MSPR: MULTI-SCALE PREDICTIVE REPRESENTATIONS FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper investigates robust representation learning in offline goal-conditioned reinforcement learning (GCRL). Particularly in sparse reward scenarios, learning representations that align state and goal latents is a challenge, as the encoder can learn goal-agnostic features that destabilize policy learning. We address this issue by learning the encoder's representation with alignment objectives that capture the environment across multiple scales, from local physical dynamics to long-horizon goal-directed structure. Concretely, we propose *MSPR*, a framework that leverages multi-scale predictive supervision to enforce goal-directed alignment within the latent space. We demonstrate that *MSPR* leads to strong performance on both vision and state-based tasks. Furthermore, we show that our approach is resilient under realistic, challenging data regimes, maintaining state-of-the-art performance across a wide variety of tasks.

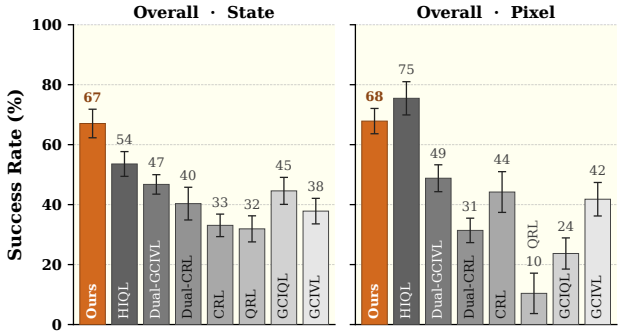


Figure 1: **Overview.** Average success on 12 state-based and 5 pixel-based OGBench tasks spanning locomotion and manipulation. *MSPR* achieves the best average on state-based tasks and is second only to the HIQL on pixel-based tasks.

1 INTRODUCTION

Advancements in representation learning have expanded the capabilities of deep reinforcement learning (RL). Self-supervised objectives (Schwarzer et al., 2021; Srinivas et al., 2020) and predictive world models (Hafner et al., 2020; Hansen et al., 2023) enable agents to capture underlying environment dynamics rather than memorizing trajectories (Schwarzer et al., 2021; Srinivas et al., 2020). Yet deploying RL in real-world scenarios exposes a critical bottleneck: dense, task-specific rewards are notoriously difficult to design and collect. Agents must increasingly learn from offline datasets guided only by sparse, task-agnostic success signals.

Goal-conditioned RL (GCRL) provides a principled framework for this paradigm, tasking an agent to reach any specified goal from any starting state (Kaelbling, 1993; Park et al., 2024a). Yet GCRL introduces profound representation challenges (Liu et al., 2022). To succeed, an agent must align its state and goal representations to understand not just what actions are physically possible, but how those actions contribute to maximizing the goal-conditioned cumulative reward. Under sparse rewards, standard methods lack the gradient feedback required to enforce this alignment, resulting in encoders that produce goal-agnostic features, leading to value overestimation, poor trajectory stitching, and brittle performance.

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.

To learn effective goal-aware representations, we argue that following three key conditions can be beneficial. 1. *Dynamical alignment* requires the encoder to capture immediate transition dynamics, grounding the agent in what actions are physically feasible. 2. *Behavioral alignment* requires the encoder to organize state-goal pairs such that goal-directed actions and successor states are predictable from their latent representations. 3. *Temporal alignment* requires the latent geometry to reflect goal-conditioned returns, providing the critic with a pre-organized substrate rather than requiring it to discover this structure from sparse rewards alone.

Existing GCRL methods satisfy at most two of these conditions. Model-free methods such as GCBC (Lynch et al., 2019), GCIVL (Park et al., 2024b), and GCIQL (Kostrikov et al., 2021) prioritize behavioral relations but bypass physical and temporal dynamics entirely. Representation-based methods such as Dual (Park et al., 2025b) and VIP (Ma et al., 2023) target temporal alignment by leveraging value-based objectives and reward-distance estimation, respectively, where VIP specifically treats the distance between state and goal embeddings as a reward signal. However, they lack physical grounding and become brittle under high-dimensional observations or suboptimal data.

To address these limitations, we introduce Multi-scale Predictive Representations (*MSPR*), a unified framework that jointly enforces all three alignment conditions. *MSPR* operates at two complementary temporal granularities: at the local scale, it predicts immediate single-step transitions, grounding the encoder in physical dynamics; and at the global scale, it predicts goal-conditioned transitions and actions toward distant objectives, aligning the encoder with goal-directed intent. An end-to-end actor-critic agent trained on this structured representation benefits from a latent space where value approximation is substantially simplified. *MSPR* is a flat, single-policy method: it introduces no subgoal generator, so hierarchical decomposition remains an orthogonal extension.

Contribution. We propose an end-to-end, multi-scale predictive framework that satisfies all three conditions. Unlike prior representation learning approaches that suffer from algorithmic brittleness (Park et al., 2025b; Lawson et al., 2025; Ma et al., 2023; Alemi et al., 2019; Myers et al., 2025) , *MSPR* constructs a robust latent space that maintains stability across diverse tasks. On the challenging OGBench benchmark, *MSPR* achieves an average success rate of **67%** on state-based and **68%** on pixel-based tasks, improving over the prior representation learning baseline ($\text{Dual}_{\text{GCIVL}}$) by 20 points (Figure 1). Notably, *MSPR* outperforms the hierarchical state-of-the-art HIQL (Park et al., 2024c) on state-based environments (54%) without any hierarchy. *MSPR* also remains effective when training data are scarce (Sec 5.2),

2 Related Work

Goal Representations for Offline GCRL. Representation learning in goal-conditioned RL (GCRL) seeks latent spaces that encode task progress and environment geometry (Kaelbling, 1993; Lynch et al., 2019). Prior approaches predominantly construct goal representations via temporal-distance metrics, contrastive objectives, or value-derived embeddings, such as Dual Goal Representations (Park et al., 2025b), VIP (Ma et al., 2023), QRL (Wang et al., 2023), HILP (Park et al., 2024d), CRL (Eysenbach et al., 2023), and state-abstraction methods (Li et al., 2006; Castro, 2019). However, because distance-based representations are coupled to or downstream of value estimation, errors in critic updates, particularly under high-dimensional observations, noisy offline data, and trajectory stitching, propagate directly back to the latent space, degrading downstream policy extraction (Lei et al., 2025; Yang et al., 2022). In contrast, *MSPR* supervises the encoder with self-contained predictive targets decoupled from critic gradients, preventing critic error propagation and enabling representations to survive severe data degradation (Sec-5).

Predictive Representations in RL. Predictive auxiliary tasks and latent world models (Gelada et al., 2019; Schwarzer et al., 2021; Fujimoto et al., 2025; Hafner et al., 2022; 2023; Hansen et al., 2023; Ha & Schmidhuber, 2018; Hafner et al., 2020; 2019; Schrittwieser et al., 2020; Srinivas et al., 2020) have established that forward-dynamics and reward prediction structure latent spaces effectively for single-task or dense-reward RL. Leveraging system dynamics is a foundational approach for shaping representations in complex, partially observable environments (Littman et al., 2001; Parr et al., 2008). Moreover, existing latent-MDP guarantees (e.g., Theorem 1 & 2 of MR.Q (Fujimoto et al., 2025)) require only transition and reward sufficiency for a single-task policy $\pi(a|z_s)$, whereas goal-conditioned control demands joint encoding of state-goal pairs (s, g) such that goal-reaching actions remain recoverable. Proposition 3 states this goal-conditioned behavioral alignment condition.

Thus, rather than introducing new head architectures, our contribution establishes the predictive conditions required specifically for the sparse offline GCRL setting.

3 PRELIMINARY

Offline Goal-Conditioned RL. We consider offline goal-conditioned reinforcement learning (GCRL) in an environment modeled as a goal-conditioned Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mathcal{G})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s,a)$ describes transition dynamics, $\gamma \in (0,1)$ is the discount factor, and $\mathcal{G} \subseteq \mathcal{S}$ is the goal space. The agent learns from a static dataset of transition tuples $\mathcal{D} = \{(s, a, s', g)\}$, augmented via hindsight goal relabeling following standard OGBench protocols (Park et al., 2024a). The reward function $r(s,a,g)$ is sparse binary: $r(s,a,g) = 0$ if the state reaches goal g, and -1 otherwise. The objective is to learn a policy $\pi(a|s,g)$ that maximizes the expected cumulative discounted return $J(\pi) = \mathbb{E}_{\pi,\mathcal{P}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g)]$. In the rest of the manuscript, timestep subscripts are generally omitted to simplify the notation; instead, we use (s, s') to indicate the current and next states. To model long-horizon progress within trajectory rollouts, we define the empirical Monte-Carlo return along a dataset sequence as $r_{\text{MC}}(s,a,g) = \sum_{k=0}^{T-t} \gamma^k r(s_{t+k}, a_{t+k}, g)$. We write $Q^\beta(s,a,g) := \mathbb{E}_{\mathcal{D}}[\, r_{\text{MC}}(s,a,g) \mid s,a,g\,]$ for the conditional mean of the Monte-Carlo return under the relabeled dataset distribution; r_{MC} is an unbiased sample of this quantity, which in general differs from the optimal value Q^* (Sutton, 1988; Quillen et al., 2018).

4 METHOD

To construct a latent representation that captures both environment physics and long-horizon intent without relying on critic gradients, *MSPR* jointly enforces the three alignment conditions introduced in Sec-1. We realize these conditions through four auxiliary predictive heads: a forward dynamics head f_{dyn} implementing *dynamical alignment*, a return prediction head f_{rew} implementing *temporal alignment*, and twin heads $f_{\text{g-dyn}}$ and $f_{\text{g-act}}$ implementing *behavioral alignment*. Each head corresponds to one of the three conditions. In the following subsections, we state how each objective controls the error for the downstream actor-critic, which only observes $(\mathbf{z}_s, \mathbf{z}_{sa}, \mathbf{z}_g)$ and not the raw observations. The complete set of encoders, predictive heads, and RL components is summarized in Figure 2 (left).

Encoders	
State representation	$\mathbf{z}_s = \mathcal{E}^s_\psi(s)$
State-action representation	$\mathbf{z}_{sa} = \mathcal{E}^{sa}_\psi(\mathbf{z}_s, a)$
Goal representation	$\mathbf{z}_g = \mathcal{E}^s_\psi(g)$
Multi-scale Predictors	
Forward dynamics	$\tilde{\mathbf{z}}_{s'} = f_{\text{dyn}}(\mathbf{z}_{sa})$
Goal-level dynamics	$\tilde{\mathbf{z}}^{\text{goal}}_{s'} = f_{\text{g-dyn}}(\mathbf{z}_s, \mathbf{z}_g)$
Goal-conditioned action	$\tilde{\mathbf{a}}^{\text{goal}} = f_{\text{g-act}}(\mathbf{z}_s, \mathbf{z}_g)$
Reward predictor	$\tilde{r} = f_{\text{rew}}(\mathbf{z}_{sa}, \mathbf{z}_g)$
Goal-Conditioned RL	
Critic function	$Q_\theta(\mathbf{z}_{sa}, \mathbf{z}_g)$
Actor policy	$\pi_\phi(a|\mathbf{z}_s, \mathbf{z}_g)$

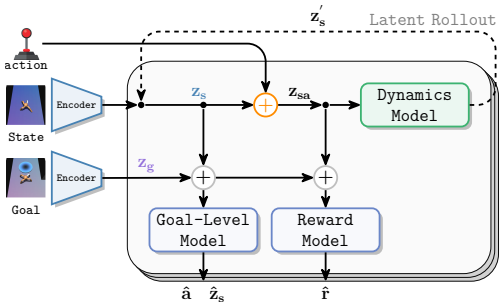


Figure 2: **Multi-scale Predictive Representations.** (*Left*) Summary of *MSPR* encoders, multi-scale predictors, and RL modules. (*Right*) Architectural overview illustrating self-supervised predictive representation learning decoupled from actor-critic optimization.

