PcLast: Discovering Plannable Continuous Latent States

Anurag Koul [*1] **Shivakanth Sujit** [*2 3 4] **Shaoru Chen** [1] **Ben Evans** [5] **Lili Wu** [1] **Byron Xu** [1] **Rajan Chari** [1] **Riashat Islam** [3 6] **Raihan Seraj** [3 6] **Yonathan Efroni** [7] **Lekan Molu** [1] **Miro Dudik** [1] **John Langford** [1] **Alex Lamb** [1]

Abstract

Goal-conditioned planning benefits from learned low-dimensional representations of rich observations. While compact latent representations typically learned from variational autoencoders or inverse dynamics enable goal-conditioned decision making, they ignore state reachability, hampering their performance. In this paper, we learn a representation that associates reachable states together for effective planning and goal-conditioned policy learning. We first learn a latent representation with multi-step inverse dynamics (to remove distracting information), and then transform this representation to associate reachable states together in ℓ_2 space. Our proposals are rigorously tested in various simulation testbeds. Numerical results in reward-based settings show significant improvements in sampling efficiency. Further, in reward-free settings this approach yields layered state abstractions that enable computationally efficient hierarchical planning for reaching ad hoc goals with zero additional samples.

1. Introduction

Deep reinforcement learning (RL) has emerged as a choice tool in mapping rich and complex perceptual information to compact low-dimensional representations for onward (motor) control in virtual environments (Silver et al., 2016), software simulations (Brockman et al., 2016), and hardware-in-the-loop tests (Finn & Levine, 2017). Its impact traverses diverse disciplines spanning games (Moravčík et al., 2017; Brown & Sandholm, 2018), virtual control (Tunyasuvunakool et al., 2020), healthcare (Johnson et al., 2016), and autonomous driving (Maddern et al., 2017; Yu et al., 2018). Fundamental catalysts that have spurred these advancements include progress in algorithmic innovations (Mnih et al., 2013; Hessel et al., 2017; Schrittwieser et al., 2020) and learned (compact) latent representations (Bellemare et al., 2019; Lyle et al., 2021; Lan et al., 2022; Rueckert et al., 2023; Lan et al., 2023).

Latent representations, typically learned by variational autoencoders (Kingma & Welling, 2013) or inverse dynamics (Paster et al., 2020; Wu et al., 2023), are mappings from high-dimensional observation spaces to a reduced space of essential information where extraneous perceptual information has already been discarded. Good compact representations foster sample efficiency in learning-based control settings (Ha & Schmidhuber, 2018; Lamb et al., 2022). Latent representations however often fail to correctly model the underlying states' affordances. Consider an agent in the 2D maze of Figure 1a. If we learn a typical representation (such as ACRO of Islam et al., 2022) and cluster it, we observe it correctly identifies the agent's (low-level) position information as indicated by states with nearby coordinate position falling in the same clusters. However, it ignores the scene geometry such as the wall barriers so that states naturally demarcated by obstacles are clustered together (see in Figure 1b). This inadequacy in creating poor abstractions is a drag on the efficacy of planning and deep RL algorithms despite their impressive showings in the last few years.

In this paper, we present two contributions. *First*, we de-

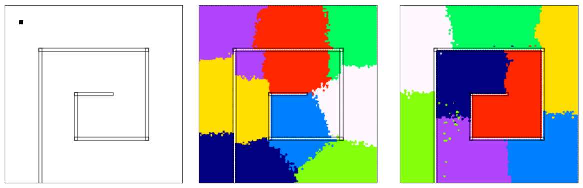

(a) Maze-Spiral (b) ACRO (c) PCLAST

Figure 1. Comparative view of clustering representations learned for a 2D maze environment with spiral walls (a). The agent's location is marked by black-dot in the maze image. The clustering of representations learned via ACRO (b) and PCLAST (c) are overlaid on the maze image.

