MULTI-SCALE PREDICTIVE REPRESENTATIONS FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-conditioned reinforcement learning (GCRL) requires agents to learn effective state and goal representations, which represents a challenging problem, especially in high-dimensional vision-based environments, as differences in the observations can be uncorrelated with dynamical distances. Classical deep reinforcement learning techniques often fail to capture the alignment between state and goal spaces, requiring additional representation learning techniques. To address this, we propose *Ms.PR*, a representation learning framework that augments model-free GCRL methods with a multi-scale predictive architecture. Leveraging predictive dynamics learning, the latent embedding space captures both physical causality and temporal distances between states. Furthermore, by learning information at multiple timescales, the agent acquires a better understanding of how close and distant goals relate to a given state. We demonstrate that Ms.PR leads to improved representation quality and strong performance on the OGBench benchmark, both on vision and state-based tasks.

1 INTRODUCTION

Advancements in representation learning have played a pivotal role in improving deep reinforcement learning (RL) methods Echchahed & Castro (2025); Nabati et al. (2023). A general strategy consists of using self-supervised objectives to capture important features in large inputs spaces Schwarzer et al. (2021); Srinivas et al. (2020) or predictive models to capture the underlying dynamics of the environment Hafner et al. (2020); Hansen et al. (2023). The adoption of auxiliary models and losses to support learning behaviors with RL helps advancing environment and task understanding beyond trajectory and reward memorization, enabling both efficient adaptation to new environments and strong generalization across diverse tasks Hafner et al. (2023); Rajeswar et al. (2023). Goal-conditioned RL (GCRL) introduces additional challenges compared to single-task RL settings. In order to reach any goal from any state, an agent needs not only to acquire a set of empowering skills, but also to understand the underlying world dynamics Park et al. (2024a). Despite the potential of representation learning to address these issues, current approaches still struggle to scale effectively to complex, high-dimensional GCRL tasks.

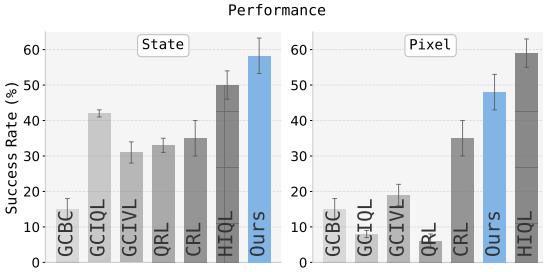

Figure 1: **Aggregate performance on OGBench spanning 23 environments with different dynamics and inputs**. The evaluation includes 12 state-based and 11 pixel-based environments. Our method, Ms.PR, consistently matches or outperforms state-of-the-art approaches, such as GCBC, GCIQL, GCIVL, CRL, and QRL, as well as the hierarchical approach HIQL.

Similarly to other recent works Park et al. (2025b), we believe a fundamental limitation lies in the misalignment between the learned state and goal spaces. Existing methods frequently fail to establish a precise mapping between the agent's current state and target goal representations Shah et al. (2023). When this mapping is imprecise or ambiguous, the agent cannot accurately measure its progress or generate valid plans. From an RL perspective, this can translate into problematic value overestimation

and suboptimal policies Fujimoto et al. (2018), leading to brittle performance where the agent fails to generalize beyond its training distribution.

In this work, we introduce Multi-scale Predictive Representations (Ms.PR), a representation learning framework for GCRL that learns a latent predictive representation of the environment leveraging dynamics information at multiple timescales. Similarly to latent world models, the agent learns a representation that allows single-step dynamics predictions of the environment Ha & Schmidhuber (2018). Additionally, to ensure state-goal relations are properly understood by the agent, a goal-level dynamics of the environment is learned Sharma et al. (2020).

After learning a latent representation that captures dynamics information at multiple timescales, Ms.PR enable actor-critic RL training on the learned latent representation Fujimoto et al. (2025; 2023); Hafner et al. (2020). We empirically validate the performance of our agent on the OGBench offline GCRL benchmark Park et al. (2024a), both on state-based and pixel-based environments. As shown in Figure 1, our representation learning strongly improves performance across several tasks.

In order to confirm the source of the empirical success of our method, we ablate the representation learning module, showing the importance of learning multiscale representations. Furthermore, we analyze the representation learned by Ms.PR and compare it to baselines, providing insights into the effectively reduced value overestimation and improved temporal dynamics understanding.

2 Related Works

Dynamics-based Representation Learning. Leveraging system dynamics is a foundational approach for handling partial observability and feature selection Littman et al. (2001); Parr et al. (2008). In the model-free regime, auxiliary tasks that predict future latent states have proven effective for shaping representations Gelada et al. (2019); Munk et al. (2016); Schwarzer et al. (2021); Guo et al. (2020); Bagatella et al. (2025). These methods share theoretical connections with Model-Based RL, where latent dynamics are explicit and central to planning Ha & Schmidhuber (2018); Hafner et al. (2020; 2019); Schrittwieser et al. (2020); Hansen et al. (2022). In pixel-based control, predictive latent spaces offer rich supervision, generally superior to scalar TD targets, by forcing the agent to understand state transitions and action effects Srinivas et al. (2020); Hafner et al. (2022; 2023). Recent work, such as MRQ Fujimoto et al. (2025), explicitly leverages this signal for Q-learning. We extend these ideas to the goal-conditioned setting, investigating how predictive modeling can best support policy learning when operating entirely in latent space.

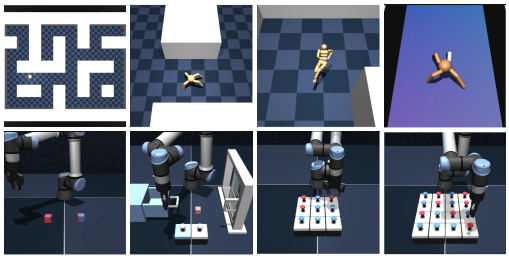

Figure 2: **OGBench environments.** Navigation tasks include AntMaze, PointMaze, and HumanoidMaze; manipulation tasks include Cube, Scene, and Puzzle.

Goal-Conditioned RL (GCRL). GCRL extends standard RL by conditioning policies on specific objectives to generalize across tasks Kaelbling (1993); Lynch et al. (2019); Ghosh et al. (2020). While standard approaches utilize hindsight relabeling Andrychowicz et al. (2018), contrastive objectives Eysenbach et al. (2021; 2023), or state-occupancy matching Ma et al. (2022), visual GCRL faces the critical challenge of representation stability. Drifting representations can degrade performance when observations or goals vary slightly Han et al. (2021). To address this, prior work has explored actionable representations Ghosh et al. (2019), metric-based learning Ma et al. (2023); Shah et al. (2023), and hierarchical abstractions Park et al. (2024b). Our work builds on these foundations, utilizing multi-scale predictive modeling to learn robust representations for goal-conditioned offline RL.

3 Preliminaries

Environment settings. We model the environment as a Goal-Conditioned Markov Decision Process (GC-MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mathcal{G})$. Here, $\mathcal{S}$ denotes the high-dimensional state space, $\mathcal{A}$ the action space, and $\mathcal{P}(s_{t+1}|s_t, a_t)$ the transition dynamics. The goal space $\mathcal{G} \subseteq \mathcal{S}$

consists of desired configurations the agent seeks to reach. Unlike standard RL, the reward function $r(s, a, g)$ is conditioned on a specific goal $g \in \mathcal{G}$, typically formulated as a binary signal where $r = 0$ if the goal is achieved and $r = -1$ 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.