Population errors. Throughout, expectations are over $(s, a, s', g) \sim \mathcal{D}$ and $\mathbf{z}_{s'} = \mathcal{E}^s_{\bar{\psi}}(s')$ is the target-encoder representation. We work with expected values of the training losses; for the temporal head we take the best value achievable by *any* head, so that the quantity depends on the encoder alone:

$$\delta_{\text{dyn}} = \mathbb{E}\big[\|f_{\text{dyn}}(\mathbf{z}_{sa}) - \mathbf{z}_{s'}\|_2^2\big] \;=\; \mathbb{E}[\mathcal{L}_{\text{dyn}}], \tag{1}$$

$$\Delta_{\text{temp}} = \min_f \; \mathbb{E}\big[\big(r_{\text{MC}}(s,a,g) - f(\mathbf{z}_{sa}, \mathbf{z}_g)\big)^2\big] \;\le\; \mathbb{E}[\mathcal{L}_{\text{rew}}], \tag{2}$$

$$\delta_{\text{beh}} = \mathbb{E}\big[\|f_{\text{g-act}}(\mathbf{z}_s, \mathbf{z}_g) - a\|_2^2\big] \;=\; \mathbb{E}[\mathcal{L}_{\text{g-act}}]. \tag{3}$$

The minimum in equation 2 is over all measurable functions of $(\mathbf{z}_{sa}, \mathbf{z}_g)$; a trained f_{rew} attains at best this value, so reducing $\mathcal{L}_{\text{rew}}$ reduces an upper bound on Δ_{temp}.

4.1 Dynamical Alignment

Dynamical alignment grounds the representation space with environment physics, ensuring that the encoder preserves feature necessary for forward simulation under arbitrary actions.

Forward Dynamics (f_{dyn}). Given the current state-action latent representation $\mathbf{z}_{sa} = \mathcal{E}^{sa}_{\psi}(\mathbf{z}_s, a)$, the forward dynamics module predicts the subsequent latent state $\tilde{\mathbf{z}}_{s'}$. Minimizing forward prediction error enforces transition consistency. The objective is formulated as:

$$\mathcal{L}_{\text{dyn}} = \|\tilde{\mathbf{z}}_{s'} - \mathbf{z}_{s'}\|_2^2, \qquad \text{where} \quad \tilde{\mathbf{z}}_{s'} = f_{\text{dyn}}(\mathbf{z}_{sa}). \tag{4}$$

Stability via Target Encoder. Targets are computed by a target encoder $\mathcal{E}^{s}_{\bar{\psi}}$ with stopped gradients (Bardes et al., 2024), whose weights are copied from the online encoder every K steps (Algorithm 1). This ensures that the next-state latent is a stable regression target across updates, preventing the encoder from collapsing.

Proposition 1 (Latent transition consistency). *Let $V : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be any function that is L_V-Lipschitz in its first argument. Then*

$$\mathbb{E}\Big[\big(V(f_{\text{dyn}}(\mathbf{z}_{sa}), \mathbf{z}_g) - V(\mathbf{z}_{s'}, \mathbf{z}_g)\big)^2\Big] \;\le\; L_V^2\, \delta_{\text{dyn}}. \tag{5}$$

Take V to be the next-state value from which the critic bootstraps in Eq. equation 12. When δ_{dyn} is small, $\mathbf{z}_{sa}$ retains the information needed to predict the value of the state it leads to, which is exactly what a Bellman-consistent critic on $(\mathbf{z}_{sa}, \mathbf{z}_g)$ must be able to represent. When it is large, two state-action pairs with different successors may share a latent, and no critic on that latent can assign them different values. Note that the critic does not use f_{dyn} directly; the proposition only says that $\mathbf{z}_{sa}$ keeps the information needed to predict the next state's value.

4.2 Temporal Alignment

While dynamical alignment models local transitions, it provides no information regarding the cumulative cost of reaching a goal, information the critic cannot recover from sparse binary rewards alone. *Temporal alignment* addresses this by pre-organizing the latent space to reflect goal-conditioned return structure, reducing the burden on value estimation under sparse feedback.

Return Predictor (f_{rew}). Given the state-action representation $\mathbf{z}_{sa}$ and goal embedding $\mathbf{z}_g$, the return prediction module outputs a scalar prediction $\tilde{r} = f_{\text{rew}}(\mathbf{z}_{sa}, \mathbf{z}_g)$. Standard single-step reward targets carry no signal under sparse binary rewards ($r = 0$ at goal, -1 elsewhere), remaining constant across almost all offline transitions. We address this by training f_{rew} to predict Monte-Carlo returns r_{MC} computed over offline trajectories:

$$\mathcal{L}_{\text{rew}} = \|\tilde{r} - r_{\text{MC}}\|_2^2. \tag{6}$$

Because r_{MC} captures trajectory-level discounted returns, $\mathcal{L}_{\text{rew}}$ provides a dense, multi-step regression signal that organizes latent states by goal proximity. Although r_{MC} estimates the dataset return Q^{β} (Sec-3) rather than the optimal value, this dense ordering provides enough temporal structure to stabilize value learning, as the long-horizon locomotion results in Table 2 show.

With Q^{β} as defined in Sec-3, let $\sigma^2_{\text{MC}} := \mathbb{E}[(r_{\text{MC}} - Q^{\beta})^2]$ be the return variance that remains after conditioning on (s, a, g), a property of the data independent of the encoder, and $\varepsilon_V^{\min} := \min_Q \mathbb{E}\big[(Q(\mathbf{z}_{sa}, \mathbf{z}_g) - Q^{\beta}(s, a, g))^2\big]$ the smallest error that any critic on the latents can achieve against Q^{β}.

Proposition 2 (Temporal alignment controls the representational value error).

$$\Delta_{\text{temp}} \;=\; \sigma^2_{\text{MC}} \;+\; \varepsilon_V^{\min}. \tag{7}$$

Because σ^2_{MC} is fixed by the data, minimizing Δ_{temp} over the encoder minimizes $\varepsilon_V^{\min}$, the value error that no critic, however expressive, can remove because the latent has discarded the information. The error against the *optimal* value further splits into this representation term and a data term ε_{β}, the gap between Q^{β} and Q^{*}, which no encoder can reduce (Corollary 1). We expect ε_{β} to be larger on datasets whose trajectories rarely reach the goal, such as the `stitch` datasets (Appendix E.3). The condition binds only for learned, compressive encoders (Remark 1).

4.3 BEHAVIORAL ALIGNMENT

Dynamical and temporal alignment ground the encoder in physical transitions and goal proximity, but neither explicitly models the relationship between a current state and a target goal. *Behavioral alignment* bridges this gap by supervising the latent space with goal-conditioned predictions, coupling state and goal representations directly to the control objective.

Goal-level Dynamics ($f_{\text{g-dyn}}$). Given current state $\mathbf{z}_s$ and goal embedding $\mathbf{z}_g$, this module predicts the representation of the next state $\tilde{\mathbf{z}}^{\text{goal}}_{s'}$ implied by goal-directed behavior. This forces $(\mathbf{z}_s, \mathbf{z}_g)$ to jointly encode the expected physical transition toward goal g:

$$\mathcal{L}_{\text{g-dyn}} = \|\tilde{\mathbf{z}}^{\text{goal}}_{s'} - \mathbf{z}_{s'}\|_2^2, \qquad \text{where} \quad \tilde{\mathbf{z}}^{\text{goal}}_{s'} = f_{\text{g-dyn}}(\mathbf{z}_s, \mathbf{z}_g). \tag{8}$$

Goal-conditioned Action Prediction ($f_{\text{g-act}}$). Given $(\mathbf{z}_s, \mathbf{z}_g)$, this module predicts the single-step dataset action $\tilde{\mathbf{a}}^{\text{goal}}$ taken toward the goal:

$$\mathcal{L}_{\text{g-act}} = \|\tilde{\mathbf{a}}^{\text{goal}} - a\|_2^2, \qquad \text{where} \quad \tilde{\mathbf{a}}^{\text{goal}} = f_{\text{g-act}}(\mathbf{z}_s, \mathbf{z}_g). \tag{9}$$

In contrast to local dynamics models that condition on actions a, $f_{\text{g-dyn}}$ and $f_{\text{g-act}}$ condition explicitly on goal g. This conditions state-goal pairs $(\mathbf{z}_s, \mathbf{z}_g)$ to encode goal-reaching behavior directly, establishing the behavioral alignment useful for effective policy extraction.

Let $\pi^*(s, g) \in \arg\max_a Q^*(s, a, g)$ be an optimal action, assume Q^* is L_Q-Lipschitz in its action argument, write $\hat{a} := f_{\text{g-act}}(\mathbf{z}_s, \mathbf{z}_g)$, and let $\varepsilon_\pi := \mathbb{E}[\|a - \pi^*(s, g)\|_2^2]$ be the distance between dataset and optimal actions (a property of the data).

Proposition 3 (Behavioral alignment bounds the one-step gap of a latent policy)**.** *There exists a deterministic policy on the latent pair, namely $\hat{a}(\mathbf{z}_s, \mathbf{z}_g)$, whose expected one-step suboptimality satisfies*

$$\mathbb{E}\big[\, Q^*(s, \pi^*(s, g), g) - Q^*(s, \hat{a}, g) \,\big] \;\le\; L_Q\big(\sqrt{\delta_{\text{beh}}} + \sqrt{\varepsilon_\pi}\,\big). \tag{10}$$

If δ_{beh} is small, $(\mathbf{z}_s, \mathbf{z}_g)$ jointly encode enough of the state–goal relation for a near-optimal action to be read off them; the actor π_ϕ is trained by Eq. equation 13, not by copying $f_{\text{g-act}}$, and the proposition says the latent has the capacity to support it. In single-task latent-MDP analyses (Fujimoto et al., 2025) the policy depends on $\mathbf{z}_s$ alone and no such condition arises; in the goal-conditioned setting π^* depends jointly on (s, g), and behavioral alignment is what makes that joint dependence representable. $f_{\text{g-dyn}}$ introduces no fourth condition: its achievable error is bounded by $2\delta_{\text{dyn}} + 2L_{\text{dyn}}^2\delta_{\text{beh}}$ (Remark 2), but it supervises directly a quantity the other heads reach only by composition, which matters in practice (Sec-5.2).

The three terms are distinct. None of $\delta_{\text{dyn}}, \Delta_{\text{temp}}, \delta_{\text{beh}}$ controls another: a latent can be transition-consistent yet discard goal distance, return can be predictable from $(\mathbf{z}_{sa}, \mathbf{z}_g)$ while $(\mathbf{z}_s, \mathbf{z}_g)$ does not determine the goal-directed action, and knowing that action does not make its consequence predictable (Appendix A.7).

4.4 AGENT TRAINING AND OPTIMIZATION

We jointly train the multi-scale representation modules and the downstream goal-conditioned policy using offline trajectory data. Optimization alternates between (i) multi-step representation updates on unrolled trajectory chunks and (ii) standard actor-critic updates on random transition batches. Algorithm 1 summarizes the procedure.