[*]Equal contribution [1]Microsoft Research [2]Work done as Intern at Microsoft, NYC [3]Mila - Quebec AI Institute [4]ETS Montreal [5]New York University [6]McGill University [7]Meta. Correspondence to: Anurag Koul <anuragkoul@microsoft.com>, Shivakanth Sujit <shivakanth.sujit.1@ens.etsmtl.ca>, Alex Lamb <lambalex@microsoft.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

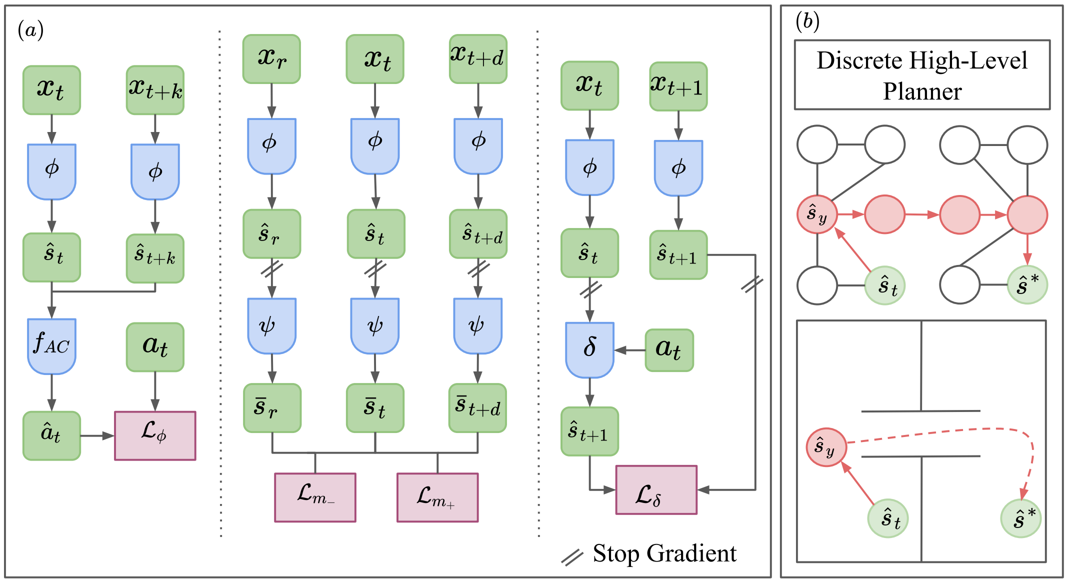


Figure 2. (a) Overview of the proposed method: (a) The encoder ϕ, which maps observations x to continuous latent states $\hat{s}$, is learned with a multi-step inverse model f_{AC} (left). A temporal contrastive objective ($\mathcal{L}_{m_-}$ and $\mathcal{L}_{m_+}$) is used to learn a metric space $\bar{s}$ (middle), a forward model (δ) is learned in the latent space $\hat{s}$ (right). (b) High-level and low-level planners. The high-level planner generates coarse goals ($\hat{s}_y$) to be used as targets for low-level continuous planner. The dashed line indicates the expected trajectory after $\hat{s}_y$ is reached.

velop latent representations that accurately abstract state reachability in the quest towards sample-efficient planning from rich observations. We call this new approach *Plannable Continuous Latent States* or PCLAST which is a map from observations to latent representation and associates neighboring states together by optimizing a contrastive objective inspired by the likelihood function of a Gaussian random walk. The Gaussian is a reasonable model for random exploration *in the embedding space*. Figure 2 shows an overview of our approach, with a specific choice of the initial latent representation based on inverse dynamics.