Representation Learning for Offline GCRL. We address the offline setting, where the agent learns from a fixed dataset $\mathcal{D} = \{(s, a, r, s', g)\}$ without further interaction with the environment. Our objective is to learn effective representations that decouple the learning of environment structure from the reinforcement learning objective—a strategy proven beneficial in high-dimensional and sparse-reward environments (Fujimoto et al., 2025; 2023; Hafner et al., 2023).

Formally, we define a state encoder $\mathcal{E}_\psi : \mathcal{S} \to \mathcal{Z}$, mapping states to a latent space $\mathcal{Z}$ such that $\mathbf{z}_s = \mathcal{E}_\psi(s)$. Similarly, we define a goal encoder $\mathcal{E}_\psi(g)$ (sharing weights with the state encoder) and a state-action representation $\mathbf{z}_{sa} = f_\psi(\mathbf{z}_s, a)$. To capture the relational information between states and goals, we employ a projection head ψ_ω to yield a goal embedding $\mathbf{g}_{\text{rep}} = \psi_\omega(\mathbf{z}_s, \mathbf{z}_g)$. Based on these representations, we parameterize the actor and critic as follows:

$$\text{Actor:} \quad \pi_\phi(\mathbf{z}_s, \mathbf{g}_{\text{rep}}), \qquad \text{Critic:} \quad Q_\theta(\mathbf{z}_{sa}, \mathbf{g}_{\text{rep}}).$$

4 METHOD

The primary objective of this work is to learn representations that capture both the underlying physical dynamics and the intent-driven relationship between states and goals, leveraging fixed offline trajectories. A robust representation must satisfy two key desiderata: (1) *sufficiency & discriminability* during training, ensuring the latent space contains enough information for accurate prediction of rewards and transitions (Park et al., 2025b); and (2) *smoothness & robustness* during evaluation, facilitating generalization to unseen goal configurations.

To this end, we propose **Ms.PR**, a representation learning framework based on a multi-scale predictive architecture. Ms.PR learns a latent predictive representation by leveraging dynamics information at two distinct granularities: *local physical dynamics* and *goal-conditioned temporal dynamics*. In the following sections, we detail the components of this framework. An overview is provided in Table 1.

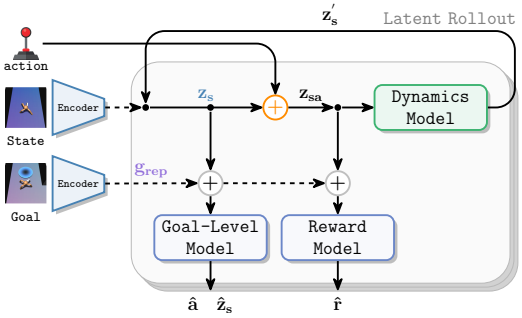

Figure 3: Diagram of the proposed multi-scale predictive representation. A latent rollout is performed by iteratively applying the dynamics and goal-level predictors. The initial state and goal are encoded once and used to initialize the rollout, while the action representation is injected at every step. Thick arrows denote signals propagated at each rollout step, whereas dotted arrows indicate inputs provided only at initialization. The $(+)$ symbol denotes feature concatenation.

4.1 LEARNING ENVIRONMENT DYNAMICS

Model-based auxiliary tasks, such as forward dynamics and reward prediction, are effective regularizers for representation learning in dense-reward settings Fujimoto et al. (2025); Hafner et al. (2023). We employ these predictors to enforce local temporal consistency.

We consider the following predictive modules:

- **Forward dynamics (f_{dyn}):** Predicts the next latent state $\tilde{\mathbf{z}}_{s'}$ given the current state-action latent $\mathbf{z}_{sa}$, enforcing consistency with the environment's transition function.
- **Reward predictor (f_{rew}):** Predicts the immediate reward associated with the current state–action pair and goal configuration. To mitigate reward sparsity and prevent representation collapse, the reward predictor is trained to model Monte Carlo returns rather than single-step rewards. Castro et al. (2022); Echchahed & Castro (2025)

These modules are trained via the following objectives:

$$\mathcal{L}_{\text{dyn}} = \|\tilde{\mathbf{z}}_{s'} - \mathbf{z}_{s'}\|_2^2, \qquad \mathcal{L}_{\text{rew}} = \|\tilde{r} - r\|_2^2,$$

where $\tilde{\mathbf{z}}_{s'} = f_{\text{dyn}}(\mathbf{z}_{sa})$ and $\tilde{r} = f_{\text{rew}}(\mathbf{z}_{sa}, \mathbf{g}_{rep})$. To prevent representational collapse—a common failure mode in latent dynamics learning—targets for the dynamics loss are computed using a target encoder (with stopped gradients).

While reward prediction offers some regularization, it is often insufficient in sparse-reward GCRL tasks. To impose further structure on the latent space, we introduce an inverse dynamics predictor:

- **Inverse dynamics (f_{inv}):** Reconstructs the executed action $\tilde{a}$ given consecutive latent states, enforcing that the latent space retains action-discriminative features.

The inverse dynamics loss is defined as:

$$\mathcal{L}_{\text{inv}} = \|\tilde{a} - a\|_2^2, \quad \text{with} \quad \tilde{a} = f_{\text{inv}}(\mathbf{z}_s, \mathbf{z}_{s'}).$$

4.2 Learning Goal-Level Dynamics

While forward and inverse dynamics capture local physical transitions, they do not explicitly model the long-term dependencies required for goal-reaching. In sparse-reward GCRL, understanding the relationship between the current state and a distant goal is crucial.

To bridge this gap, we introduce goal-level dynamics predictors. These modules integrate the goal embedding $\mathbf{g}_{\text{rep}}$ to model "intent-conditioned" transitions:

- **Goal-level dynamics ($f_{\text{g-dyn}}$):** Predicts the next latent state $\tilde{\mathbf{z}}_{s'}^{\text{goal}}$ based on the current state and the goal embedding. This effectively models the state transition implied by an optimal goal-reaching policy.
- **Goal-conditioned action prediction ($f_{\text{g-act}}$):** Predicts the action $\tilde{\mathbf{a}}^{\text{goal}}$ taken to move towards the goal. This serves as a goal-conditioned behavioral cloning auxiliary task, aligning the representation with the policy's objective.

The corresponding losses are:

$$\mathcal{L}_{\text{g-dyn}} = \|\tilde{\mathbf{z}}_{s'}^{\text{goal}} - \mathbf{z}_{s'}\|_2^2, \qquad \mathcal{L}_{\text{g-act}} = \|\tilde{\mathbf{a}}^{\text{goal}} - a\|_2^2,$$

where $\tilde{\mathbf{z}}_{s'}^{goal} = f_{\text{g-dyn}}(\mathbf{z}_s, \mathbf{g}_{rep})$ and $\tilde{\mathbf{a}}^{goal} = f_{\text{g-act}}(\mathbf{z}_s, \mathbf{g}_{rep})$. Note that unlike the local inverse dynamics which infer actions from state pairs (s, s'), the goal-conditioned action predictor infers actions from (s, g), directly coupling the representation to the control task.

4.3 Agent Training

We jointly optimize the multi-scale representation and the policy using offline data. The training process alternates between (i) horizon-based representation learning on trajectory chunks and (ii) standard actor-critic updates using random batches. Algorithm 1 summarizes the procedure.