Representation Learning Objective. Given an unrolled trajectory chunk $\tau = \{(s_h, a_h, r_h, s_{h+1})\}_{h=0}^{H-1}$ sampled from $\mathcal{D}$ and goal g, we encode initial state and goal latents as $\mathbf{z}_s^0 = \mathcal{E}_\psi^s(s_0)$ and $\mathbf{z}_g = \mathcal{E}_\psi^s(g)$. Unrolling latent dynamics over horizon H, we compute action-conditioned representations $\mathbf{z}_{sa}^h = \mathcal{E}_\psi^{sa}(\mathbf{z}_s^h, a_h)$ at step h. The total representation loss aggregates multi-scale prediction errors over horizon H:

$$\mathcal{L}_{\textit{MSPR}} = \sum_{h=0}^{H-1} \left[\lambda_{\text{dyn}}\mathcal{L}_{\text{dyn}} + \lambda_{\text{rew}}\mathcal{L}_{\text{rew}} + \lambda_{\text{g-dyn}}\mathcal{L}_{\text{g-dyn}} + \lambda_{\text{g-act}}\mathcal{L}_{\text{g-act}}\right]. \tag{11}$$

Goal-Conditioned Actor-Critic. Value estimation updates two critics by minimizing Huber loss against a shared n-step TD target along dataset trajectory segments (Fujimoto et al., 2018; 2025):

$$\mathcal{L}_Q = \sum_{i=1,2} \mathrm{Huber}_\delta\left(Q_{\theta_i}(\mathbf{z}_{sa}, \mathbf{z}_g) - \left(R_t^{(n)} + \gamma^n \min_{j=1,2} Q_{\theta'_j}(\mathbf{z}^n_{sa}, \mathbf{z}_g)\right)\right), \tag{12}$$

where $R_t^{(n)}$ is the n-step discounted return, $\mathbf{z}^n_{sa}$ is the state-action representation n steps ahead with the target policy's action, and $Q_{\theta'_j}$ are the target critics. The actor policy $\pi_\phi(a|\mathbf{z}_s, \mathbf{z}_g)$ is trained to maximize predicted value while constrained by behavior cloning regularization:

$$\mathcal{L}_\pi = -0.5 \sum_{i=1,2} Q_{\theta_i}(\mathbf{z}_{sa}, \mathbf{z}_g) + \lambda_{\mathrm{BC}} \|\pi_\phi(\mathbf{z}_s, \mathbf{z}_g) - a\|_2^2. \tag{13}$$

Crucially, representation weights ψ receive zero gradients from critic loss $\mathcal{L}_Q$ or actor loss $\mathcal{L}_\pi$. This actor-critic is the goal-conditioned TD3+BC agent (Fujimoto et al., 2018; Fujimoto & Gu, 2021); without the shared predictive encoder, we refer to it as GCTD3 and use it as the backbone baseline in Sec-5.2.

Algorithm 1 Representation and Policy Learning

Input: Offline dataset $\mathcal{D}$, horizon H, target update frequency K
Initialize: Encoders ψ, predictors ω, actor ϕ, critic θ; targets $\bar{\psi}, \bar{\phi}, \bar{\theta}$
for $t = 1 \dots T$ **do**
  if $t \bmod K = 0$ **then**
    Update targets: $\bar{\psi} \leftarrow \psi,\ \bar{\phi} \leftarrow \phi,\ \bar{\theta} \leftarrow \theta$
    for $k = 1 \dots K$ **do**
      Sample trajectory chunk $\tau \sim \mathcal{D}$
      Update representation (ψ, ω) by minimizing $\mathcal{L}_{MSPR}(\tau)$ (Eq. 11)
    end for
  end if
  Sample batch $(s, a, r, s', g) \sim \mathcal{D}$
  Update critic θ using TD loss $\mathcal{L}_Q$ (Eq. 12)
  Update actor ϕ using $\mathcal{L}_\pi$ (Eq. 13)
end for

5 Experiments

Our experimental evaluation assesses the representations learned by *MSPR* across state-based and pixel-based offline goal-conditioned reinforcement learning (GCRL) tasks from OGBench (Park et al., 2024a). After reporting the main benchmark results, we structure our analysis around four core questions:

(RQ1) Ablation: Which alignment condition matters most, and where?

(RQ2) Backbone: How much of the gain comes from the representation?

(RQ3) Data efficiency: Does the predictive representation help when data is limited?

(RQ4) Failure modes: Where does *MSPR* fall short, and why?

Tasks and Environments. For main benchmark, we evaluate across 12 state-based and 5 pixel-based environments from OGBench (Park et al., 2024a), covering two main domains: *locomotion* (AntMaze, PointMaze, HumanoidMaze) and *manipulation* (Cube, Scene, Puzzle), using the `navigate` and `play` dataset splits. Additional results on the `stitch` datasets are reported in Appendix E.3.

Baselines. We compare against standard GCRL baselines: **GCBC** (Lynch et al., 2019), **GCIVL** (Park et al., 2024b), **GCIQL** (Kostrikov et al., 2021), **QRL** (Wang et al., 2023), **CRL** (Eysenbach et al., 2023), and the hierarchical **HIQL** (Park et al., 2024c). As the primary goal-representation benchmark, we evaluate **Dual Goal Representations** (Park et al., 2025b) on the two best backbones reported, GCIVL and CRL ($\text{Dual}_{\text{GCIVL}}$, Dual_{CRL}). Additionally, to isolate the effect of representation learning, we compare against **GCTD3** (Fujimoto & Gu, 2021), the TD3+BC agent that *MSPR* is built on. GCTD3 follows the OGBench network design (Park et al., 2024a): the actor and critic each learn their own features from their own losses. *MSPR* differs only in the encoder, which is shared between the actor and critic and trained using the predictive losses.

Environment	GCBC	GCIVL	GCIQL	QRL	CRL	$\text{Dual}_{\text{GCIVL}}$	Dual_{CRL}	HIQL	OURS
State-based environments									
pointmaze-medium-navigate	9 ±6	79 ±1	66 ±8	**82** ±5	29 ±7	**86** ±2	33 ±10	79 ±3	54 ±6
pointmaze-large-navigate	29 ±6	65 ±8	38 ±2	**86** ±9	39 ±7	67 ±4	39 ±12	73 ±5	48 ±8
antmaze-medium-navigate	29 ±4	87 ±1	90 ±3	88 ±3	**95** ±1	82 ±4	93 ±3	**96** ±2	**99** ±1
antmaze-large-navigate	24 ±2	33 ±7	40 ±5	75 ±6	83 ±4	41 ±1	87 ±2	92 ±2	**98** ±2
antmaze-giant-navigate	0 ±0	1 ±1	1 ±1	14 ±3	16 ±3	1 ±2	21 ±4	**68** ±6	**68** ±7
humanoidmaze-medium-navigate	8 ±2	31 ±4	34 ±3	21 ±8	60 ±4	32 ±4	57 ±4	**92** ±2	67 ±5
humanoidmaze-large-navigate	1 ±0	3 ±1	7 ±3	5 ±1	24 ±4	5 ±2	18 ±4	**58** ±3	**58** ±3
cube-single-play	6 ±2	53 ±6	60 ±3	5 ±1	19 ±2	**80** ±3	60 ±1	16 ±4	65 ±5
cube-double-play	1 ±1	36 ±3	42 ±2	1 ±0	10 ±2	**57** ±6	24 ±5	6 ±2	13 ±3
scene-play	5 ±1	55 ±5	35 ±5	5 ±1	19 ±2	**91** ±4	44 ±5	43 ±9	45 ±5
puzzle-3x3-play	2 ±0	3 ±2	90 ±8	1 ±0	3 ±1	5 ±1	6 ±1	13 ±3	**98** ±1
puzzle-4x4-play	0 ±0	8 ±4	33 ±5	0 ±0	0 ±0	14 ±3	2 ±0	7 ±2	**93** ±5
Avg. State	10 ±1	38 ±1	45 ±1	32 ±1	33 ±1	47 ±1	40 ±2	54 ±1	**67** ±1
Pixel-based environments									
v-antmaze-medium-navigate	11 ±2	48 ±1	36 ±6	0 ±0	**95** ±1	85 ±2	85 ±3	**94** ±2	90 ±3
v-antmaze-large-navigate	4 ±0	25 ±8	7 ±3	0 ±0	**84** ±1	57 ±3	43 ±3	**81** ±1	**81** ±4
v-cube-single-play	5 ±1	65 ±8	50 ±9	41 ±15	31 ±15	64 ±6	14 ±8	**81** ±4	**82** ±5
v-scene-play	12 ±2	33 ±2	15 ±1	10 ±1	11 ±2	38 ±7	15 ±1	46 ±2	**56** ±5
v-puzzle-3x3-play	0 ±0	38 ±4	10 ±2	1 ±1	0 ±0	0 ±0	0 ±0	**75** ±11	30 ±4
Avg. Pixel	6 ±1	42 ±2	24 ±2	10 ±3	44 ±3	49 ±2	31 ±2	**75** ±2	68 ±2

Table 1: **Main Results** across state-based and pixel-based settings. Bold: within 95% of the row best. GCBC, QRL, CRL, and Dual_{CRL}: published numbers; all others re-run with $n = 3$.

Target Horizon (n-step). Since n-step TD targets significantly influence offline GCRL performance, we re-run all TD-based baselines (GCIQL, GCIVL, $\text{Dual}_{\text{GCIVL}}$, and HIQL) using $n = 3$; results at $n = 1$ are reported in Appendix E.2. n-step targets are undefined for GCBC, QRL, CRL, and Dual_{CRL}, which are reported using published numbers.

Evaluation. State-based results are averaged over 8 random seeds and pixel-based results over 4, reporting the mean success rate across the final three checkpoints. For pixel-based OGBench (Park et al., 2024a), prior work reported that frame stacking (FS) does not improve performance; we therefore use single frames for baselines. In contrast, *MSPR* uses three stacked frames. We isolate the specific effect of FS on both *MSPR* and GCTD3 in Sec-5.2.

5.1 Main Results

Table 1 shows that *MSPR* leads on average across state-based tasks (67 versus 54 for HIQL), while HIQL leads on pixel-based tasks (75 versus 68). Much of the gap on pixel-based tasks comes from `visual-puzzle-3x3`, where *MSPR* scores 30 compared with HIQL's 75. While other baselines are stronger in a particular category, either manipulation or locomotion, our method performs well on environments from both categories. *MSPR* is particularly strong on `antmaze` and on the state-based puzzle tasks, where it achieves the best results. On pixel-based tasks, it is the best or within 5 points of the best on four of the five environments, and falls clearly behind only on `visual-puzzle-3x3`.

The performance on `PointMaze` is more surprising. Despite appearing simpler than the other locomotion tasks, these environments remain difficult for our method. The drop from `cube-single` (65) to `cube-double` (13) is also striking. This weakness might partly come from the TD3's Actor-Critc backbone (Appendix Table 3), and the ablations (Sec-5.2) show that no single component is specifically hurting the performance. One possible explanation is that handling two cubes requires more complex decision making that the representation does not reliably preserve. The additional object may also make the goal-conditioned representation harder to learn.

5.2 Ablations and Analysis

To understand how each alignment objective contributes to the overall performance, we perform a leave-one-out ablation study. Additionally, we also study the contribution of n-step prediction and frame stacking (Pixel envs only) with respect to the full version of MSPR. These ablations help us understand which components are important and identify the environments where our method still falls behind other approaches.