We hypothesize that PCLAST representations are better aligned with the reachability structure of the environment. Our experiments validate that these representations improve the performance of reward-based policy learning and reward-free task completion schemes. One key benefit of this representation is that it can be used to construct a discretized model of the environment and enable model-based planning to reach an arbitrary state from another arbitrary state. A discretized model (in combination with a simple local continuous planner) can also be used to solve more complex planning tasks that may require combinatorial solvers, like planning a tour across several states in the environment. Similarly to other latent state learning approaches, the learned representations can be used to drive more effective exploration of new states (Machado et al., 2017; Hazan et al., 2019; Jinnai et al., 2019; Amin et al., 2021).

Secondly, since the distance in the PCLAST representation corresponds to the number of transitions between states, discretizing states at different levels of granularity gives rise to different levels of state abstraction. We hypothesize these abstractions can be efficiently used for hierarchical planning. This is validated in our experiments [1] where we show using multiple levels of hierarchy leads to substantial speed-ups in plan computation.

2. Related Work

Our work relates to challenges in representation learning for forward/inverse latent-dynamics and using it for ad-hoc goal conditioned planning. We next discuss each of these aspects.

Representation Learning. Learning representations can be decomposed into *reward-based* and *reward-free* approaches. The former involves both model-free and model-based methods. Model-free methods (e.g., Mnih et al., 2013) directly learn a policy with rich observation as input. One can consider the penultimate layer as a latent-state representation. Model-based approaches (e.g. Hafner et al., 2019a) learn policy, value, and/or reward functions along with the representation. These end-to-end approaches induce task-bias in the representation which makes them unsuitable for diverse tasks. In *reward-free* approaches, the representation is learned in isolation from the task. This includes model-based approaches (Ha & Schmidhuber, 2018), which learn a

[1] Code for reproducing our experimental results can be found at https://github.com/shivakanthsujit/pclast

low-dimensional auto-encoded latent-representation. To robustify, contrastive methods (Laskin et al., 2020) learn representations that are similar across positive example pairs, while being different across negative example pairs. They still retain exogenous noise requiring greater sample and representational complexity. This noise can be removed from latent state by methods like ACRO (Islam et al., 2022) which learns inverse dynamics (Mhammedi et al., 2023). These *reward-free* representations tend to generalize better for various tasks in the environment. The prime focus of discussed reward-based/free approaches is learning a representation robust to observational/distractor noise; whereas not much attention is paid to enforce the geometry of the state-space. Existing approaches hope that such geometry would emerge as a result of end-to-end training. We hypothesize lack of this geometry affects sample efficiency of learning methods. Temporal contrastive methods (such as HOMER (Misra et al., 2020) and DRIML (Mazoure et al., 2020)) attempt to address this by learning representations that discriminate among adjacent observations during rollouts, and pairs random observations (Wang & Gupta, 2015; Nair et al., 2022). However, this is still not invariant to exogenous information (Efroni et al., 2021).

Planning. Gradient descent methods abound for planning in learned latent states. For example, UPN (Srinivas et al., 2018) applies gradient descent for planning. For continuous latent states and actions, the cross-entropy method (CEM) (Rubinstein, 1999), has been widely used as a trajectory optimizer in model-based RL and robotics (Finn & Levine, 2017; Wang & Ba, 2019; Hafner et al., 2019b). Variants of CEM have been proposed to improve sample efficiency by adapting the sampling distribution of Pinneri et al. (2021) and integrating gradient descent methods (Bharadhwaj et al., 2020). Here, trajectory optimizers are recursively called in an online setting using an updated observation. This conforms with model predictive control (MPC) (Mattingley et al., 2011). However, it s limited by planning horizon and returns suboptimal plans in complex tasks. In our work, we ease planning by generating multi-level representation hierarchy and adopting multi-level planner that uses Dijkstra's graph-search algorithm (Dijkstra, 1959) for coarse planning in each hierarchy level for subgoal generation. A low-level planner (such as CEM) along with a learned latent world model is used to search action sequences to reach next subgoal.

