoal representation across the low-level policy and value function, fully integrating skill reuse into the algorithm. Pre-training alone improved performance on most state-based tasks but regressed on humanoidmaze tasks. Co-training, by contrast, outperforms pre-training on 14 of 16 tasks (Table 4), with the largest gains on longer-horizon tasks; pre-training is comparable on `antmaze-large` and better only on `antmaze-teleport`. We attribute this to the shared subgoal representation benefiting both the policy and value function, and view it as further evidence for the importance of integrated skill-reuse abstractions in HRL.

C.2 Surface Sampling vs. Interior Sampling

Table 5: Success rates (%) comparing CARL representations trained with interior and surface sampling. Results are reported as mean ± 95% confidence interval over 8 seeds. Win rates for each sampling technique are reported, with the best win rate bolded.

Task	Interior	Surface
pointmaze-medium	**84.4 ± 3.1**	71.4 ± 5.9
pointmaze-large	**75.4 ± 3.1**	9.9 ± 9.7
pointmaze-giant	**64.5 ± 15.8**	7.2 ± 5.9
pointmaze-teleport	**31.4 ± 14.0**	26.7 ± 6.1
antmaze-medium	**97.9 ± 0.9**	**95.0 ± 4.7**
antmaze-large	**91.9 ± 2.4**	**92.4 ± 3.1**
antmaze-giant	**75.2 ± 3.8**	**71.8 ± 6.4**
antmaze-teleport	**41.4 ± 2.1**	**39.7 ± 5.7**
humanoidmaze-medium	**90.5 ± 2.4**	**89.3 ± 3.1**
humanoidmaze-large	**58.3 ± 3.8**	52.6 ± 5.7
humanoidmaze-giant	**27.2 ± 3.1**	21.5 ± 3.3
antsoccer-medium	**12.8 ± 2.4**	11.4 ± 2.4
antsoccer-arena	**63.7 ± 3.1**	56.9 ± 4.0
cube-single	**32.8 ± 3.1**	28.5 ± 3.8
cube-double	**23.4 ± 4.5**	15.7 ± 4.3
cube-triple	**15.2 ± 3.5**	9.9 ± 4.0
cube-quadruple	0.1 ± 0.2	**0.3 ± 0.5**
puzzle-3x3	**45.5 ± 7.3**	**47.9 ± 8.0**
puzzle-4x4	**35.2 ± 7.1**	**38.2 ± 5.0**
puzzle-4x5	4.2 ± 2.4	**6.5 ± 2.1**
puzzle-4x6	3.6 ± 2.1	**4.0 ± 1.4**
scene	**70.5 ± 5.2**	**71.6 ± 3.5**
win-rate	**15/22**	7/22

Given a tuple $(s_t, \mathbf{a}_k, s_{t+k})$, one simple sampling technique would be to select (s_t, s_{t+k}) and $\mathbf{a}_k$ as a positive pair. We call this surface sampling, because this technique samples state-goal pairs that lie on the surface of a k-step walk from s_t. However, we could also form positive pairs for (s_t, s_{t+m}), where $1 \leq m \leq k$, because these shorter state pairs are still achieved by a portion of the longer action sequence. This also benefits the low-level policy, which may have to reason about state-goal pairs that are fewer than k steps apart. We call this interior sampling, because it samples state-goal pairs that lie in the interior of the k-step walk from s_t. Empirically, we find that interior sampling performs better on average, obtaining a win rate of 15/22, as shown in Table 5.

C.3 Training HIQL with Larger Representation Dimensions

Table 6: Success rates (%) comparing CARL-augmented HRL algorithms against their 100-dimensional representation variants and default variants. Results are reported as mean ± 95% confidence interval over 8 seeds, with performance within 95% of the best **bolded**. For win-rate calculations, we ignore ties and **bold** the highest win rate.