Table 1: **Multi-scale Predictive Representations**

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

Latent Representation Learning. Given a trajectory chunk $\tau = \{(s_h, a_h, r_h, s_{h+1})\}_{h=0}^{H-1}$ sampled from $\mathcal{D}$ and a goal g, we encode the initial state and goal as $\mathbf{z}_s^0 = \mathcal{E}_\psi^s(s_0)$ and $\mathbf{z}_g = \mathcal{E}_\psi^s(g)$. We then unroll the latent dynamics model for H steps. At each step h, we compute the action-conditioned latent $\mathbf{z}_{sa}^h = \mathcal{E}_\psi^{sa}(\mathbf{z}_s^h, a_h)$ and the goal-conditioned context $\mathbf{g}_{\text{rep}}^h = \psi_\omega(\mathbf{z}_s^h, \mathbf{z}_g)$. The model then predicts the

next latent state, reward, and goal-consistent action. Crucially, regression targets for state predictions are generated by the slow-moving target encoder, $\mathbf{z}_s^{h+1} = \mathcal{E}_{\bar{\psi}}^{s}(s_{h+1})$.

The total representation learning objective aggregates the multi-step prediction errors over the horizon H:

$$\mathcal{L}_{\text{Ms.PR}} = \sum_{h=0}^{H-1} \Big(\lambda_{\text{dyn}}\, \mathcal{L}_{dyn} + \lambda_{\text{inv}}\, \mathcal{L}_{inv} + \lambda_{\text{g-dyn}}\, \mathcal{L}_{g-dyn} + \lambda_{\text{g-act}}\, \mathcal{L}_{g-act} + \lambda_{\text{rew}}\, \mathcal{L}_{rew} \Big). \quad (1)$$

This horizon-based training ensures that the learned latent space captures both local physics (via $\mathcal{L}_{dyn}, \mathcal{L}_{inv}$) and longer-horizon, goal-directed intent (via $\mathcal{L}_{g-dyn}, \mathcal{L}_{g-act}$), providing a rich substrate for the actor and critic.

Goal-conditioned RL For value learning in the critic, we minimize the Huber loss Fujimoto et al. (2025) against an n-step TD target. Let $R_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k}$ denote the n-step discounted return and $\mathbf{z}'_{sa}$ the representation of the state-action pair at step $t+n$. The critic loss is defined as:

$$\mathcal{L}_Q = \Big\| Q_\theta(\mathbf{z}_{sa}, \mathbf{g}_{\text{rep}}) - \Big(R_t^{(n)} + \gamma^n Q_{\theta'}(\mathbf{z}'_{sa}, \mathbf{g}_{\text{rep}}) \Big) \Big\|_\delta, \quad (2)$$

where $Q_{\theta'}$ denotes the target critic. The actor is trained to find the action policy that maximizes the estimated value while staying close to the behavior distribution of the offline dataset, akin to Fujimoto & Gu (2021):

$$\mathcal{L}_\pi = -Q_\theta(\mathbf{z}_{sa}, \mathbf{g}_{\text{rep}}) + \lambda_{\text{BC}} \|\pi_\phi(\mathbf{z}_s, \mathbf{g}_{\text{rep}}) - a\|_2^2. \quad (3)$$

Algorithm 1 Representation and Policy Learning

Input: Replay buffer $\mathcal{D}$, 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: $\bar{\psi} \leftarrow \psi,\ \bar{\phi} \leftarrow \phi,\ \bar{\theta} \leftarrow \theta$
    Sample trajectory chunk $\tau \sim \mathcal{D}$
    Update representation (ψ, ω) by minimizing $\mathcal{L}_{\text{Ms.PR}}(\tau)$ (Eq. 1)
  end if
  Sample batch $(s, a, r, s', g) \sim \mathcal{D}$
  Update critic θ using TD loss $\mathcal{L}_Q$ (Eq. 2)
  Update actor ϕ using $\mathcal{L}_\pi$ (Eq. 3)
end for

Optimization. As shown in Algorithm 1, representation updates are performed every K steps using trajectory chunks, while the actor and critic are updated at every step. Target networks are updated periodically via hard updates.

5 Experiments

Our experimental evaluation focuses on assessing the representation learned by Ms.PR in offline GCRL settings. We adopt multiple environments from the OGBench benchmark suite, including both locomotion and manipulation tasks, and spanning both state-based and pixel-based inputs.

In addition to assessing Ms.PR performance on the benchmark, we aim to answer the following questions: (i) which components of Ms.PR contribute to its success?, (ii) how important are predictions from the model for succeeding in solving the task?, and (iii) how does the improved representation affect value and policy learning?

5.1 Representation Learning in Pixel-based Tasks

We evaluate performance on challenging pixel-based benchmarks, summarizing the results in Table-2. HIQL, which combines representation learning and hierarchical policy learning, performs best in this

	Non-hierarchical						Hierarchical
Dataset	**GCBC**	**GCIVL**	**GCIQL**	**QRL**	**CRL**	**Ms.PR**	**HIQL**
v-antmaze-medium-navigate	11 ±2	22 ±2	11 ±1	0 ±0	**95** ±1	**94** ±2	**93** ±4
v-antmaze-large-navigate	4 ±0	5 ±1	4 ±1	0 ±0	**84** ±1	74 ±6	53 ±9
v-antmaze-teleport-navigate	5 ±1	8 ±1	6 ±1	6 ±3	**48** ±2	33 ±7	37 ±2
v-antmaze-medium-stitch	67 ±4	6 ±2	2 ±0	0 ±0	69 ±2	**87** ±4	**87** ±2
v-antmaze-large-stitch	24 ±3	1 ±1	0 ±0	1 ±1	11 ±3	14 ±3	**28** ±2
v-antmaze-teleport-stitch	32 ±3	1 ±1	1 ±0	1 ±2	32 ±6	10 ±2	**37** ±4
v-cube-single-play	5 ±1	60 ±5	30 ±5	41 ±15	31 ±15	**86** ±4	**89** ±0
v-cube-double-play	1 ±1	10 ±2	1 ±1	5 ±0	2 ±1	10 ±2	**39** ±2
v-scene-play	12 ±2	25 ±3	12 ±2	10 ±1	11 ±2	**56** ±5	49 ±4
v-puzzle-3x3-play	0 ±0	21 ±1	1 ±2	1 ±1	0 ±0	37 ±5	**73** ±8
v-puzzle-4x4-play	10 ±1	**60** ±5	16 ±4	0 ±0	10 ±6	13 ±2	**60** ±41
Average	10 ±3	19 ±3	8 ±1	6 ±2	35 ±5	47 ±4	**59** ±4

Table 2: **Comparison of Ms.PR with offline GCRL methods Park et al. (2024a) across 11 pixel-based OGBench environments.** Ms.PR demonstrates competitive overall performance. Numbers at or above 95% of the best value in the row are highlighted in bold.

experiment with Ms.PR following. We note that using two policies, as opposed to one, to reduce the action prediction horizon is a well-established approach to improve performance on the OGBench benchmark (Park et al., 2025a) that is orthogonal to our representation learning approach. In the future, it would be interesting to see how our representation could be combined with hierarchical policy learning to further improve performance.

Among the non-hierarchical approaches, Ms.PR significantly outperforms other strong baselines such as CRL and GCIVL, achieving an average success rate of 47% compared to 35% for CRL. In navigation and stitching tasks, Ms.PR demonstrates that a good representation can help achieve long-horizon capabilities. In `v-antmaze-medium-navigate`, Ms.PR achieves 94% success, effectively matching the hierarchical HIQL (93%). More notably, in the challenging `v-antmaze-medium-stitch` task—which requires stitching suboptimal visual trajectories—Ms.PR attains 87%, tying with HIQL and significantly surpassing the strongest non-hierarchical baseline CRL (69%). This indicates that by aligning the latent space with the underlying dynamics, Ms.PR allows the policy to reason over long horizons without explicit temporal abstraction.