Goal Conditioned Reinforcement Learning (GCRL). In GCRL, the goal is specified along with the current state and the objective is to reach the goal in least number of steps. Several efforts have been made to learn GCRL policies (Kaelbling, 1993; Andrychowicz et al., 2017; Nasiriany et al., 2019; Fang et al., 2018; Nair et al., 2018). Further, reward-free goal-conditioned latent-state planning requires estimating the distance between the current and goal latent state, generally using Euclidean norm (ℓ_2) for the same. However, it is not clear whether the learned representation is suitable for ℓ_2 norm and may lead to infeasible/non-optimal plans; even if one has access to true state. So, either one learns a new distance metric (Tian et al., 2020; Mezghani et al., 2023; Wang et al., 2023) which is suitable for the learned representation or learns a representation suitable for the ℓ_2 norm. In our work, we focus on the latter. Further, GCRL reactive policies often suffer over long-horizon problems which is why we use hierarchical planning on learned latent state abstractions as discussed earlier.

3. PCLaSt: Discovery, Representation, and Planning

In this section, we discuss learning the PCLaSt representation, constructing a transition model, and implementing a hierarchical planning scheme.

3.1. Notations and Preliminaries.

In our work, we extend Exogenous Block Markov Decision Process (EX-BMDP) (Efroni et al., 2021) with continuous state and action spaces. We begin by introducing BMDP (Du et al., 2019) and then discuss its extension with exogenous (EX) noise. In our discussion, indices of time like t, t_0, τ will always be integers and $\tau \gg t > t_0$. The Euclidean norm of a matrix X is denoted as $\|X\|$.

BMDP. A BMDP is a tuple $(\mathcal{X}, \mathcal{Z}, \mathcal{A}, T, q, R, \mu)$. Here, $\mathcal{X}, \mathcal{Z}$, and $\mathcal{A}$ are the spaces of observations, latent states, and actions, respectively. The transition function (T) is defined over the latent states as $T\colon \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$. Observations are sampled using the emission function $q\colon \mathcal{Z} \to \mathcal{X}$, with initial latent state distribution given by $z_0 \sim \mu(\cdot)$. The reward is given as $R\colon \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. In contrast to MDPs, BMDP requires *block assumption* (Du et al., 2019), i.e., emission distribution of any two latent states is disjoint.

EX-BMDP. An EX-BMDP is a BMDP whose latent states can be decoupled into two parts $z = (s, \xi)$, where $s \in \mathcal{S}$ is an endogenous state and $\xi \in \Xi$ is the exogenous state. Further, the transition function and initial distribution can be decoupled as $T(z'|z, a) = T(s'|s, a)T_\xi(\xi'|\xi)$ and $\mu(z) = \mu(s)\mu_\xi(\xi)$, respectively.

3.2. ACRO: Learning Endogenous State

We learn endogenous state representation ($\hat{s}$) using an encoder ϕ and an action-controllable (AC) multi-step inverse dynamics f_{AC} as done in ACRO (Islam et al., 2022). The encoder $\phi : \mathcal{X} \to \mathcal{S}$ maps high-dimensional images to low-dimensional representation and the inverse dynamics model $f_{AC} : \mathcal{S} \times \mathcal{S} \times [K_{max}] \to \mathcal{A}$ predicts the likelihood of the next action (a_t) between a pair of states separated by k steps, i.e., $\mathbb{P}(a_t|\phi(x_t), \phi(x_{t+k}))$ (we assume that this conditional distribution is Gaussian with a fixed variance). The functions ϕ and f_{AC} are optimized together using Equations (1a) and (1b) below, where $t \sim U(1, \mathcal{T})$ is the index of time, and $k \sim U(1, K_{max})$ is the amount of look-ahead steps with K_{max} as the diameter of the control-endogenous MDP (Lamb et al., 2022; Islam et al., 2022):

$$\begin{aligned}\mathcal{L}_s(\phi, f_{\text{AC}}, x_t, a_t, x_{t+k}, k) \\ = \|a_t - f_{\text{AC}}(\phi(x_t), \phi(x_{t+k}); k)\|^2, \quad (1a)\end{aligned}$$

$$\arg\min_{\phi} \min_{f_{\text{AC}}} \mathop{\mathbb{E}}_{t} \mathop{\mathbb{E}}_{k} \mathcal{L}_s(\phi, f_{\text{AC}}, x_t, a_t, x_{t+k}, k) \quad (1b)$$