Condition sensitivity (RQ1). Given that these are goal-conditioned tasks, removing the temporal alignment objective $\mathcal{L}_{\text{rew}}$ has a clear impact on performance. Manipulation environments are less affected, while the impact is much larger on locomotion, especially for long-horizon tasks such as `antmaze-giant`, `humanoidmaze-medium`, and `humanoidmaze-large`, where

Environment	w/o $\mathcal{L}_{\text{dyn}}$	w/o $\mathcal{L}_{\text{rew}}$	w/o $\mathcal{L}_{\text{g-dyn}}$	w/o $\mathcal{L}_{\text{g-act}}$	$n=1$	w/o FS	OURS
Pixel-based environments							
v-antmaze-medium-navigate	89 ±4	89 ±3	87 ±2	90 ±3	92 ±3	86 ±4	90 ±3
v-antmaze-large-navigate	72 ±7	28 ±10	69 ±5	86 ±3	77 ±5	78 ±5	81 ±4
v-cube-single-play	83 ±5	71 ±3	87 ±1	79 ±9	82 ±2	63 ±1	82 ±5
v-scene-play	57 ±7	65 ±2	32 ±2	49 ±6	36 ±6	25 ±3	56 ±5
v-puzzle-3x3-play	21 ±1	39 ±2	0 ±0	25 ±2	20 ±1	21 ±1	30 ±4
Average	64 ±2	58 ±2	55 ±1	66 ±2	61 ±2	55 ±1	**68** ±2
State-based environments							
pointmaze-medium-navigate	43 ±4	36 ±7	60 ±3	47 ±6	50 ±1	–	54 ±6
pointmaze-large-navigate	24 ±9	31 ±3	62 ±3	33 ±3	30 ±2	–	48 ±8
antmaze-medium-navigate	96 ±2	98 ±1	99 ±0	98 ±0	98 ±1	–	**99** ±1
antmaze-large-navigate	97 ±1	92 ±3	98 ±1	97 ±1	96 ±1	–	**98** ±2
antmaze-giant-navigate	54 ±1	0 ±0	57 ±1	55 ±3	43 ±4	–	**68** ±7
humanoidmaze-medium-navigate	56 ±3	0 ±0	0 ±0	75 ±9	35 ±3	–	67 ±5
humanoidmaze-large-navigate	53 ±2	0 ±0	0 ±0	43 ±5	11 ±3	–	**58** ±3
cube-single-play	50 ±3	63 ±2	68 ±6	62 ±4	46 ±4	–	65 ±5
cube-double-play	14 ±2	12 ±4	12 ±1	11 ±2	11 ±2	–	13 ±3
scene-play	45 ±2	47 ±1	44 ±4	38 ±1	30 ±4	–	45 ±5
puzzle-3x3-play	93 ±1	99 ±0	99 ±0	100 ±1	81 ±4	–	98 ±1
puzzle-4x4-play	81 ±3	84 ±5	95 ±0	94 ±1	73 ±2	–	93 ±5
Average	59 ±1	47 ±1	58 ±1	63 ±1	50 ±1	–	**67** ±1

Table 2: **Ablation study.** Each cell shows the variant's score. Red ( ) and green ( ) indicate performance w.r.t *MSPR*, respectively, by ≥ 15, $[5, 15)$, and $[0, 5)$ success rate.

performance drops to zero. Two exceptions are `visual-puzzle-3x3` and `visual-scene`, which perform better without temporal alignment. Removing $\mathcal{L}_{\text{g-dyn}}$ reduces the average performance from 67 to 58 and causes a similar collapse on the two humanoid maze tasks as removing temporal alignment.

On `visual-puzzle-3x3`, performance also drops to zero, suggesting that behavioral alignment, particularly the goal-conditioned dynamics objective, helps learn representations that are important for this task. In comparison, $\mathcal{L}_{\text{g-act}}$ (Proposition 3) has the smallest effect on average, although removing it still leads to a 15-point drop on `humanoidmaze-large`. Finally, dropping $\mathcal{L}_{\text{dyn}}$ (Proposition 1) costs 8 points on state and 4 on pixel, and the drop is spread across environments rather than concentrated. Frame stacking, in particular, provides a clear improvement on visual manipulation tasks, while the improvements on locomotion tasks are relatively smaller. Finally, as we scale *MSPR* to longer-horizon tasks, n-step returns become important, with a larger effect on state-based environments. Removing any single objective does not outperform the full *MSPR*, but the ablations show how each objective contributes to the overall performance.

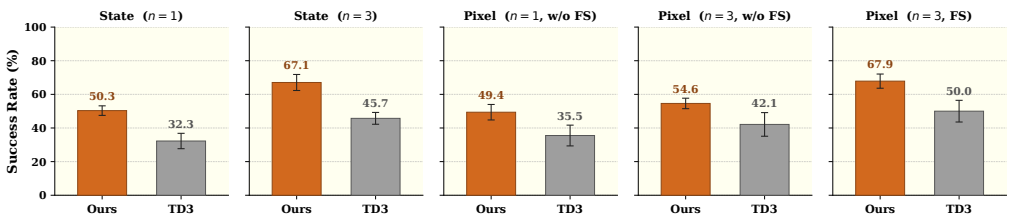


Figure 3: **Representation vs. backbone.** Average success of *MSPR* (orange) and its GCTD3 backbone (grey). Both use the same TD3+BC agent and n-step targets. Per-environment numbers in Appendix Tables 3 and 4.

Representation vs. backbone (RQ2). Following the same architecture as the OGBench baselines, GCTD3 shares the actor-critic algorithm and n-step targets with *MSPR*. The only difference is the representation learning used in *MSPR*, allowing us to isolate the gains from the predictive representation. Each subplot in Figure 3 compares the two agents under the same setting, and in every case *MSPR* performs better, showing the positive impact of learning representations through the alignment conditions. Across subplots, both agents improve by a similar amount (17 and 14 points on state), suggesting that the gain from n-step targets mainly comes from the TD3+BC backbone. This also suggests that the comparison is fair, as at $n=3$ the backbone reaches 46 on state, close to the other baselines. This is consistent with Proposition 2: The predictive losses, in particular return prediction, organize the latent space by proximity to the goal, helping the critic learn a value function that reflects this proximity. The Q-value traces in Appendix F illustrate this, with the critic's value increasing as the agent approaches the goal (Figure 6). Per-environment results are provided in Appendix Tables 3 and 4.

Frame stacking on pixel-based tasks. OGBench reports that frame stacking does not help the baselines on pixel-based tasks (Park et al., 2024a), so we follow that setup and run the baselines with single frames. For our method, we isolate its effect by running both *MSPR* and its GCTD3 backbone with and without frame stacking. Frame stacking helps both, but it helps *MSPR* more, and the gain is largest on manipulation tasks, where the agent must infer object motion from consecutive frames. Without frame stacking, *MSPR* still clearly improves on GCTD3, isolating the gain from predictive representation.

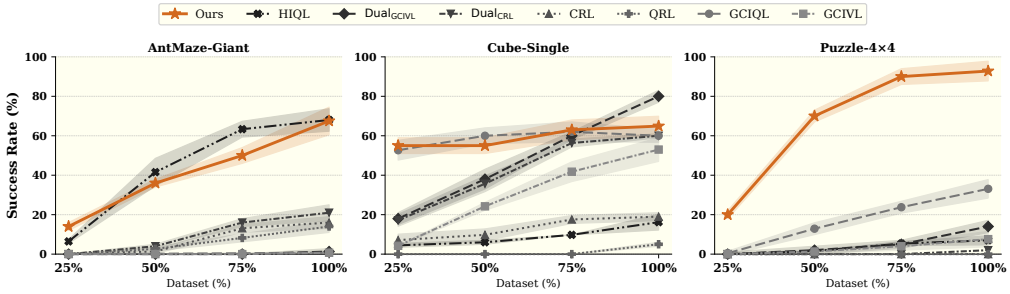


Figure 4: **Data efficiency** under reduced training data (25%, 50%, 75%, 100%); shaded bands are one std over 4 seeds.

Data Efficiency (RQ3). To test how much the predictive representation helps when data is limited, we train each method on 25%, 50%, 75%, and 100% of the original offline datasets (Figure 4). We choose three environments that are challenging for all methods and cover different types of difficulty: `antmaze-giant` for long-horizon navigation, `cube-single` and `puzzle-4x4` for manipulation tasks. Across all three, the performance of *MSPR* suggests that the predictive representation helps it learn useful features from fewer samples. On `puzzle-4x4`, *MSPR* keeps most of its performance down to half of the data, while every baseline stays low at every fraction. Among the baselines, HIQL stands out on `antmaze-giant`, and GCIQL performs similarly to *MSPR* on `cube-single`.

Failure Analysis (RQ4). We observe two situations where *MSPR* is less effective. The first is visual bias in pixel-based navigation: in `visual-antmaze`, the agent sometimes reaches a state that looks almost identical to the goal but lies in a different part of the maze, and the critic assigns it a high value (Appendix F, Figure 8); in `puzzle-3x3`, the policy can also settle into a repeated action loop (Figure 9). The second arises in state-based tasks such as `cube-double` and `scene`. There, methods that restrict learning to dataset actions, such as GCIQL, GCIVL, and Dual$_{\text{GCIVL}}$, are better suited, since they learn Q-values only from actions in the dataset, which limits how far errors can spread (Kostrikov et al., 2021; Kumar et al., 2020). The GCTD3 backbone shows the same failure pattern as *MSPR* (Table 3), suggesting that some of the limitations of the backbone carry over to *MSPR* and are not fully addressed by the learned representation.

6 Conclusion

From a representation learning perspective, offline GCRL demands representations that captures the environment at more than one scale, not just effective policies or well tuned value functions. *MSPR* learns such a representation by predicting the information that matters for reaching the goal: how the state dynamics changes, how the return changes toward the goal, and which actions lead to it. Across state-based and pixel-based tasks, this representation gives a clear improvement over its own backbone and over prior goal-representation methods. The alignment conditions improve performance, specifically temporal alignment matters most for long-horizon navigation, behavioral alignment matters most for pixel-based tasks, and dynamical alignment gives a smaller gain spread across most tasks. This supports learning the representation from several predictive signals at once.

Limitations. On some long-horizon tasks the representation alone is not enough and explicit planning helps, and on tasks such as `cube-double` and `scene`, methods that stay close to the dataset perform better. Since *MSPR* does not depend on the policy that uses its representation, combining it with hierarchical planning is a natural next step.

AI USE STATEMENT

For this work, Generative AI tools were used to improve the writing, verifying the implementation, cross checking the math proofs. Generative AI tools were not used for literature retrieval and discovery, research ideation or execution, synthetic dataset generation, or proving mathematical claims.

ETHICS STATEMENT

There is no ethics concern for this work *MSPR*.

REPRODUCIBILITY STATEMENT

For full transparency and reproducibility, our implementation relies on publicly available codebases and benchmark suites. Detailed descriptions of key hyperparameters and Code Listings are provided in the Appendix to allow straightforward replication of our findings.

REFERENCES

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck, 2019. URL https://arxiv.org/abs/1612.00410.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL https://arxiv.org/abs/2404.08471.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes, 2019. URL https://arxiv.org/abs/1911.09291.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir 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, 2018. URL https://arxiv.org/abs/1802.01561.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning, 2023. URL https://arxiv.org/abs/2206.07568.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning, 2025. URL https://arxiv.org/abs/2501.16142.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning, 2019. URL https://arxiv.org/abs/1906.02736.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019. URL https://arxiv.org/abs/1811.04551.

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

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.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, 1993. URL https://api.semanticscholar.org/CorpusID:5538688.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pp. 267–274, 2002.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Daniel Lawson, Adriana Hugessen, Charlotte Cloutier, Glen Berseth, and Khimya Khetarpal. Self-predictive representations for combinatorial generalization in behavioral cloning, 2025. URL https://arxiv.org/abs/2506.10137.

Xing Lei, Xuetao Zhang, Zifeng Zhuang, and Donglin Wang. Q-wsl: Optimizing goal-conditioned rl with weighted supervised learning via dynamic programming, 2025. URL https://arxiv.org/abs/2410.06648.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. In *Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 1555–1561, Cambridge, MA, USA, 2001. MIT Press.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions, 2022. URL https://arxiv.org/abs/2201.08299.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play, 2019. URL https://arxiv.org/abs/1903.01973.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training, 2023. URL https://arxiv.org/abs/2210.00030.

Vivek Myers, Bill Chunyuan Zheng, Anca Dragan, Kuan Fang, and Sergey Levine. Temporal representation alignment: Successor features enable emergent compositionality in robot instruction following, 2025. URL https://arxiv.org/abs/2502.05454.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024a.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl?, 2024b. URL https://arxiv.org/abs/2406.09329.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions, 2024c. URL https://arxiv.org/abs/2307.11949.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations, 2024d. URL https://arxiv.org/abs/2402.15567.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable, 2025a. URL https://arxiv.org/abs/2506.04168.