	HIQL Variants			HGCBC Variants		
Task	HIQL+CARL	HIQL (rep_dim=100)	HIQL	HGCBC+CARL	HGCBC (rep_dim=100)	HGCBC
pointmaze-medium	**84.4 ± 3.1**	65.5 ± 7.3	79 ± 4.3	**13.6 ± 1.8**	0.0 ± 0.0	0.0 ± 0.0
pointmaze-large	**75.4 ± 3.1**	60.1 ± 10.2	58 ± 4.3	**19.5 ± 3.8**	0.3 ± 0.2	0.4 ± 0.3
pointmaze-giant	**64.5 ± 15.8**	21.7 ± 6.9	46 ± 7.6	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
pointmaze-teleport	**31.4 ± 14.0**	16.0 ± 4.5	18 ± 3.3	**29.7 ± 1.8**	12.6 ± 1.1	16.7 ± 1.2
antmaze-medium	**97.9 ± 0.9**	**96.5 ± 1.7**	**96 ± 0.9**	**73.1 ± 1.5**	60.7 ± 2.2	59.8 ± 1.6
antmaze-large	**91.9 ± 2.4**	**89.9 ± 3.3**	**91 ± 1.7**	**64.9 ± 0.9**	48.0 ± 0.9	59.3 ± 1.9
antmaze-giant	**75.2 ± 3.8**	68.8 ± 5.4	65 ± 4.3	**18.4 ± 1.3**	4.4 ± 1.1	8.5 ± 0.8
antmaze-teleport	41.4 ± 2.1	**45.6 ± 6.6**	42 ± 2.6	**38.5 ± 1.3**	34.6 ± 2.1	**36.9 ± 1.9**
humanoidmaze-medium	**90.5 ± 2.4**	**87.6 ± 4.5**	**89 ± 1.7**	**44.4 ± 1.3**	40.8 ± 1.8	32.4 ± 0.9
humanoidmaze-large	**58.3 ± 3.8**	52.9 ± 5.2	49 ± 3.3	**31.0 ± 0.8**	19.8 ± 1.7	22.6 ± 0.8
humanoidmaze-giant	**27.2 ± 3.1**	22.8 ± 4.5	12 ± 3.3	**20.0 ± 1.1**	12.0 ± 1.3	9.5 ± 1.3
antsoccer-medium	**12.8 ± 2.4**	11.7 ± 2.8	**13 ± 1.7**	**7.0 ± 0.4**	4.9 ± 1.0	5.9 ± 0.7
antsoccer-arena	**63.7 ± 3.1**	57.9 ± 3.1	58 ± 1.7	**24.9 ± 1.6**	15.2 ± 1.2	12.5 ± 0.8
cube-single	**32.8 ± 3.1**	19.9 ± 4.0	15 ± 2.6	**6.9 ± 0.7**	5.6 ± 1.1	4.4 ± 0.8
cube-double	**23.4 ± 4.5**	4.5 ± 2.4	6 ± 1.7	**1.3 ± 0.3**	**1.3 ± 0.5**	1.1 ± 0.3
cube-triple	**15.2 ± 3.5**	3.3 ± 1.2	3 ± 0.9	**2.1 ± 0.6**	0.0 ± 0.0	0.4 ± 0.3
cube-quadruple	0.1 ± 0.2	**0.3 ± 0.5**	0 ± 0.0	**0.1 ± 0.1**	0.0 ± 0.0	0.0 ± 0.0
puzzle-3x3	45.5 ± 7.3	**63.4 ± 8.3**	12 ± 1.7	**6.0 ± 0.9**	4.0 ± 0.8	4.1 ± 0.3
puzzle-4x4	**35.2 ± 7.1**	4.3 ± 2.6	7 ± 1.7	**5.9 ± 0.9**	0.3 ± 0.2	0.3 ± 0.2
puzzle-4x5	4.2 ± 2.4	**5.2 ± 1.9**	4 ± 0.9	**1.6 ± 0.6**	1.1 ± 0.5	0.4 ± 0.2
puzzle-4x6	**3.6 ± 2.1**	2.5 ± 1.7	3 ± 0.9	**0.6 ± 0.3**	**0.6 ± 0.2**	0.3 ± 0.2
scene	**70.5 ± 5.2**	41.0 ± 2.6	38 ± 2.6	**15.8 ± 1.0**	7.7 ± 1.3	8.5 ± 1.1
win-rate	**17/22**	4/22	1/22	**19/19**	0/19	0/19

We use a different representation dimension than Park et al. (2023), who do not ablate this choice. To verify that our gains do not come from the change in representation dimension alone, Table 6 compares our CARL-augmented HRL algorithms against both their default versions and variants with a 100-dimensional representation. Increasing the representation dimension alone yields mixed results: HIQL (`rep_dim`=100) degrades on most pointmaze tasks and on `puzzle-4x4`, improves substantially on `puzzle-3x3` and `humanoidmaze-giant`, and changes little elsewhere. This confirms that increased capacity alone cannot explain our performance gains—CARL's representation structure is essential for the consistent improvements we see across nearly every task.

C.4 k-STEP SENSITIVITY ANALYSIS

Table 7: Sensitivity to horizon length k on locomotion environments. Success rates (%) averaged over 8 seeds with 95% confidence intervals reported. Results within 95% of the best are **bolded**.