3.3. Learning the PCLAST map

While the encoder ϕ and the inverse dynamics model f_{AC} are designed to filter out the exogenous noise, they do not lead to representations that reflect the reachability structure (see Figure 1b). To enforce states' reachability, we learn a map $\psi : S \to \bar{S}$, which associates nearby states ($s \in S$) to have similar representation in $\bar{s} \in \bar{S}$, based on transition deviations.

Learning ψ is inspired by local random exploration that enforces a Gaussian random walk in the embedding space ($\bar{S}$). This allows states visited in fewer transitions to be closer to each other. A Gaussian random walk with variance σI (where I is an identity matrix) for k steps in $\bar{S}$ would induce a conditional distribution $\mathbb{P}(\bar{s}_{t+k}|\bar{s}_t) \propto \exp\left\{-\frac{\|\bar{s}_{t+k}-\bar{s}_t\|^2}{2k\sigma^2}\right\}$. *This also requires a reversibility assumption that there exists a k-step path from s_{t+k} to s_t.* Instead of fitting ψ to this likelihood directly, we fit a contrastive version, based on the following process for generating triples $\langle y, \bar{s}_t, \bar{s}_{t+k}\rangle$. First, we flip a random coin with outcome $y \in \{0, 1\}$ and then predict y using $\bar{s}_t$ and $\bar{s}_{t+k}$, yielding the likelihood function

$$\mathbb{P}_k(y = 1|\bar{s}_t, \bar{s}_{t+k}) = \sigma(e^{\alpha} - e^{\beta}\|\bar{s}_t - \bar{s}_{t+k}\|), \quad (2)$$

for suitable α and β (see the derivation in Appendix C). We use e^{α} and e^{β} to smoothly enforce positive values.

We employ a contrastive learning loss $\mathcal{L}_{\psi}$ in Equations (3a) to (3d) to fit ψ as well as the parameters α and β by averaging over the expected loss. In $\mathcal{L}_{\psi}$, $t \sim U(1, \mathcal{T})$, $r \sim U(1, \mathcal{T})$, $d \sim U(1, d_m)$ for a hyperparameter d_m. Positive examples (x_t and x_{t+d}) are drawn for the contrastive objective uniformly over d_m steps and optimized using Equation (3a). Negative examples (x_r) are sampled uniformly from a data buffer and optimized with Equation (3b), which encourages change in state representation to be higher for negative samples.

$$\begin{aligned}&\mathcal{L}_{m_+}(\psi, \hat{s}_A, \hat{s}_B, \alpha, \beta) \\ &\quad = -\log(\sigma(e^{\alpha} - e^{\beta}\|\psi(\hat{s}_A) - \psi(\hat{s}_B)\|^2)) \quad (3a)\\ &\mathcal{L}_{m_-}(\psi, \hat{s}_A, \hat{s}_B, \alpha, \beta) \\ &\quad = -\log(1 - \sigma(e^{\alpha} - e^{\beta}\|\psi(\hat{s}_A) - \psi(\hat{s}_B)\|^2)) \quad (3b)\end{aligned}$$

$$\begin{aligned}&\mathcal{L}_{\psi}(\psi, \phi, \alpha, \beta, x_t, x_{t+d}, x_r) \\ &\quad = \mathcal{L}_{m_+}(\psi, \phi(x_t), \phi(x_{t+d}), \alpha, \beta) \\ &\qquad + \mathcal{L}_{m_-}(\psi, \phi(x_t), \phi(x_r), \alpha, \beta) \quad (3c)\end{aligned}$$

$$\arg\min_{\substack{\psi\in\Psi,\\ \alpha,\beta\in\mathbb{R}}} \mathop{\mathbb{E}}_{t,r} \mathop{\mathbb{E}}_{d} \mathcal{L}_{\psi}(\psi, \phi, \alpha, \beta, x_t, x_{t+d}, x_r) \quad (3d)$$