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

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 752–759, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390251. URL https://doi.org/10.1145/1390156.1390251.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods, 2018. URL https://arxiv.org/abs/1802.10264.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations, 2021. URL https://arxiv.org/abs/2007.05929.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020. URL https://arxiv.org/abs/2004.04136.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3 (1):9–44, August 1988. doi: 10.1007/BF00115009.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning, 2023. URL https://arxiv.org/abs/2304.01203.

Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl, 2022. URL https://arxiv.org/abs/2202.04478.

A Proofs and Extended Analysis

Notation is that of Sections 3–4. All expectations are over $(s, a, s', g) \sim \mathcal{D}$ and, wherever r_{MC} appears, over the remainder of the dataset trajectory that determines it. The latents $\mathbf{z}_s = \mathcal{E}^s_\psi(s)$, $\mathbf{z}_g = \mathcal{E}^s_\psi(g)$, and $\mathbf{z}_{sa} = \mathcal{E}^{sa}_\psi(\mathbf{z}_s, a)$ are deterministic functions of (s, a, g), and $\mathbf{z}_{s'} = \mathcal{E}^s_\psi(s')$ is a deterministic function of s'. We write $\|\cdot\|$ for the Euclidean norm and $\|Y\|_{L^2} := (\mathbb{E}[Y^2])^{1/2}$ for a scalar random variable Y. Rewards are bounded and $\gamma < 1$, so r_{MC} and all value functions are square-integrable.

Two standard facts. We use these repeatedly; both are textbook properties of conditional expectation.

- **F1** *Conditional mean is the L^2 projection.* For a square-integrable random variable Y and a random vector W, $\min_f \mathbb{E}[(Y - f(W))^2]$ over measurable f is attained at $f(W) = \mathbb{E}[Y \mid W]$.
- **F2** *Tower property.* If W is a function of X, then $\mathbb{E}\big[\mathbb{E}[Y \mid X] \,\big|\, W\big] = \mathbb{E}[Y \mid W]$.

A.1 Proof of Proposition 1

Proof. Fix a sample (s, a, s', g). Since V is L_V-Lipschitz in its first argument,

$$\big|V(f_{\mathrm{dyn}}(\mathbf{z}_{sa}), \mathbf{z}_g) - V(\mathbf{z}_{s'}, \mathbf{z}_g)\big| \;\le\; L_V \,\big\|f_{\mathrm{dyn}}(\mathbf{z}_{sa}) - \mathbf{z}_{s'}\big\|.$$

Squaring both sides and taking expectations,

$$\mathbb{E}\Big[\big(V(f_{\mathrm{dyn}}(\mathbf{z}_{sa}), \mathbf{z}_g) - V(\mathbf{z}_{s'}, \mathbf{z}_g)\big)^2\Big] \;\le\; L_V^2\, \mathbb{E}\big[\|f_{\mathrm{dyn}}(\mathbf{z}_{sa}) - \mathbf{z}_{s'}\|^2\big] \;=\; L_V^2\, \delta_{\mathrm{dyn}},$$

where the last equality is the definition equation 1. □

A.2 Proof of Proposition 2

Throughout this proof write

$$X := (s, a, g), \qquad W := (\mathbf{z}_{sa}, \mathbf{z}_g), \qquad \mu(W) := \mathbb{E}[r_{\mathrm{MC}} \mid W].$$

W is a function of X, and by definition $Q^\beta(X) = \mathbb{E}[r_{\mathrm{MC}} \mid X]$.

Lemma 1. $\mu(W) = \mathbb{E}[Q^\beta \mid W]$.

Proof. By (F2) applied with $Y = r_{\mathrm{MC}}$, $\mathbb{E}[r_{\mathrm{MC}} \mid W] = \mathbb{E}\big[\mathbb{E}[r_{\mathrm{MC}} \mid X] \,\big|\, W\big] = \mathbb{E}[Q^\beta \mid W]$. □

Proof of Proposition 2. We establish the identity $\Delta_{\mathrm{temp}} = \sigma^2_{\mathrm{MC}} + \varepsilon^{\min}_V$ in four steps.

Step 1: $\Delta_{\mathrm{temp}} = \mathbb{E}[(r_{\mathrm{MC}} - \mu(W))^2]$. Definition equation 2 minimizes $\mathbb{E}[(r_{\mathrm{MC}} - f(W))^2]$ over measurable f. By (F1) with $Y = r_{\mathrm{MC}}$, the minimizer is $f(W) = \mathbb{E}[r_{\mathrm{MC}} \mid W] = \mu(W)$.

Step 2: $\varepsilon^{\min}_V = \mathbb{E}[(Q^\beta - \mu(W))^2]$. The definition of $\varepsilon^{\min}_V$ (Sec-4.2) minimizes $\mathbb{E}[(Q^\beta - Q(W))^2]$ over measurable Q. By (F1) with $Y = Q^\beta$, the minimizer is $Q(W) = \mathbb{E}[Q^\beta \mid W]$, which equals $\mu(W)$ by Lemma 1.

Step 3: the cross term vanishes, $\mathbb{E}\big[(r_{\mathrm{MC}} - Q^\beta)(Q^\beta - \mu(W))\big] = 0$. The factor $Q^\beta - \mu(W)$ is a function of X (both Q^β and W are). Conditioning on X and using (F2),

$$\mathbb{E}\big[(r_{\mathrm{MC}} - Q^\beta)(Q^\beta - \mu(W))\big] = \mathbb{E}\Big[(Q^\beta - \mu(W)) \cdot \underbrace{\mathbb{E}[r_{\mathrm{MC}} - Q^\beta \mid X]}_{= Q^\beta - Q^\beta = 0}\Big] = 0.$$

Step 4: assemble. Write $r_{\mathrm{MC}} - \mu(W) = (r_{\mathrm{MC}} - Q^\beta) + (Q^\beta - \mu(W))$, square, and take expectations. The cross term is zero by Step 3, so

$$\underbrace{\mathbb{E}[(r_{\mathrm{MC}} - \mu(W))^2]}_{\Delta_{\mathrm{temp}}\ \text{(Step 1)}} \;=\; \underbrace{\mathbb{E}[(r_{\mathrm{MC}} - Q^\beta)^2]}_{\sigma^2_{\mathrm{MC}}} \;+\; \underbrace{\mathbb{E}[(Q^\beta - \mu(W))^2]}_{\varepsilon^{\min}_V\ \text{(Step 2)}}.$$

This is equation 7. □

Why the trained loss is an upper bound. A specific trained head f_{rew} is one particular measurable function of W, so $\mathbb{E}[\mathcal{L}_{\mathrm{rew}}] = \mathbb{E}[(r_{\mathrm{MC}} - f_{\mathrm{rew}}(W))^2] \geq \Delta_{\mathrm{temp}}$, with equality only if $f_{\mathrm{rew}} = \mu$ almost surely. This is the inequality in equation 2.

A.3 Representation error versus data error

Corollary 1 (Representation error versus data error)**.** *Let* $\varepsilon_\beta := \big(\mathbb{E}[(Q^\beta - Q^*)^2]\big)^{1/2}$. *Then*

$$\min_Q \Big(\mathbb{E}\big[\big(Q(\mathbf{z}_{sa}, \mathbf{z}_g) - Q^*(s,a,g)\big)^2\big]\Big)^{1/2} \;\leq\; \sqrt{\varepsilon_V^{\min}} \;+\; \varepsilon_\beta. \tag{14}$$

The error against the *optimal* value splits into a representation term, controlled by the encoder through Proposition 2, and a data term ε_β, the behavior policy's suboptimality, which no encoder can reduce. As the behavior data degrades, for instance on stitching datasets where individual behavior trajectories do not reach the goal, ε_β grows while the representation term does not, which is the mechanism behind the graceful, rather than abrupt, degradation reported in Sec-5.2.

Proof. For any critic Q on W, Minkowski's inequality in L^2 gives

$$\|Q(W) - Q^*\|_{L^2} \;\leq\; \|Q(W) - Q^\beta\|_{L^2} + \|Q^\beta - Q^*\|_{L^2} \;=\; \|Q(W) - Q^\beta\|_{L^2} + \varepsilon_\beta.$$

The second term does not depend on Q. Taking the minimum over Q on both sides and using $\min_Q \|Q(W) - Q^\beta\|_{L^2} = \sqrt{\varepsilon_V^{\min}}$ (the square root of the definition of $\varepsilon_V^{\min}$ in Sec-4.2) yields equation 14. □

A.4 When the condition binds

Remark 1 (When the condition binds). If $\mathcal{E}_\psi^s$ and $\mathcal{E}_\psi^{sa}$ are injective on the data support, as when raw states are used with no compression, then $(\mathbf{z}_{sa}, \mathbf{z}_g)$ determines (s, a, g) and $\varepsilon_V^{\min} = 0$ with no auxiliary objective at all. Propositions 1–3 are therefore statements about *learned, compressive* encoders: the regime of pixel observations and high-dimensional state.

Proof. If $\mathcal{E}_\psi^s$ and $\mathcal{E}_\psi^{sa}$ are injective on the data support, the map $X \mapsto W$ is injective there, so W determines X and the two generate the same information: $\mathbb{E}[Q^\beta \mid W] = Q^\beta$. By Step 2 of the proof of Proposition 2, $\varepsilon_V^{\min} = \mathbb{E}[(Q^\beta - \mathbb{E}[Q^\beta \mid W])^2] = 0$. □

A.5 Proof of Proposition 3

Proof. Fix (s, a, g) and write $a^* := \pi^*(s, g)$ and $\hat{a} := f_{\text{g-act}}(\mathbf{z}_s, \mathbf{z}_g)$. Because a^* maximizes $Q^*(s, \cdot, g)$, the gap $Q^*(s, a^*, g) - Q^*(s, \hat{a}, g)$ is nonnegative, and because Q^* is L_Q-Lipschitz in its action argument,

$$0 \;\leq\; Q^*(s, a^*, g) - Q^*(s, \hat{a}, g) \;\leq\; L_Q \, \|a^* - \hat{a}\|.$$

Insert the dataset action a and apply the triangle inequality:

$$\|a^* - \hat{a}\| \;\leq\; \|a^* - a\| + \|a - \hat{a}\|.$$

Take expectations over $(s, a, g) \sim \mathcal{D}$. For each term, Jensen's inequality gives $\mathbb{E}\|Y\| \leq (\mathbb{E}\|Y\|^2)^{1/2}$, hence

$$\mathbb{E}\big[Q^*(s, a^*, g) - Q^*(s, \hat{a}, g)\big] \;\leq\; L_Q\Big(\mathbb{E}\|a^* - a\| + \mathbb{E}\|a - \hat{a}\|\Big) \;\leq\; L_Q\Big(\sqrt{\varepsilon_\pi} + \sqrt{\delta_{\mathrm{beh}}}\Big),$$

where $\mathbb{E}\|a - \hat{a}\|^2 = \delta_{\mathrm{beh}}$ by equation 3 and $\mathbb{E}\|a^* - a\|^2 = \varepsilon_\pi$ by definition. This is equation 10. □

Multi-step form (standard; stated for completeness, not used in the main text). Let $\epsilon := \mathbb{E}[Q^*(s, \pi^*(s, g), g) - Q^*(s, \hat{a}, g)]$ denote the one-step gap bounded above. If the same gap holds under the state distribution induced by the latent policy $\hat{a}$, the performance-difference lemma (Kakade & Langford, 2002) gives a return gap of at most $\epsilon/(1-\gamma)$. When the gap is only controlled under the dataset distribution $\mathcal{D}$, as here, the additional distribution shift between $\mathcal{D}$ and the latent policy's occupancy contributes a further factor of $1/(1-\gamma)$, giving $O(\epsilon/(1-\gamma)^2)$ (Ross et al., 2011). Both factors are properties of the compounding-error argument, not of the representation; Proposition 3 controls only ϵ.