Ms.PR excels in manipulation tasks, achieving 86% in `v-cube-single-play` and achieving state-of-the-art performance in `v-scene-play` with 56%, outperforming even the hierarchical HIQL (49%). We hypothesize that standard baselines struggle here because they discard fine-grained contact information as visual noise. Ms.PR mitigates this by enforcing *dynamic consistency* via the inverse dynamics auxiliary loss. This ensures that the latent space distinguishes between random visual changes and purposeful, action-driven contact, allowing the actor to learn precise manipulation policies even from compressed latent states.

	Non-hierarchical						Hierarchical
Dataset	**GCBC**	**GCIVL**	**GCIQL**	**QRL**	**CRL**	**Ms.PR**	**HIQL**
pointmaze-medium-navigate	9 ±6	63 ±6	53 ±8	**82** ±5	29 ±7	64 ±9	**79** ±5
pointmaze-large-navigate	29 ±6	45 ±5	34 ±3	**86** ±9	39 ±7	54 ±9	58 ±5
antmaze-medium-navigate	29 ±4	72 ±8	71 ±4	88 ±3	**95** ±1	**95** ±6	**96** ±1
antmaze-large-navigate	24 ±2	16 ±5	34 ±4	75 ±6	83 ±4	**95** ±3	**91** ±2
antmaze-giant-navigate	0 ±0	0 ±0	0 ±0	14 ±3	16 ±3	48 ±6	**65** ±5
humanoidmaze-medium-navigate	8 ±2	24 ±2	27 ±2	21 ±8	60 ±4	55 ±4	**89** ±2
humanoidmaze-large-navigate	1 ±0	2 ±1	2 ±1	5 ±1	24 ±4	41 ±14	**49** ±4
cube-single-play	6 ±2	53 ±4	**68** ±6	5 ±1	19 ±2	**65** ±8	15 ±3
cube-double-play	1 ±1	36 ±3	**40** ±5	1 ±0	10 ±2	12 ±4	6 ±2
scene-play	5 ±1	42 ±4	**51** ±4	5 ±1	19 ±2	**49** ±8	38 ±3
puzzle-3x3-play	2 ±0	6 ±1	**95** ±1	1 ±0	3 ±1	75 ±5	12 ±2
puzzle-4x4-play	0 ±0	13 ±2	26 ±3	0 ±0	0 ±0	**47** ±7	7 ±2
Average	15 ±3	31 ±3	42 ±1	33 ±2	35 ±5	**58** ±7	50 ±4

Table 3: **Comparison of Ms.PR with offline GCRL methodsPark et al. (2024a)) across 12 state-based OGBench environments.** Ms.PR demonstrates competitive performance overall. Numbers at or above 95% of the best value in the row are highlighted in bold.

5.2 Representation Learning in State-Based Tasks

While representation learning has shown large benefits in pixel-based environments (Hafner et al., 2023), we also study Ms.PR on state-based benchmarks to verify that our representation learning objective captures valid temporal structures directly from low-dimensional observations.

As shown in Table 3, Ms.PR excels from state-based inputs, outperforming all approaches, including the hierarchical policy HIQL approach.. On `antmaze-large`, we achieve 95% success, strictly surpassing standard end-to-end methods like GCIQL (34%) and GCIVL (16%). This result reinforces the findings from the visual experiments: by explicitly training the high-level model to predict long-horizon returns, Ms.PR learns a "distance-aware" manifold that facilitates goal-conditioned RL. In manipulation environments, Ms.PR performance is competitive with GCIQL, which generally outperforms other approaches in these settings.

Q: Does a better representation lead to a better policy?

In addition to the OGBench baselines, we compare Ms.PR against VIB (Alemi et al., 2019), VIP (Ma et al., 2023), TRA (Myers et al., 2025), BYOL-γ (Lawson et al., 2025), and Dual (Park et al., 2025b). These are approaches that, similarly to our work, specifically focus on improving the representation learned, decoupling the RL components. Crucially, to avoid underestimating baseline performance, we allow each representation learning method to be paired with its strongest associated downstream algorithm (selected among GCIVL, CRL, and GCFBC), reporting the maximum performance found in (Park et al., 2025b). Despite this advantageous setup for the baselines, Ms.PR achieves superior results. Figure 4 aggregates the performance across all 12 state-based environments. Ms.PR achieves an average success rate of 58%, setting a new state-of-the-art for representation learning approaches on this benchmark. Notably, we outperform strong representation learning baselines such as TRA (42%) and BYOL-γ (41%) by a wide margin. This confirms that even when baselines are optimized with their ideal downstream algorithms, the structural advantage of Ms.PR, specifically the alignment of dynamics and goal representations, provides a more robust foundation for offline policy learning.

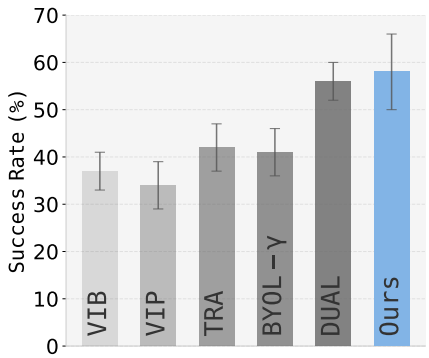

Figure 4: **Aggregate Performance on State-Based Tasks.** Ms.PR achieves the highest average success rate (58%), outperforming strong baselines like Dual (56%). Detailed results in Appendix.

Q: Is the goal-level dynamics critical for GCRL?

Hypothesis: In Model-Based RL, a common failure mode is "objective mismatch", where a model achieves low prediction error but fails to capture features relevant for control. Standard reconstruction losses only ensure that $z_{s'}$ is predictable from z_s. However, this does not inform the agent on whether $z_{s'}$ is useful for reaching the goal z_g. We hypothesize that our high-level dynamics objective (f_{g-dyn}) aligns the representation specifically for goal-directed control.

Finding: As shown in Figure 5, we observe a strong correlation between representation quality and task success. While the dynamics error is negligible in simpler tasks (locomotion), complex environments (manipulation) show a clear correlation between performance drops and increased *Goal-Level Dynamics Error*. This identifies goal-level loss as a critical proxy for controllability: a failure to predict the goal-conditioned abstract state directly impairs the policy's navigation. Ultimately, representation quality proves to be the bottleneck for downstream control, necessitating precise goal-level dynamics understanding to operate effectively.

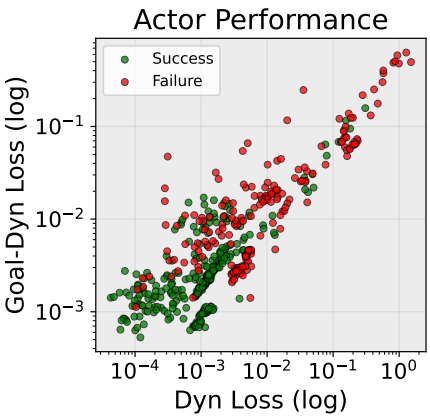

Figure 5: **Relationship between dynamics quality and downstream performance.** Lower prediction loss consistently corresponds to higher task success and improved actor performance, indicating that poor representation learning directly degrades downstream control.