In our approach, ϕ is not optimized with respect to the contrastive loss $\mathcal{L}_{\psi}$. This is motivated by the work of Efroni et al. (2022), who proved that temporal contrastive objective loss can be reduced by capturing exogenous noise. Hence, optimizing ϕ with contrastive loss may lead it to acquire information on the exogenous state. To avoid this failure case, we task ϕ with capturing the agent-centric state and task ψ with learning the local neighborhood structure.

3.4. Learning a latent forward model

In order to plan in our learned latent space, we learn a one-step latent dynamics $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. This estimates observational dynamics by predicting $\phi(x_{t+1}) \approx \delta(\phi(x_t), a_t)$. The forward model δ is parameterized as a fully-connected network of a parameterized family $\mathcal{F}$, optimized with the following objective:

$$\mathcal{L}_{\delta}(\delta, x_t, a_t, x_{t+1}) = \|\phi(x_{t+1}) - \delta(\phi(x_t), a_t)\|^2, \quad (4a)$$

$$\arg\min_{\delta\in\mathcal{F}} \mathop{\mathbb{E}}_{t} \mathcal{L}_{\delta}(\delta, x_t, a_t, x_{t+1}) \quad (4b)$$

In our approach, we jointly optimize all of our architecture components $\phi(\cdot)$, $f_{AC}(\cdot)$, $\psi(\cdot)$, and $\delta(\cdot, \cdot)$.

3.5. Planning

We describe utility of PCLAST for abstraction and generating goal-conditioned abstract plans.

High-Level Planner. Let $\hat{s}_t = \phi(x_t)$ denote the latent state. In the planning problem, we aim to navigate the agent from an initial latent state s_{init} to a target latent state s_{goal} following the latent forward dynamics $\hat{s}_{t+1} = \delta(\hat{s}_t, a_t)$. Since δ is highly nonlinear, it presents challenges for use in global planning tasks. Therefore, we posit that a hierarchical planning scheme with abstractions can improve the performance and efficacy of planning by providing waypoints for the agent to track using global information of the environment.

To find a waypoint $\hat{s}^*$ in the latent space, we first divide the latent space into C clusters by applying k-means to an offline dataset D_ψ, which is created by PCLaSt transformation of offline observations, i.e., $D_\psi = \{\ldots, \psi(\phi(x)), \ldots\}$. We use $\{c_i\}_{i=1}^{C}$ to denote each cluster. Our assumption of reversibility described in Section 3.3 ensures a path exists between any two states of the same cluster. For each cluster, we store the latent-state representation of its centroid alongside the PCLaSt transformation of this latent state.

An abstraction of the environment is given by a graph $\mathcal{G}$ with nodes $\{c_i\}_{i=1}^{C}$ and edges defined by the reachability of each cluster, i.e., an edge from node c_i to node c_j is added to the graph if there are transitions of latent states from cluster c_i to cluster c_j in the offline transition dataset. On the graph $\mathcal{G}$, we apply Dijkstra's shortest path algorithm (Dijkstra, 1959) to find the next cluster the agent should go to and choose the latent state corresponding to centroid of that cluster as the waypoint $\hat{s}^*$. This waypoint is passed to a low-level planner to compute the action.