A.6 GOAL-LEVEL DYNAMICS INTRODUCES NO FOURTH CONDITION

Remark 2 (Goal-level dynamics). $f_{\text{g-dyn}}$ supervises the same latent pair as $f_{\text{g-act}}$ with a complementary target, the successor latent. Its achievable error is controlled by the other two conditions: if $a \mapsto f_{\text{dyn}}(\mathcal{E}_\psi^{sa}(\mathbf{z}_s, a))$ is L_{dyn}-Lipschitz, then

$$\min_f \mathbb{E}\|f(\mathbf{z}_s, \mathbf{z}_g) - \mathbf{z}_{s'}\|_2^2 \;\le\; 2\,\delta_{\text{dyn}} + 2\,L_{\text{dyn}}^2\,\delta_{\text{beh}}. \tag{15}$$

$f_{\text{g-dyn}}$ therefore introduces no fourth condition; it supervises directly a quantity the other heads reach only by composition. Direct supervision matters in practice: removing it is the largest single loss on pixel-based tasks (Table 2, Sec-5.2).

Proof. Define $h_s(a) := f_{\text{dyn}}\big(\mathcal{E}_\psi^{sa}(\mathbf{z}_s, a)\big)$, the forward-dynamics prediction as a function of the action for a fixed $\mathbf{z}_s$; by assumption h_s is L_{dyn}-Lipschitz. Note that for the dataset action, $h_s(a) = f_{\text{dyn}}(\mathbf{z}_{sa})$. Consider the candidate goal-level dynamics head

$$\bar{f}(\mathbf{z}_s, \mathbf{z}_g) \;:=\; h_s\big(\hat{a}\big), \qquad \hat{a} = f_{\text{g-act}}(\mathbf{z}_s, \mathbf{z}_g),$$

which depends on $(\mathbf{z}_s, \mathbf{z}_g)$ only. By the triangle inequality and the Lipschitz property,

$$\|\bar{f}(\mathbf{z}_s, \mathbf{z}_g) - \mathbf{z}_{s'}\| \;\le\; \|h_s(\hat{a}) - h_s(a)\| + \|h_s(a) - \mathbf{z}_{s'}\| \;\le\; L_{\text{dyn}}\,\|\hat{a} - a\| + \|f_{\text{dyn}}(\mathbf{z}_{sa}) - \mathbf{z}_{s'}\|.$$

Using $(x+y)^2 \le 2x^2 + 2y^2$ and taking expectations,

$$\mathbb{E}\|\bar{f}(\mathbf{z}_s, \mathbf{z}_g) - \mathbf{z}_{s'}\|^2 \;\le\; 2\,L_{\text{dyn}}^2\,\mathbb{E}\|\hat{a} - a\|^2 + 2\,\mathbb{E}\|f_{\text{dyn}}(\mathbf{z}_{sa}) - \mathbf{z}_{s'}\|^2 \;=\; 2\,L_{\text{dyn}}^2\,\delta_{\text{beh}} + 2\,\delta_{\text{dyn}}.$$

The minimum over all heads $f(\mathbf{z}_s, \mathbf{z}_g)$ is at most the error of this particular $\bar{f}$, which gives the stated bound. □

A.7 THE THREE TERMS ARE DISTINCT

None of δ_{dyn}, Δ_{temp}, δ_{beh} controls another. A latent can be transition-consistent yet discard goal distance encoding local pose but not maze position giving small δ_{dyn} and large Δ_{temp}. Return can be predictable from $(\mathbf{z}_{sa}, \mathbf{z}_g)$, which sees the action, while $(\mathbf{z}_s, \mathbf{z}_g)$ does not determine which action is goal-directed small Δ_{temp}, large δ_{beh}. And knowing the goal-directed action does not make its consequence predictable small δ_{beh}, large δ_{dyn}. Removing any one head leaves negatively impacts the overall performance Table 2.

B Environment Details

Our main benchmark uses 17 environments from the OGBench (Park et al., 2024a) suite (12 state-based and 5 pixel-based), and we report 5 additional `stitch` datasets (3 state-based and 2 pixel-based) in Appendix E.3.

State-Based Environments (12 tasks):

- **Locomotion (7 tasks):**
  - `pointmaze-medium-navigate`
  - `pointmaze-large-navigate`
  - `antmaze-medium-navigate`
  - `antmaze-large-navigate`
  - `antmaze-giant-navigate`
  - `humanoidmaze-medium-navigate`
  - `humanoidmaze-large-navigate`
- **Manipulation (5 tasks):**
  - `cube-single-play`
  - `cube-double-play`
  - `scene-play`
  - `puzzle-3x3-play`
  - `puzzle-4x4-play`

Pixel-Based Environments (5 tasks):

- **Locomotion (2 tasks):**
  - `v-antmaze-medium-navigate`
  - `v-antmaze-large-navigate`
- **Manipulation (3 tasks):**
  - `v-cube-single-play`
  - `v-scene-play`
  - `v-puzzle-3x3-play`

Stitch Datasets (5 tasks, Appendix E.3):

- **State-based (3 tasks):**
  - `antmaze-medium-stitch`
  - `humanoidmaze-medium-stitch`
  - `pointmaze-medium-stitch`
- **Pixel-based (2 tasks):**
  - `v-antmaze-medium-stitch`
  - `v-antmaze-large-stitch`

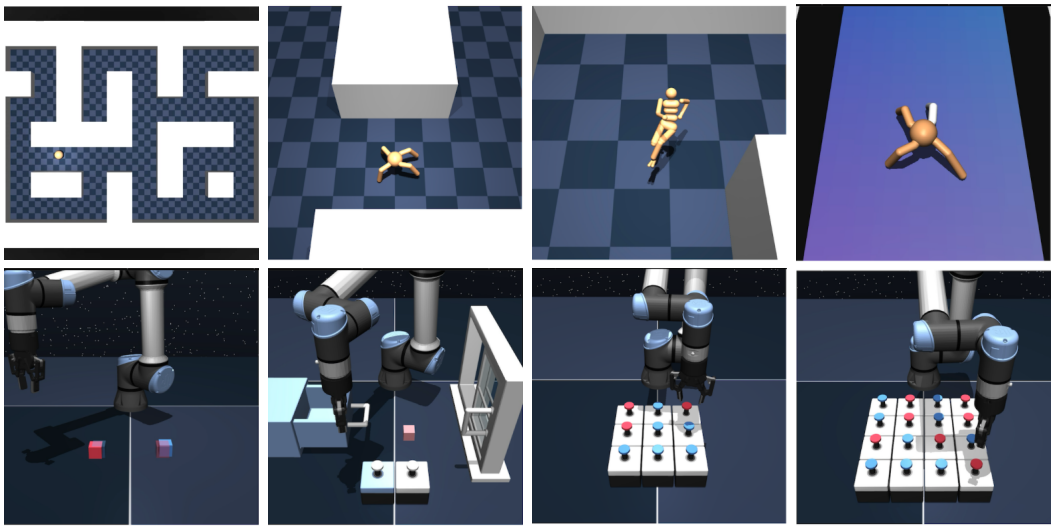


Figure 5: **OGBench environments.** Navigation tasks include AntMaze, PointMaze, and Humanoid-Maze; manipulation tasks include Cube, Scene, and Puzzle.

C Implementation Details

C.1 Hyperparameter

Hyperparameter	Value
Agent	
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning rate	$3e{-4}$
Weight decay	$1e{-4}$
Gradient Steps	1000000 (states), 500000 (pixels)
Batch Size	1024 (states), 256 (pixels)
Policy MLP dimensions	$(512, 512, 512)$
Value MLP dimensions	$(512, 512, 512)$
Nonlinearity	GELU (Hendrycks & Gimpel, 2023)
Target update frequency T_{target}	250
Target policy noise σ	$\mathcal{N}(0, 0.2^2)$
Target policy noise clipping c	$(-0.1, 0.1)$
Discount factor γ	0.995 (`antmaze-giant`, `humanoidmaze`), 0.99 (others)
Reward values	$\{-1, 0\}$
MSPR Prediction Horizon H	5
Q-Horizon n	3
Frame Stacking	3 (Pixel)
BC-Coefficient λ	0.1 (`Locomotion tasks`), 1.0 (`Manipulation Tasks`)
Multi-stage Predictive Architecture	
Encoder module	MLP Block (state), IMPALA (pixel) (Espeholt et al., 2018)
$\mathbf{z}_s$ dimension	512
$\mathbf{z}_{sa}$ dimension	512
$\mathbf{z}_a$ dimension	256
Hidden dimension	512
Learning rate	$1e{-4}$
Weight decay	$1e{-4}$
Multi-stage Predictors Loss	
Dynamics loss weight λ_{dyn}	1.0
Return loss weight λ_{rew}	0.3
Goal-level dynamics loss weight $\lambda_{\text{g-dyn}}$	0.1
Goal-conditioned action loss weight $\lambda_{\text{g-act}}$	0.1

C.2 Network Architecture

This section describes the networks used in our method using PyTorch code (Paszke et al., 2019). We utilize a standardized Multilayer Perceptron (MLP) block for vector inputs and an Impala CNN (Small) (Espeholt et al., 2018; Park et al., 2024a) for image inputs. The state encoder, goal representation, value, and policy networks are described below. the backbone implementation is close to (Fujimoto & Gu, 2021; Fujimoto et al., 2025).

Preamble We define a helper function to initialize MLPs. Unless specified otherwise, our MLPs utilize GELU activations.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from typing import Sequence

def apply_variance_scaling_init(net, scale=1.0):
    fan = (net.weight.size(-2) + net.weight.size(-1)) / 2
    init_w = math.sqrt(scale / fan)
    net.weight.data.uniform_(-init_w, init_w)
    net.bias.data.fill_(0)

def MLP(sizes, activation=nn.GELU, output_activation=nn.Identity,
        layer_norm=True):
    layers = []
    for j in range(len(sizes) - 1):
        # Determine activation for current layer
        act = activation if j < len(sizes) - 2 else output_activation

        # Linear layer setup
        fc = nn.Linear(sizes[j], sizes[j + 1])

        # Apply LayerNorm (Pre-LN) to hidden layers (j > 0)
        if layer_norm and j > 0:
            layers += [nn.LayerNorm(sizes[j], eps=1e-6), fc, act()]
        else:
            layers += [fc, act()]

    return nn.Sequential(*layers)
```

Visual Encoder (Impala CNN) For image-based environments, we utilize the Impala CNN architecture similar to the one used in OGBench Baselines (Park et al., 2024a).

```python
class ResidualBlock(nn.Module):
    def __init__(self, in_ch, features):
        super().__init__()
        self.cnn1 = nn.Conv2d(in_ch, features, 3, stride=1, padding=1)
        self.cnn2 = nn.Conv2d(features, features, 3, stride=1, padding=1)
        self.cnn3 = nn.Conv2d(features, features, 3, stride=1, padding=1)
        self.max_pool = nn.MaxPool2d(3, stride=2, padding=1)

    def forward(self, x):
        x1 = self.max_pool(self.cnn1(x))
        x2 = self.cnn2(F.relu(x1))
        x2 = self.cnn3(F.relu(x2))
        return x1 + x2

class ImpalaCNN(nn.Module):
    def __init__(self, state_dim, out_dim):
        super().__init__()
        self.block1 = ResidualBlock(state_dim, 16)
        self.block2 = ResidualBlock(16, 32)
        self.block3 = ResidualBlock(32, 32)
```

```
        # Assumes standard OGBench input sizes 3x64x64, projecting to out_dim
        self.fc = MLP([2048, out_dim], output_activation=nn.GELU)

    def forward(self, x):
        x = x / 255.0
        x = self.block1(x)
        x = self.block2(x)
        x = self.block3(x).reshape(x.shape[0], -1)
        return self.fc(x)
```

State/Goal Encoder For vector states, the state encoder $\mathcal{E}^s_\psi$ is a three-layer MLP with hidden dimensions of 512. For both state and goal the encoders are shared.