5.3 Analysis and Ablations

To understand the mechanism behind our performance gains, we investigate the relationship between the quality of the learned representation and the downstream policy performance. We specifically analyze whether the auxiliary objectives in Ms.PR translate to better control.

To isolate the contribution of each component in our multi-scale architecture, we conduct an ablation study by incrementally adding auxiliary objectives to the world model. We define four variants: `D` (Dynamics only), `DI` (Dynamics + Inverse Dynamics), `DIR` (Dynamics + Inverse Dynamics + Reward Predictor), and the full **Ms.PR** model (which adds High-Level Goal Dynamics and High-Level Inverse Dynamics). The aggregate results are visualized in Figure 6, with detailed per-task performance reported in Table 5.

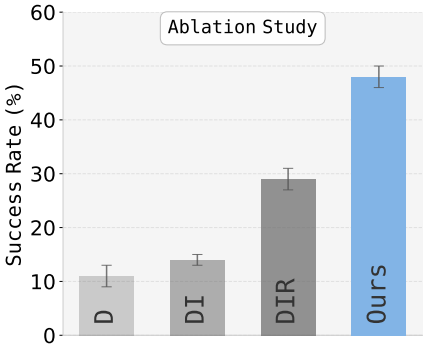

Figure 6: Ablation study of Ms.PR and its variants: `D` (Dynamics only), `DI` (Dynamics + Inverse Dynamics), `DIR` (Dynamics + Inverse Dynamics + Reward Predictor), and the full Ms.PR model. Detailed results are reported in Table 5 in Appendix D.

Role of Action Consistency (`D` to `DI`). The base model `D` achieves a low average success rate of 11%. Adding the inverse dynamics objective in `DI` yields a modest improvement to 14%. This suggests that while predicting future states is necessary, it is not sufficient for control if the latent transitions are not grounded in the agent's actions. `DI` begins to recover performance in simple manipulation tasks (e.g., `v-cube-single-play` improves from 8% to 29%), where action-state causality is immediate, but it fails to scale to long-horizon navigation.

Role of Task Awareness (`DI` to `DIR`). Incorporating the reward predictor in `DIR` results in a significant performance jump to 29%. By forcing the latent space to encode task-relevant reward signals, the model becomes task-aware. This is particularly effective in straightforward navigation tasks; for instance, `v-antmaze-medium-navigate` improves drastically from 32% to 86%. However, `DIR` still struggles with tasks requiring long-horizon reasoning, such as stitching and complex manipulation (e.g., `v-scene-play` remains at 32%), indicating that local reward prediction is insufficient for solving temporal credit assignment in high-dimensional spaces.

Role of High-Level Goal Alignment (`DIR` to Ms.PR). The maximum improvement comes from adding the goal-level dynamics and inverse models. Ms.PR achieves an overall success rate of 47%, nearly doubling the performance of `DIR`. This jump is driven by the model's newfound ability to reason over long horizons.

- *Stitching:* In `v-antmaze-medium-stitch`, where the agent must connect suboptimal trajectories, performance triples from 28% (`DIR`) to 87% (Ms.PR). The goal-level dynamics model effectively bridges the gap between distant states.
- *Long-horizon Manipulation:* In `v-scene-play`, which involves multi-stage object interaction, success rises from 32% to 57%.

These results confirm that the "Multi-scale" component of Ms.PR, specifically the goal-level alignment between state and goal representations, is the decisive factor for scaling to complex, offline goal-conditioned tasks. **Value Estimation Accuracy.** Finally, we investigate the root cause of the performance gap by analyzing the quality of the learned value functions. A fundamental challenge in offline GCRL is "value estimation error," where the critic fails to accurately predict returns for states outside the immediate training distribution. To quantify this, we compute the difference between the agent's learned Q-values and the ground-truth Monte Carlo (MC) returns relative to a fixed goal. We perform this study on the antmaze-large-navigate environment, from pixel-based inputs.

As visualized in Figure-7, baseline methods such as GCIQL and GCIVL exhibit high value estimation error (indicated by dark purple regions), particularly exhibiting a classical phenomenon known as value overestimation (Fujimoto et al., 2018). These high-error regions effectively blind the policy, preventing it from distinguishing between viable paths and dead ends. In contrast, Ms.PR (left)

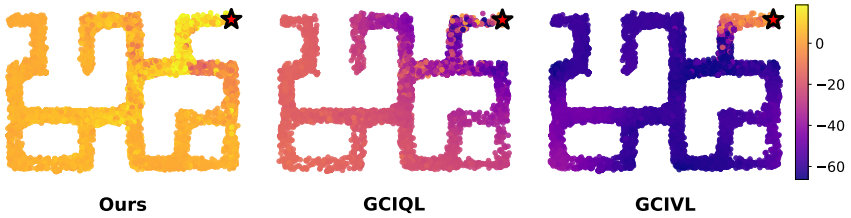

Figure 7: **Value Error Heatmap.** Visualization of value prediction error for a fixed goal across varying state locations, computed by comparing Monte Carlo rollouts with the agent's value estimates for Ms.PR, GCIQL, and GCIVL. Additional Results in Appendix-E

maintains a consistently low error profile (indicated by bright yellow regions) across the maze. By anchoring the critic to a dynamically consistent latent space and utilizing Monte Carlo targets, Ms.PR significantly reduces estimation bias. This precise value alignment is the "major problem" that previous approaches fail to solve, and it is the primary driver behind our agent's ability to navigate complex, long-horizon environments. Additional Details in appendix-E

Overall Insights. The consistent performance improvement across diverse domains highlights that Ms.PR successfully extracts rich, goal-aware representations from fixed offline datasets. Unlike traditional methods that often require increased model capacity or online interaction to resolve visual ambiguity, our approach maximizes the utility of available data through auxiliary dynamic consistency objectives. This formulation is particularly effective in the offline setting: by enforcing that the representation remains predictive of high-level intent, Ms.PR constrains the learned manifold to the valid physical support of the environment. This prevents the encoder from drifting into "unknown" or undefined latent regions, which is a common failure mode when learning complex representations from finite data.

6 Conclusion

In this work, we introduced **Ms.PR**, a predictive representation framework for goal-conditioned reinforcement learning that learns predictive latent representations with explicit goal awareness. Ms.PR focuses on representation alignment: shaping the latent space so that state, action, and goal embeddings are dynamically and semantically consistent.

At the core of Ms.PR is a multi-scale predictive architecture that augments model-free GCRL with auxiliary modules, learning the dynamics of the environment and the relationship between states and goals. These signals encourage the latent representation to capture both fine-grained dynamics and goal-relevant structure, improving the agent's ability to reason about long-horizon behavior. Empirically, Ms.PR achieves competitive performance across 23 OGBench environments (12 state-based and 11 pixel-based). These results demonstrate that performance gains in offline RL stem from structurally aligned and predictive representations.

Overall, our findings highlight that dynamically-consistent and goal-aware representations provide a powerful solution for offline GCRL. Nonetheless, limitations still apply to our approach. As we observed in the pixel-based experiments, hierarchical approaches for representation and policy learning seem necessary for achieving superior performance. In future work, we would like to investigate further how representation learning can enable horizon reduction for policy learning, without requiring hierarchical approaches.

In this work, we also did not investigate the advantage of our architecture for online GCRL. While this setting introduces additional complexities, such as non-stationarity in the data distribution and suboptimal policies, it would be interesting to see how multi-scale predictions could benefit online GCRL methods as well.