Environment	k=5	k=10	k=25	k=50	k=75
pointmaze-medium	76.50 ± 10.8	$\mathbf{80.75 \pm 3.2}$	$\mathbf{84.00 \pm 0.7}$	$\mathbf{81.00 \pm 3.4}$	72.75 ± 4.9
pointmaze-large	45.75 ± 17.7	61.00 ± 13.2	70.00 ± 6.8	$\mathbf{88.25 \pm 5.5}$	$\mathbf{88.75 \pm 4.8}$
pointmaze-giant	22.50 ± 15.4	47.50 ± 14.1	62.50 ± 13.3	67.75 ± 12.4	$\mathbf{72.25 \pm 9.5}$
pointmaze-teleport	24.00 ± 8.5	26.50 ± 4.0	28.50 ± 7.8	27.75 ± 6.8	$\mathbf{36.25 \pm 6.4}$
antmaze-medium	89.75 ± 0.8	$\mathbf{95.75 \pm 1.4}$	$\mathbf{96.50 \pm 1.1}$	90.75 ± 2.5	90.75 ± 3.4
antmaze-large	51.75 ± 3.2	74.75 ± 1.4	$\mathbf{86.50 \pm 6.5}$	$\mathbf{86.00 \pm 3.5}$	68.25 ± 1.7
antmaze-giant	3.75 ± 2.1	38.75 ± 6.6	$\mathbf{73.00 \pm 2.5}$	57.75 ± 4.7	26.25 ± 6.4
antmaze-teleport	36.25 ± 4.5	41.50 ± 6.1	40.75 ± 2.7	$\mathbf{48.00 \pm 3.1}$	43.25 ± 5.8
antsoccer-medium	6.25 ± 2.8	4.75 ± 2.5	$\mathbf{12.25 \pm 1.7}$	$\mathbf{12.25 \pm 1.1}$	6.50 ± 0.5
antsoccer-arena	$\mathbf{58.75 \pm 6.2}$	53.25 ± 3.6	$\mathbf{58.75 \pm 5.6}$	$\mathbf{60.00 \pm 6.3}$	$\mathbf{59.00 \pm 4.0}$

Table 8: Sensitivity to horizon length k on humanoidmaze environments. Success rates (%) averaged over 8 seeds with 95% confidence intervals reported. Results within 95% of the best are **bolded**.

Environment	k=25	k=75	k=100	k=125	k=150
humanoidmaze-medium	$\mathbf{88.00 \pm 4.4}$	$\mathbf{85.00 \pm 1.7}$	$\mathbf{87.00 \pm 3.0}$	$\mathbf{86.67 \pm 5.4}$	$\mathbf{87.33 \pm 4.6}$
humanoidmaze-large	44.67 ± 5.6	$\mathbf{48.67 \pm 2.7}$	$\mathbf{48.50 \pm 4.8}$	$\mathbf{49.33 \pm 6.2}$	$\mathbf{50.00 \pm 5.2}$
humanoidmaze-giant	18.00 ± 4.7	$\mathbf{26.67 \pm 2.4}$	22.75 ± 3.9	$\mathbf{28.00 \pm 5.9}$	21.67 ± 0.5

Table 9: Sensitivity to horizon length k on manipulation environments. Success rates (%) averaged over 8 seeds with 95% confidence intervals reported. Results within 95% of the best are **bolded**.

Environment	k=3	k=5	k=10	k=15	k=30
cube-single	27.50 ± 7.8	29.25 ± 3.8	$\mathbf{36.00 \pm 3.1}$	31.00 ± 1.0	30.75 ± 5.2
cube-double	11.50 ± 7.0	15.75 ± 3.0	$\mathbf{21.00 \pm 5.2}$	$\mathbf{21.00 \pm 5.1}$	9.75 ± 0.4
cube-triple	1.25 ± 0.8	3.25 ± 2.2	5.75 ± 2.4	8.75 ± 1.4	$\mathbf{10.50 \pm 4.0}$
cube-quadruple	0.00 ± 0.0	0.00 ± 0.0	$\mathbf{0.25 \pm 0.4}$	0.00 ± 0.0	$\mathbf{0.25 \pm 0.4}$
scene	$\mathbf{74.75 \pm 6.5}$	66.75 ± 1.3	70.75 ± 1.7	71.00 ± 7.5	50.25 ± 3.6
puzzle-3x3	8.50 ± 2.2	11.75 ± 3.0	31.75 ± 5.5	$\mathbf{50.50 \pm 10.7}$	40.00 ± 12.0
puzzle-4x4	31.00 ± 1.2	32.00 ± 3.6	34.00 ± 1.2	$\mathbf{38.00 \pm 2.6}$	29.25 ± 2.8
puzzle-4x5	6.00 ± 1.7	$\mathbf{6.75 \pm 1.6}$	3.75 ± 0.4	4.00 ± 2.0	2.75 ± 0.8
puzzle-4x6	$\mathbf{4.75 \pm 2.5}$	4.25 ± 1.3	3.25 ± 1.3	3.25 ± 1.6	1.75 ± 1.3

We provide a sensitivity analysis for our k-step hyperparameter in Table 7 for locomotion environments (except humanoidmaze), Table 8 for humanoidmaze, and Table 9 for manipulation environments. For some environments, we observe better performance with larger k (pointmaze-large, pointmaze-giant, and humanoidmaze-large). We also observe the opposite trend, where performance drops off when k is increased too much (puzzle-3x3, cube-double, and scene). Other environments are largely robust to the choice of horizon k.