```
class StateEncoder(nn.Module):
    def __init__(self, state_dim, out_dim):
        super().__init__()
        # 3-layer MLP: Input -> 512 -> 512 -> Output
        self.net = MLP([state_dim, 512, 512, out_dim],
                       layer_norm=True, output_activation=nn.GELU)

    def forward(self, state):
        return self.net(state)
```

```
class MSPR(nn.Module):
    def __init__(self,
                 zs_dim: int,
                 action_dim: int,
                 state_encoder: nn.Module,
                 za_dim: int = 256,
                 zsa_dim: int = 512):
        super().__init__()
        self.zs = state_encoder()   # zs

        goal_dim = zs_dim      # zg uses the same encoder

        # f_dyn: zsa -> next zs
        self.model = MLP(sizes=[zsa_dim, zs_dim], activation=nn.GELU,
                         layer_norm=False)
        # f_rew: (zsa, phi) -> return
        self.rew_model = MLP(sizes=[zsa_dim + goal_dim, 1],
                         activation=nn.GELU, layer_norm=False)

        # zsa
        self.za = MLP(sizes=[action_dim, za_dim], activation=nn.GELU,
                         output_activation=nn.GELU, layer_norm=False)
        self.zsa = MLP(sizes=[zs_dim + za_dim, 512, 512, zsa_dim],
                         activation=nn.GELU, layer_norm=True)

        # f_g-act: (zs, zg) -> action
        self.high_inv_model = MLP(sizes=[zs_dim+goal_dim, action_dim],
                                  activation=nn.GELU, layer_norm=False)
        # f_g-dyn: (zs, zg) -> next zs
        self.high_dyn_model = MLP(sizes=[zs_dim+goal_dim, zs_dim],
                                  activation=nn.GELU, layer_norm=False)

    def forward(self, zs, action):
        za = self.za(action)
        return self.zsa(torch.cat([zs, za], -1))

    def model_all(self, zs, zg, action):
        zsa = self.forward(zs, action)

        pred_next_zs = self.model(zsa)
```

```
        pred_reward = self.rew_model(torch.cat([zsa, zg],-1))
        pred_action_high = self.high_inv_model(torch.cat([zs, zg],-1))
        pred_future_zs = self.high_dyn_model(torch.cat([zs, zg],-1))

        return pred_next_zs, pred_action_high, pred_reward,
    pred_future_zs
```

Value Q Networks The value network (Q or V). It takes the concatenated state-action representation z_{sa} and goal representation z_g, and outputs the value of the state-action pair given the goal.

```
class Value(nn.Module):
    def __init__(self, state_dim: int,
                 hidden_dims: Sequence[int] = [512, 512, 512],
                 ):
        super().__init__()

        class ValueNetwork(nn.Module):
            def __init__(self, state_dim):
                super().__init__()
                self.mlp_module = MLP(sizes=[state_dim]+hidden_dims +
    [1], activation=nn.GELU, layer_norm=True)

            def forward(self, zsa, goal):
                return self.mlp_module(torch.concat([zsa, goal], -1))

        self.q1 = ValueNetwork(state_dim)
        self.q2 = ValueNetwork(state_dim)

    def forward(self, zsa, zg):
        return self.q1(zsa, goal), self.q2(zsa, goal)
```

Policy π Network The policy network π processes the state representation z_s and goal representation z_g the output is squashed via a Tanh activation.

```
class Actor(nn.Module):
    def __init__(self, state_dim: int, action_dim: int,
                 hidden_dims: Sequence[int] = [512, 512, 512],
    tansquash=True):
        super().__init__()
        self.actor_net = MLP(sizes=[state_dim] + hidden_dims,
    activation=nn.GELU, output_activation=nn.GELU)
        self.mean_net = nn.Linear(512, action_dim)
        apply_variance_scaling_init(self.mean_net, scale=0.01)
        self.activ = nn.Tanh() if tansquash else nn.Identity()