REFERENCES

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

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018. URL https://arxiv.org/abs/1707.01495.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. Td-jepa: Latent-predictive representations for zero-shot reinforcement learning, 2025. URL https://arxiv.org/abs/2510.00739.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes, 2022. URL https://arxiv.org/abs/2106.08229.

Ayoub Echchahed and Pablo Samuel Castro. A survey of state representation learning for deep reinforcement learning, 2025. URL https://arxiv.org/abs/2506.17518.

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, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification, 2021. URL https://arxiv.org/abs/2011.08909.

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, Wei-Di Chang, Edward J. Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning, 2023. URL https://arxiv.org/abs/2306.02451.

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.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies, 2019. URL https://arxiv.org/abs/1811.07819.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning, 2020. URL https://arxiv.org/abs/1912.06088.

Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning, 2020. URL https://arxiv.org/abs/2004.14646.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

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.

Beining Han, Chongyi Zheng, Harris Chan, Keiran Paster, Michael R. Zhang, and Jimmy Ba. Learning domain invariant representations in goal-conditioned block mdps, 2021. URL https://arxiv.org/abs/2110.14248.

Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations, 2022. URL https://arxiv.org/abs/2212.05698.

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.

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.

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.

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, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression, 2022. URL https://arxiv.org/abs/2206.03023.

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.

Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4667–4673, 2016. doi: 10.1109/CDC.2016.7798980.

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.

Ofir Nabati, Guy Tennenholtz, and Shie Mannor. Representation-driven reinforcement learning, 2023. URL https://arxiv.org/abs/2305.19922.

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

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

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.

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels, 2023. URL https://arxiv.org/abs/2209.12016.

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.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models, 2023. URL https://arxiv.org/abs/2104.05859.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills, 2020. URL https://arxiv.org/abs/1907.01657.

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

A Implementation Details

A.1 Hyperparameter

Table 4: **Ms.PR Hyperparameters**

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	$(512, 512)$ (State) & $(512, 512, 512)$ (State)
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	$\{-1, 0\}$
Multi-step return Horizon H_Q	3
Image augmentation probability	0.5 (pixel-based `Manipulation Tasks`), 0 (others)
Frame Stacking	3 (pixel-based `Manipulation Tasks`), 0 (others)
BC-Coefficient λ	0.1 (`Locomotion tasks`), 1.0 (`Manipulation Tasks`) 0.01 (`Pointmaze`), 3.0 (`Visual-Cube`)
Multi-stage Predictive Architecture	
Encoder module	MLP Block (state), IMPALA (pixel) Espeholt et al. (2018)
Goal representation dim g_{rep}	128
Length normalization	True
$\mathbf{z}_s$ dimension	512
$\mathbf{z}_{sa}$ dimension	512
$\mathbf{z}_a$ dimension	512
Hidden dimension	512
Learning rate	$1e{-4}$
Weight decay	$1e{-4}$
Multi-stage Predictors Loss	
Dynamics loss weight λ_{dyn}	1.0
Reward loss weight λ_{reward}	0.3(state-based), 0.5 (pixel-based)
Inverse dynamics loss weight λ_{idyn}	0.1
High-level dynamics loss weight λ_{hdyn}	0.3 (pixel-based `Manipulation Tasks`), 0.1 (others)
High-level inverse dynamics loss weight λ_π	0.1

A.2 Network Architecture

This section describes the networks used in our method using PyTorch code blocks. 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.

Preamble

We define a helper function to initialize MLPs. Unless specified otherwise, our MLPs utilize GELU activations. When LayerNorm is enabled, we apply a Pre-LN formulation (normalization applied before the linear layer) for all layers except the input layer.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

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 (Park et al., 2024a). The network consists of three residual blocks. Each block is composed of a convolutional layer, a max-pooling layer, and two subsequent convolutional layers with a residual connection. The resulting feature map is flattened and projected via a linear layer.

```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, 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 and Goal Encoders

For vector states, the state encoder ϕ_s is a three-layer MLP with hidden dimensions of 512. The goal representation ϕ_g (optionally used for goal-conditioned tasks) follows a similar architecture but may include an additional Length Normalization layer to constrain the embedding magnitude.

```
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 GoalEncoder(nn.Module):
    def __init__(self, input_dim, goal_dim, use_ln=False):
        super().__init__()
        self.net = mlp([input_dim, 512, 512, goal_dim],
                       layer_norm=True, activation=nn.GELU)
        self.ln = LengthNormalize() if use_ln else nn.Identity()

    def forward(self, x):
        return self.ln(self.net(x))
```

Multi-scale Predictive Architecture

The multi-scale predictive architecture is designed to learn a compact latent representation of the environment's dynamics. In addition to the forward dynamics and reward predictors, we include inverse dynamics models for both low-level action recovery and high-level temporal abstractions, as well as a high-level dynamics model for long-horizon prediction.

```
class MsPred(nn.Module):
    def __init__(self, state_dim, action_dim, latent_dim=512,
                 act_chunk=5):
        super().__init__()

        # --- Encoders ---
        # Action Encoder: Action -> 256
        self.za_enc = mlp([action_dim, 256],
                          activation=nn.GELU, layer_norm=False)

        # State-Action Encoder: (State, Latent Action) -> Latent Z_sa
        self.zsa_enc = mlp([state_dim + 256, 512, 512, latent_dim],
                           activation=nn.GELU, layer_norm=True)

        # --- Forward Prediction ---
        # Dynamics: Latent Z_sa -> Next State Embedding
        self.dynamics = mlp([latent_dim, state_dim],
                            activation=nn.GELU, layer_norm=False)

        # Reward: (Latent Z_sa, Goal) -> Reward Bins
        self.reward = mlp([latent_dim + state_dim, 21],
                          activation=nn.GELU, layer_norm=False)

        # --- Inverse & High-Level Models ---
```

```
        # Low-Level Inverse: (State, Next State) -> Action
        self.low_inv = mlp([2 * state_dim, action_dim],
                           activation=nn.GELU, layer_norm=False)

        # High-Level Inverse: (State, Goal) -> Sequence of Actions
        self.high_inv = mlp([state_dim + state_dim, act_chunk * action_dim],
                            activation=nn.GELU, layer_norm=False)

        # High-Level Dynamics: (State, Goal) -> Future State
        self.high_dyn = mlp([state_dim + state_dim, state_dim],
                            activation=nn.GELU, layer_norm=False)

    def forward(self, zs, action, next_zs, goal_zs):
        # 1. Encode State-Action
        za = self.za_enc(action)
        zsa = self.zsa_enc(torch.cat([zs, za], dim=-1))

        # 2. Predictions
        pred_next_zs = self.dynamics(zsa)
        pred_reward = self.reward(torch.cat([zsa, goal_zs], dim=-1))

        # 3. Auxiliary Predictions
        pred_act_low = self.low_inv(torch.cat([zs, next_zs], dim=-1))
        pred_act_high = self.high_inv(torch.cat([zs, goal_zs], dim=-1))
        pred_future_zs = self.high_dyn(torch.cat([zs, goal_zs], dim=-1))

        return pred_next_zs, pred_reward, pred_act_low, pred_act_high, pred_future_zs
```

Value Q Networks

The value network (Q or V) is composed of a four-layer MLP with hidden dimensions of 512. It takes the concatenated state and goal (if applicable) as input. We utilize Pre-LayerNorm configurations for training stability.