C.5 Data Sensitivity Analysis

Table 10: Robustness to reduced dataset coverage. Success rates (%) are reported as mean ± 95% confidence interval over 3 seeds. Results within 95% of the best in each row are **bolded**.

Method	Environment	0% Coverage	25% Coverage	50% Coverage	75% Coverage
HIQL+CARL	pointmaze-medium	10.7 ± 30.5	76.3 ± 12.9	80.7 ± 3.0	$\mathbf{85.3 \pm 14.6}$
	antmaze-medium	11.0 ± 24.1	$\mathbf{95.7 \pm 1.3}$	$\mathbf{99.7 \pm 1.3}$	$\mathbf{98.0 \pm 5.2}$
	humanoidmaze-medium	40.0 ± 9.0	$\mathbf{92.7 \pm 11.2}$	$\mathbf{88.3 \pm 5.2}$	$\mathbf{89.3 \pm 1.3}$
HIQL	pointmaze-medium	12.7 ± 26.7	62.7 ± 28.4	$\mathbf{72.0 \pm 25.0}$	$\mathbf{69.7 \pm 12.0}$
	antmaze-medium	0.0 ± 0.0	$\mathbf{93.0 \pm 6.5}$	$\mathbf{96.7 \pm 3.9}$	$\mathbf{95.0 \pm 4.3}$
	humanoidmaze-medium	33.0 ± 19.8	$\mathbf{89.0 \pm 13.3}$	$\mathbf{85.0 \pm 9.9}$	$\mathbf{89.3 \pm 1.3}$

Table 11: Robustness to dataset imbalance induced by removing action sequences in the left half of the maze that move downwards. Success rates (%) are reported as mean ± 95% confidence interval over 3 seeds. Results within 95% of the best in each row are **bolded**.

Method	Environment	25% Removed	50% Removed	75% Removed	100% Removed
HIQL+CARL	pointmaze-medium	$\mathbf{81.0 \pm 1.7}$	$\mathbf{81.0 \pm 2.6}$	59.7 ± 18.1	19.7 ± 1.3
	antmaze-medium	$\mathbf{99.3 \pm 1.3}$	$\mathbf{96.3 \pm 7.3}$	$\mathbf{96.7 \pm 3.0}$	18.0 ± 0.0
	humanoidmaze-medium	$\mathbf{90.3 \pm 7.3}$	$\mathbf{89.0 \pm 4.3}$	$\mathbf{86.7 \pm 1.3}$	16.0 ± 5.2
HIQL	pointmaze-medium	$\mathbf{66.0 \pm 5.2}$	$\mathbf{63.7 \pm 16.8}$	36.3 ± 11.6	19.0 ± 4.3
	antmaze-medium	$\mathbf{96.0 \pm 2.6}$	$\mathbf{95.3 \pm 8.6}$	$\mathbf{95.3 \pm 1.3}$	15.0 ± 4.3
	humanoidmaze-medium	$\mathbf{88.3 \pm 5.2}$	$\mathbf{87.0 \pm 9.9}$	$\mathbf{84.0 \pm 6.5}$	21.0 ± 4.3

In this section, we analyze the sensitivity of CARL to two types of data imbalance. First, to test performance under low coverage, we remove portions of the data from the left half of the maze. Specifically, we retain 0, 25, 50, and 75% of the data in the left half of the maze, measure the success rate of a policy learned with HIQL+CARL under these conditions, and compare it to the performance of HIQL. We report results in Table 10 and find that HIQL+CARL shows a drop in performance similar to that of HIQL as coverage decreases. This suggests that CARL improves the performance of HIQL without making data sensitivity worse than the baseline.

Second, we test performance under dataset imbalance. We remove a percentage of the action sequences in the left half of the maze that move downwards, biasing the action sequence distribution. We report results for this experiment in Table 11. As in the first case, we find that HIQL+CARL's drop in performance as dataset imbalance increases is very similar to that of HIQL. The similar robustness trends between HIQL+CARL and HIQL suggest that dataset quality impacts performance primarily through the core HRL components rather than the representation. While data diversity and coverage should in principle affect CARL's performance, these results indicate that CARL does not further exacerbate existing data sensitivity.