    def forward(self, zs: torch.Tensor, goal: torch.Tensor):
        outputs = self.actor_net(torch.concat([zs, goal], -1))
        return self.activ(self.mean_net(outputs))
```

C.3 Computational Cost

All experiments were conducted on NVIDIA RTX 4090 GPUs / MIG partitions of H100s. We report approximate wall-clock training times per run for *MSPR* across state-based and pixel-based settings on 4090 Machine. *MSPR* requires ~ 3.5 hours for state-based tasks and ~ 4.5 hours for pixel-based tasks. The additional time stems from the multi-scale predictive modules trained over the horizon H. Notably, the increase in cost when moving from state to pixel inputs is modest ($+1$ hr).

D Baseline Details

D.1 GCRL Baselines.

We evaluate against the standard offline GCRL algorithms provided in the OGBench codebase (Park et al., 2024a): **GCBC**, **GCIQL**, **GCIVL**, **CRL**, **QRL**, and **HIQL**. We use the official implementations and hyperparameter configurations as reported in Park et al. (2024a) without modification, except for the n-step targets of the TD-based methods (Appendix D.1).

Representation Learning Baselines. All Dual Goal Representation (Park et al., 2025b) baselines are taken from the official repository at commit `516c73be6cd470c8748fea6550b609ce0eedd30c`. The original authors provide a pixel-based implementation for **GCIVL** only; we implemented the pixel-based version of **Dual-CRL** ourselves. All extended implementations follow the same architectural and training conventions as the original Dual codebase to ensure a fair comparison. Although there is a discrepancy between the scores reported by OGBench (Park et al., 2024a) for GCIVL and the original scores in Dual Goal Representation (Park et al., 2025b), we report the original scores from OGBench and re-run the Dual backbones for consistency in our comparisons.

n-step Targets To implement the n-step versions of HIQL and GCIVL, we follow the implementation details and official codebase of Park et al. (2025a).

E Additional Results

E.1 GCTD3 vs. *MSPR*: Per-Environment Results

Tables 3 and 4 report the per-environment numbers behind Figure 3 (Sec-5.2). GCTD3 is the TD3+BC backbone of *MSPR* with OGBench-style networks: the actor and critic each learn their own features from their own losses, without the shared predictive encoder. The agent, optimizer, n-step targets, and evaluation protocol are the same as for *MSPR*.

	$n=1$		$n=3$	
Environment	**TD3**	**OURS**	**TD3**	**OURS**
pointmaze-medium-navigate	23 ±9	**50** ±1	**77** ±6	54 ±6
pointmaze-large-navigate	0 ±0	**30** ±2	37 ±2	**48** ±8
antmaze-medium-navigate	64 ±5	**98** ±1	79 ±3	**99** ±1
antmaze-large-navigate	22 ±2	**96** ±1	25 ±4	**98** ±2
antmaze-giant-navigate	0 ±0	**43** ±4	0 ±0	**68** ±7
humanoidmaze-medium-navigate	23 ±4	**35** ±3	26 ±3	**67** ±5
humanoidmaze-large-navigate	1 ±1	**11** ±3	6 ±3	**58** ±3
cube-single-play	**58** ±7	46 ±4	**63** ±4	**65** ±5
cube-double-play	10 ±4	**11** ±2	11 ±1	**13** ±3
scene-play	27 ±4	**30** ±4	42 ±5	**45** ±5
puzzle-3x3-play	**96** ±3	81 ±4	**96** ±2	**98** ±1
puzzle-4x4-play	63 ±7	**73** ±2	88 ±4	**93** ±5
Average	32 ±5	**50** ±3	46 ±4	**67** ±5

Table 3: **GCTD3 backbone vs. *MSPR*, state-based environments**, at $n = 1$ and $n = 3$ target horizons. Mean ± std over 8 seeds.

	$n=1$, w/o FS		$n=3$, w/o FS		$n=3$, FS	
Environment	**TD3**	**OURS**	**TD3**	**OURS**	**TD3**	**OURS**
v-antmaze-medium-navigate	80 ±1	**95** ±1	**84** ±6	**86** ±4	85 ±8	**90** ±3
v-antmaze-large-navigate	30 ±9	**81** ±9	41 ±1	**78** ±5	50 ±4	**81** ±4
v-cube-single-play	30 ±5	**42** ±5	30 ±9	**63** ±1	45 ±8	**82** ±5
v-scene-play	**18** ±5	16 ±3	20 ±3	**25** ±3	30 ±6	**56** ±5
v-puzzle-3x3-play	**20** ±7	13 ±2	**36** ±11	21 ±1	**40** ±7	30 ±4
Average	36 ±6	**49** ±5	42 ±7	**55** ±3	50 ±6	**68** ±4

Table 4: **GCTD3 backbone vs. *MSPR*, pixel-based environments**, with and without frame stacking (FS) at $n = 3$, and without FS at $n = 1$. Mean ± std over 4 seeds.

E.2 Results at Target Horizon (n=1)

Table 5 reports all methods at the published $n = 1$ configuration, with *MSPR* also trained at $n = 1$. Moving from $n = 1$ to $n = 3$ raises *MSPR* from 50 to 67 (state) and 61 to 68 (pixel), which is why Table 1 re-runs every TD-based baseline at $n = 3$.

Environment	GCBC	GCIVL	GCIQL	QRL	CRL	$\text{Dual}_{\text{GCIVL}}$	Dual_{CRL}	HIQL	OURS
State-based environments									
pointmaze-medium-navigate	9 ±6	63 ±6	53 ±8	**82** ±5	29 ±7	76 ±7	33 ±10	**79** ±5	50 ±1
pointmaze-large-navigate	29 ±6	45 ±5	34 ±3	**86** ±9	39 ±7	46 ±6	39 ±12	58 ±5	30 ±2
antmaze-medium-navigate	29 ±4	72 ±8	71 ±4	88 ±3	**95** ±1	75 ±4	93 ±3	**96** ±1	**98** ±1
antmaze-large-navigate	24 ±2	16 ±5	34 ±4	75 ±6	83 ±4	28 ±11	87 ±2	91 ±2	**96** ±1
antmaze-giant-navigate	0 ±0	0 ±0	0 ±0	14 ±3	16 ±3	0 ±0	21 ±4	**65** ±5	43 ±4
humanoidmaze-medium-navigate	8 ±2	24 ±2	27 ±2	21 ±8	60 ±4	29 ±3	57 ±4	**89** ±2	35 ±3
humanoidmaze-large-navigate	1 ±0	2 ±1	2 ±1	5 ±1	24 ±4	3 ±2	18 ±4	**49** ±4	11 ±3
cube-single-play	6 ±2	53 ±4	68 ±6	5 ±1	19 ±2	**89** ±3	60 ±1	15 ±3	46 ±4
cube-double-play	1 ±1	36 ±3	40 ±5	1 ±0	10 ±2	**60** ±4	24 ±5	6 ±2	11 ±2
scene-play	5 ±1	42 ±4	51 ±4	5 ±1	19 ±2	**72** ±6	44 ±5	38 ±3	30 ±4
puzzle-3x3-play	2 ±0	6 ±1	**95** ±1	1 ±0	3 ±1	5 ±1	6 ±1	12 ±2	81 ±4
puzzle-4x4-play	0 ±0	13 ±2	26 ±3	0 ±0	0 ±0	23 ±3	2 ±0	7 ±2	**73** ±2
Avg. State	10 ±1	31 ±1	42 ±1	32 ±1	33 ±1	42 ±1	40 ±2	**50** ±1	**50** ±1
Pixel-based environments									
v-antmaze-medium-navigate	11 ±2	22 ±2	11 ±1	0 ±0	**95** ±1	78 ±4	85 ±3	**93** ±4	**92** ±3
v-antmaze-large-navigate	4 ±0	5 ±1	4 ±1	0 ±0	**84** ±1	40 ±4	43 ±3	53 ±9	77 ±5
v-cube-single-play	5 ±1	60 ±4	30 ±5	41 ±15	31 ±15	58 ±5	14 ±8	**89** ±0	82 ±2
v-scene-play	12 ±2	25 ±2	12 ±2	10 ±1	11 ±2	26 ±5	15 ±1	**49** ±4	36 ±6
v-puzzle-3x3-play	0 ±0	21 ±2	1 ±2	1 ±1	0 ±0	0 ±0	0 ±0	**73** ±8	20 ±1
Avg. Pixel	6 ±1	27 ±1	12 ±1	10 ±3	44 ±3	40 ±2	31 ±2	**71** ±3	61 ±2

Table 5: **OGBench results at $n = 1$.** Baselines use their published configurations; *MSPR* uses $n = 1$ targets (with frame stacking on pixel). Same layout and bold rule as Table 1.

E.3 Results on Stitch Datasets

Table 6 reports results on the OGBench `stitch` datasets, whose trajectories are short fragments that individually do not reach the evaluation goals, so success requires composing behavior across trajectory boundaries. All methods use the same configurations as in Table 1. *MSPR* outperforms both Dual goal-representation methods on every environment, by 22 points on `antmaze-medium-stitch` (78 vs. 56) and 46 on `visual-antmaze-medium-stitch` (83 vs. 37), and has the highest average among non-hierarchical methods in both settings (61 on state, 49 on pixel). HIQL is the strongest method on all five environments (87 and 57 on average), consistent with the advantage of subgoal decomposition when goals lie beyond the span of individual trajectories. Among the remaining baselines, QRL leads on `pointmaze-medium-stitch` (80 vs. 58), where its quasimetric objective matches the purely geometric maze, and GCBC reaches 24 on `visual-antmaze-large-stitch`, where all other methods except HIQL remain at or below 15.

Environment	GCBC	GCIVL	GCIQL	QRL	CRL	$\text{Dual}_{\text{GCIVL}}$	Dual_{CRL}	HIQL	OURS
State-based environments									
antmaze-medium-stitch	45 ±11	44 ±5	34 ±3	59 ±7	53 ±2	40 ±3	56 ±4	**93** ±4	78 ±4
humanoidmaze-medium-stitch	29 ±5	14 ±2	35 ±6	18 ±2	36 ±6	17 ±2	44 ±3	**89** ±5	46 ±3
pointmaze-medium-stitch	23 ±18	75 ±5	27 ±5	**80** ±12	0 ±1	51 ±2	2 ±5	**78** ±3	58 ±3
Avg. State	32 ±7	44 ±2	32 ±3	52 ±5	30 ±2	36 ±1	34 ±2	**87** ±2	61 ±2
Pixel-based environments									
v-antmaze-medium-stitch	64 ±4	16 ±2	12 ±2	0 ±0	69 ±2	26 ±2	37 ±3	**86** ±2	**83** ±3
v-antmaze-large-stitch	24 ±3	1 ±1	0 ±0	1 ±1	11 ±3	1 ±1	10 ±1	**28** ±2	15 ±3
Avg. Pixel	44 ±3	9 ±1	6 ±1	1 ±1	40 ±2	14 ±1	24 ±2	**57** ±1	49 ±2

Table 6: **OGBench results on `stitch` datasets.** Same layout and bold rule as Table 1; TD-based methods use $n = 3$.

F ANALYSIS



Figure 6: *Successful Trajectory*: Goal in top-left. We show the progress, along with the Q values estimated by the Critic for `v-antmaze-large-navigate`.

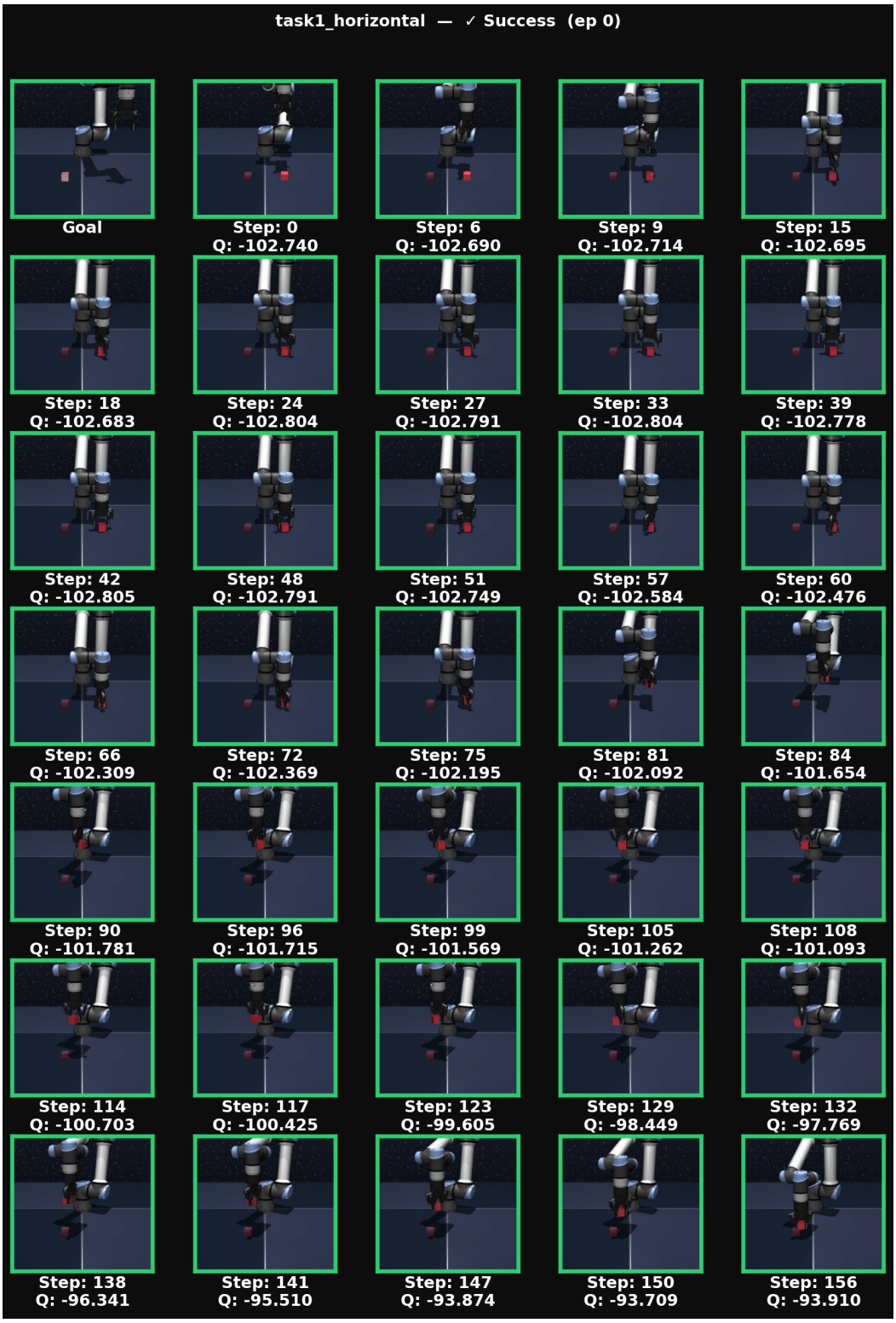


Figure 7: *Successful case* (`Cube-Single-play`): Although the trajectory successfully reaches the goal, the Q-values do not reflect this, because of the behavior cloning term in the actor objective function



Figure 8: *Failure case* : Goal in top-left. We show the progress, along with the Q values estimated by the Critic for `v-antmaze-large-navigate`. Here the agent reaches the state which is similar to the goal but not exactly the goal

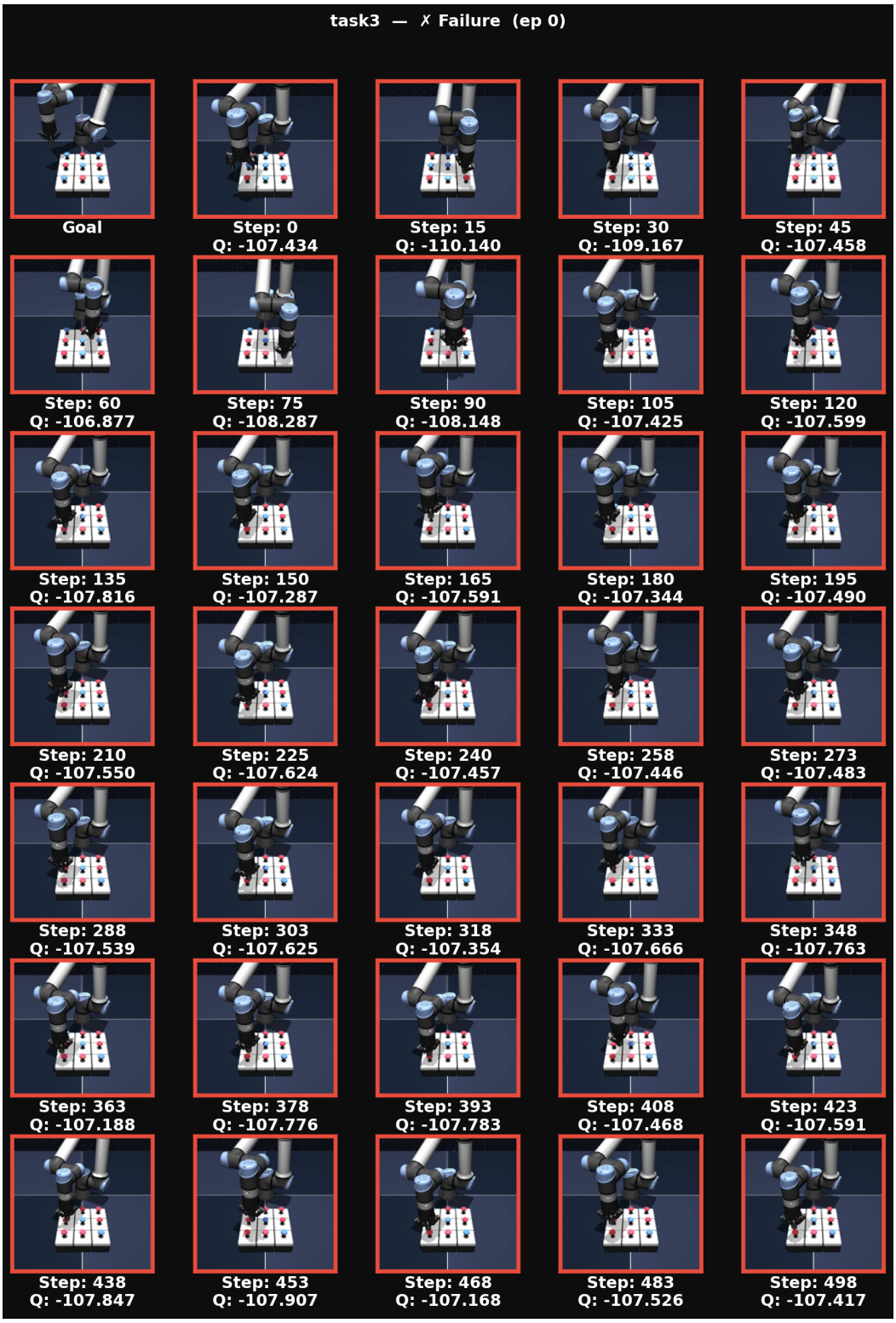


Figure 9: *Failure case* (`puzzle-3x3-play`): Here from step ∼200 onwards, the agent enters a repetitive action loop without making meaningful progress, suggesting the policy has converged to a local behavioral mode that does not correspond to any goal-directed transition in the learned representation