```
class ValueNetwork(nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        # 4-layer MLP: Input -> 512 -> 512 -> 512 -> 1
        self.net = mlp([input_dim, 512, 512, 512, 1],
                       layer_norm=True, activation=nn.GELU)

    def forward(self, zsa, goal=None):
        x = torch.cat([zsa, goal], dim=-1) if goal is not None else state
        return self.net(x)
```

Policy π Network

The policy network π processes the state representation zs and goal representation through a 3 or 2 layer MLP backbone with hidden dimensions of 512. The output head projects these features to the action dimension. For continuous control tasks, the output can be optionally squashed via a Tanh activation.

```
class PolicyNetwork(nn.Module):
    def __init__(self, input_dim, action_dim):
        super().__init__()
        # Backbone: Input -> 512 -> 512 -> 512
        self.backbone = mlp([input_dim, 512, 512, 512],
                            layer_norm=True, output_activation=nn.GELU)
        self.head = nn.Linear(512, action_dim)
```

```
    def forward(self, zs, goal=None):
        x = torch.cat([zs, goal], dim=-1) if goal is not None else state
        feats = self.backbone(x)
        return torch.tanh(self.head(feats))
```

B Baseline Details

Representation Learning Baselines. We implement all goal representation learning methods strictly following the Dual Goal Representation paper. **VIB**Shah et al. (2023) learns a stochastic goal embedding via an information bottleneck objective and is trained through the downstream value loss, with a KL regularization coefficient selected from $\{0.001, 0.003\}$. **TRA** Myers et al. (2025) trains goal representations using a symmetric InfoNCE-based CRL loss, where gradient flow from the downstream objective is disabled for base algorithm GCIVL, as prescribed in the original work. **BYOL-γ** Lawson et al. (2025) learns goal embeddings via a bidirectional temporal self-predictive loss, using the representation only for goals (not states), with downstream gradient disabled for the base algorithm GCIVL. **VIP** Ma et al. (2023) learns a metric-based goal representation by parameterizing the value function as a negative squared distance in embedding space and uses the learned representation exclusively for goal conditioning in downstream policies and critics. For the main comparison table, we use **GCIVL** as the downstream algorithm, as it has better overall performance in comparison to CRLEysenbach et al. (2023) and goal-conditioned flow BC (GCFBC)Park et al. (2025a) across tasks, consistent with findings in the Dual Goal Representation paper Park et al. (2025b). Additional baselines including **GCIQL**, **GCBC**, **CRL**, **QRL**, and **HIQL** are evaluated, with full results reported in Appendix D.

B.1 Baseline Implementation

We utilize the OGBench Park et al. (2024a) codebase and benchmark for the following algorithms (**GCIQL**, **GCBC**, **CRL**, **QRL**, and **HIQL**), and representation learning baseline from the implementation of dual goal representation Park et al. (2025b) for equal comparison.

B.2 Compute

We implement Ms.PR using the PyTorch framework Paszke et al. (2019) and conduct all experiments on an internal GPU cluster equipped with NVIDIA RTX 4090 and H100 GPUs. Each experiment on state-based environments completes within 4 hours, while experiments on pixel-based environments require no more than 7 hours.

B.3 Baseline Hyperparameter Tuning

To ensure a fair and strong comparison, we do not fix baseline hyperparameters across all environments. Instead, each baseline is trained with the best-performing hyperparameter configurations as reported in prior works Park et al. (2024a; 2025b). This protocol prevents underestimating baseline performance and ensures that reported improvements are not artifacts of suboptimal tuning.

C PERFORMANCE PLOT

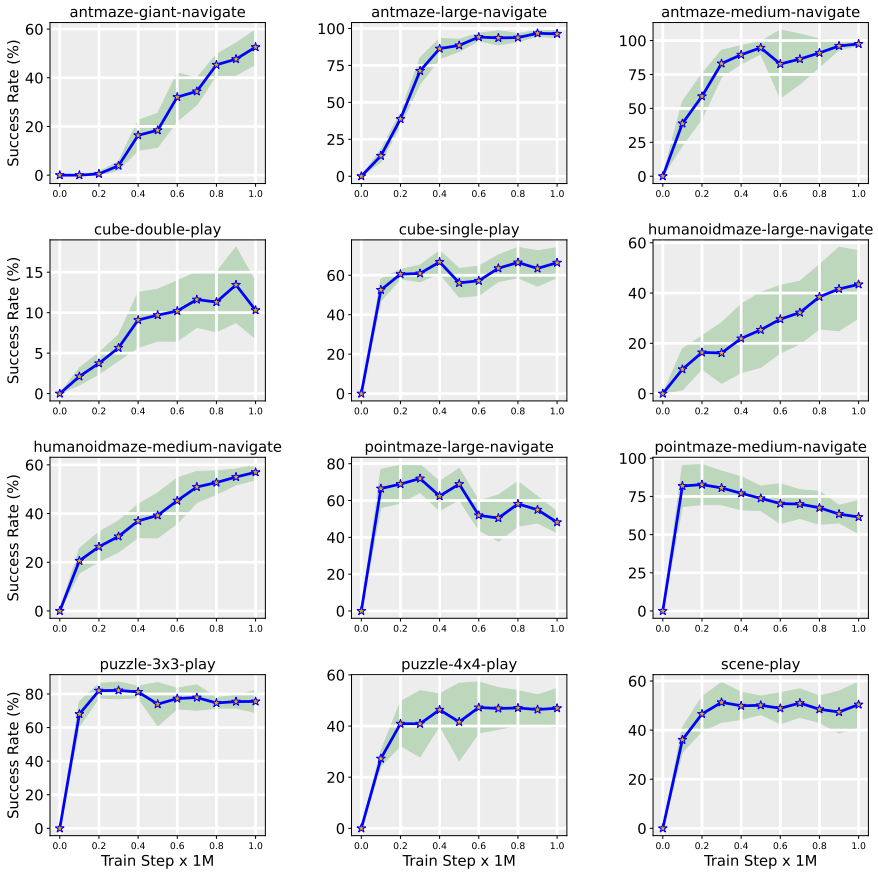

Figure 8: Overall Performance aggregating all 10 seeds for the state-based environment

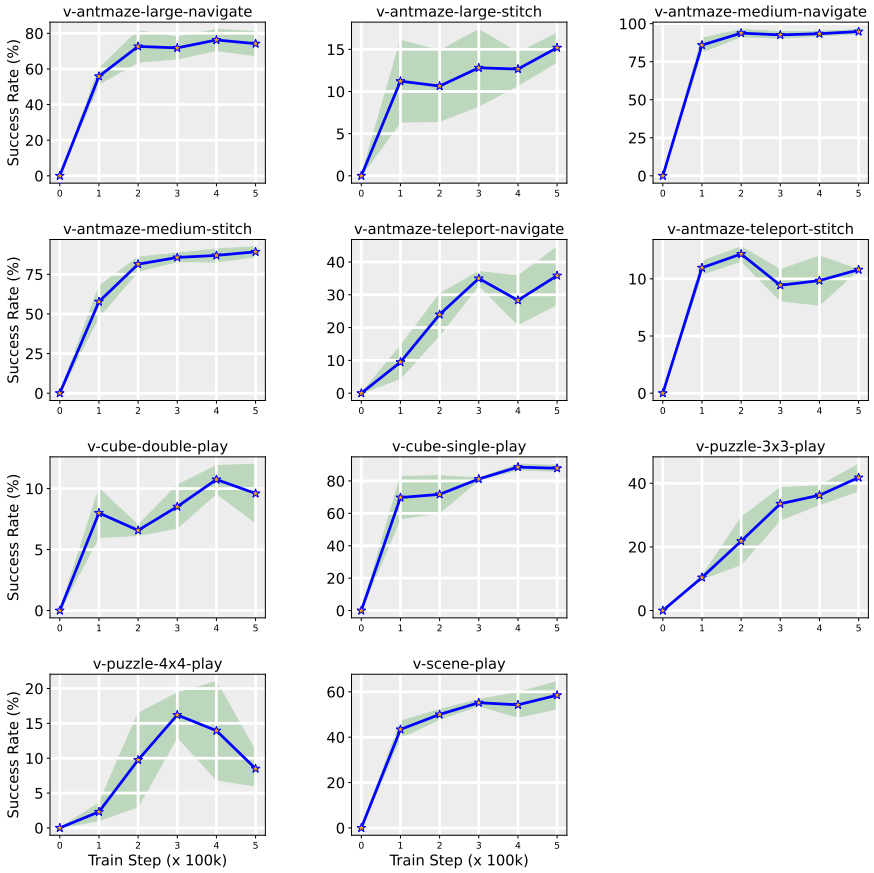

Figure 9: Overall Performance aggregating all 4 seeds for the pixel-based environment

D ADDITIONAL RESULTS

Table 5: Ablation study of Ms.PR and its variants: D (Dynamics only), DI (Dynamics + Inverse Dynamics), DIR (Dynamics + Inverse Dynamics + Reward Model), and the full Ms.PR model. Averaged over 4 random seeds and evaluated using the standard evaluation protocol.

Environment	*D*	*DI*	*DIR*	*Ms.PR*
v-antmaze-medium-navigate	34 ±4	32 ±1	86 ±2	94 ±5
v-antmaze-large-navigate	16 ±2	15 ±1	43 ±2	74 ±4
v-antmaze-teleport-navigate	12 ±1	9 ±1	31 ±5	33 ±5
v-antmaze-medium-stitch	6 ±1	7 ±1	28 ±10	87 ±3
v-antmaze-large-stitch	1 ±0	1 ±0	2 ±1	14 ±0
v-antmaze-teleport-stitch	3 ±1	4 ±1	7 ±4	10 ±3
v-cube-single-play	8 ±1	29 ±6	72 ±7	86 ±8
v-cube-double-play	0 ±0	2 ±1	5 ±1	10 ±1
v-scene-play	26 ±1	28 ±3	32 ±1	56 ±8
v-puzzle-3x3-play	6 ±2	13 ±2	14 ±3	37 ±0
v-puzzle-4x4-play	4 ±3	8 ±2	4 ±1	13 ±0
Average	11 ±2	14 ±1	29 ±2	47 ±2

Table 6: **Results on state-based tasks** We evaluate each goal representation technique VIB, VIP, TRA, BYOL-γ, and Dual using its strongest associated downstream algorithm among GCIVL, CRL, and GCFBC, and compare them against Ms.PR on state-based tasks.

Environment	VIB	VIP	TRA	BYOL-γ	Dual	**OURS**
pointmaze-medium-navigate	69 ±13	55 ±7	77 ±4	77 ±2	76 ±7	64 ±9
pointmaze-large-navigate	69 ±2	65 ±5	76 ±4	75 ±4	66 ±6	54 ±9
antmaze-medium-navigate	68 ±4	94 ±0	77 ±5	87 ±3	93 ±3	95 ±6
antmaze-large-navigate	9 ±3	64 ±10	71 ±14	77 ±4	87 ±2	95 ±3
antmaze-giant-navigate	0 ±0	4 ±1	3 ±1	2 ±1	31 ±4	48 ±6
humanoidmaze-medium-navigate	24 ±2	43 ±1	57 ±1	46 ±1	57 ±4	55 ±4
humanoidmaze-large-navigate	3 ±1	13 ±3	31 ±3	22 ±7	18 ±4	41 ±14
cube-single-play	90 ±3	42 ±6	40 ±5	51 ±11	89 ±3	65 ±8
cube-double-play	33 ±3	4 ±1	7 ±2	6 ±4	60 ±4	12 ±4
scene-play	58 ±1	23 ±6	46 ±6	44 ±9	72 ±6	49 ±8
puzzle-3x3-play	14 ±3	3 ±1	5 ±1	2 ±1	5 ±1	75 ±5
puzzle-4x4-play	6 ±3	1 ±1	10 ±3	1 ±2	23 ±3	47 ±7
Average	37 ±4	34 ±5	42 ±5	41 ±5	56 ±4	58 ±8

E Experimental Analysis: True Value vs Predicted Value

To rigorously assess the fidelity of the agent's value estimation across the state space, we performed a dense empirical evaluation on the `antmaze-large-navigate` task. We sampled 10,000 distinct start configurations covering the entire maze layout. For each start state, we executed N=20 independent Monte Carlo rollouts using the frozen policy to calculate the ground-truth expected return and the empirical success rate. We then compared these ground-truth values against the agent's internal critic estimate (Q-value) at the corresponding states. This dense sampling allowed us to construct spatial heatmaps for the *Predicted Value*, *True Value*, and *Value Estimation Error*, visually isolating regions where the agent's confidence aligns with reality versus areas where the critic overestimates or underestimates the true return.

Figure 10: Comparison of trajectory lengths from varying start locations to the goal. Trajectory length serves as a proxy for accumulated reward.

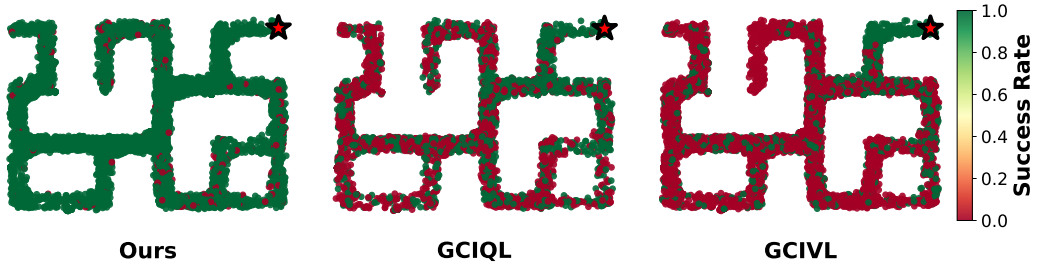

Figure 11: Comparison of success rates across varying start locations. This analysis identifies specific regions where the agent lacks navigational capability.

Figure 12: True Value estimate using Monte-Carlo trials

Figure 13: Predicted Q or V from different states.

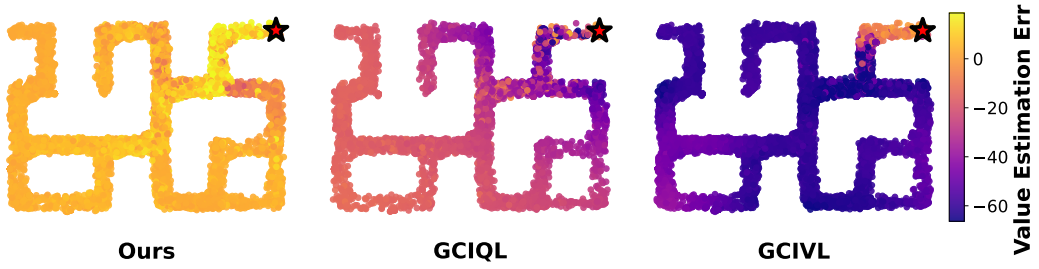

Figure 14: Estimated Value error for different agents.