Low-Level Planner. Given the current latent state $\hat{s}_0$ and the waypoint $\hat{s}^*$ to track, the low-level planner finds the action to take by solving a trajectory optimization problem using the cross-entropy method (CEM) (De Boer et al., 2005). The details are shown in Appendix E.2.

n-Level Planner. To improve the efficiency of finding the waypoint $\hat{s}^*$, we propose to build a hierarchical abstraction of the environment such that the high-level planner can be applied at different levels of granularity, leading to an overall search time reduction of Dijkstra's shortest path algorithm. The *n-Level Planner* creates n abstraction levels, indexed by $i = 1, \ldots, n$, from finest to coarsest, with $i = 1$ corresponding to low-level planner.[2] At level $i \geq 2$, we partition the latent space into C_i clusters using k-means, and we have $C_2 > C_3 > \cdots > C_n$. For each abstraction level, we construct the discrete transition graph $\mathcal{G}_i$ accordingly, which is used to search for the waypoint $\hat{s}^*$ with increasing granularity as shown in Algorithm 1. This procedure guarantees that the start and end nodes are always a small number of hops away in each call of Dijkstra's algorithm. In Section 4.4, our experiments show that multi-level planning leads to a significant speedup compared with using only the finest granularity.

4. Experiments

In this section, we address the following questions via experimentation over environments of different complexities: (1) Does the PCLaSt representation lead to performance gains in reward-based and reward-free goal-conditioned tasks? (2) Does increasing abstraction levels lead to more computationally efficient and better plans? (3) What is the effect of PCLaSt map on abstraction?

[2]When $n = 1$, we only apply the low-level planner without searching for any waypoint.

Algorithm 1 n-Level Planner

```
Require:
    Current observation $x_t$
    Goal observation $x_{goal}$
    Planning horizon $H$
    Encoder $\phi(\cdot)$
    PCLaSt map $\psi(\cdot)$
    Latent forward dynamics $\delta(\cdot,\cdot)$
    Multi-Level discrete transition graphs $\{\mathcal{G}_i\}_{i=2}^{n}$
Ensure: Action sequence $\{a_i\}_{i=0}^{H-1}$
1: Compute current continuous latent state $\hat{s}_t = \phi(x_t)$ and
   target latent state $\hat{s}^* = \phi(x_{goal})$.
   {See Appendix E for details of high-level planner and
   low-level planner.}
2: for $i = n, n-1, \ldots, 2$ do
3:     $\hat{s}^*$ = high-level planner($\hat{s}_t$, $\hat{s}^*$, $\mathcal{G}_i$)
       {Update waypoint using a hierarchy of abstraction.}
4: end for
5: $\{a_i\}_{i=0}^{H-1}$ = low-level planner($\hat{s}_t, \hat{s}^*, H, \delta, \psi$)
   {Solve the trajectory optimization problem.}
```

4.1. Environments

We consider three categories of environments for our experiments and discuss them as follows:

Maze2D—Point Mass. We created 2D maze point-mass environments with continuous actions and states. The environments comprise of different wall configurations with the goal of navigating a point mass. The size of the grid is (100×100) and each observation is a 1-channel image of the grid with "0" marking an empty location and "1" marking the ball's coordinate location (x, y). Actions comprise of $(\Delta x, \Delta y)$ and specify the coordinate space change by which the ball should be moved. This action change is bounded by $[-0.2, 0.2]$. There are three different maze variations: Maze-Hallway, Maze-Spiral, and Maze-Rooms whose layouts are shown in Figure 3(a, b and c). Further, we have dense and sparse reward variants for each environment, details of which are given in Appendix D.1. We created an offline dataset of 500K transitions using a random policy for each environment which gives significant coverage of the environment's state-action space.

Robotic Arm. We extended our experiments to the *Sawyer-Reach* environment of Nair et al. (2018) (shown in Fig. 3d). It consists of a 7 DOF robotic arm on a table with an end-effector. The end-effector is constrained to move only along the planar surface of the table. The observation is an 84-by-84 RGB image of the top-down view of the robotic arm and actions are 2-dimensional continuous vectors that control the end-effector coordinate position. The agent is tested on its ability to control the end-effector to reach random goal positions. The goals are

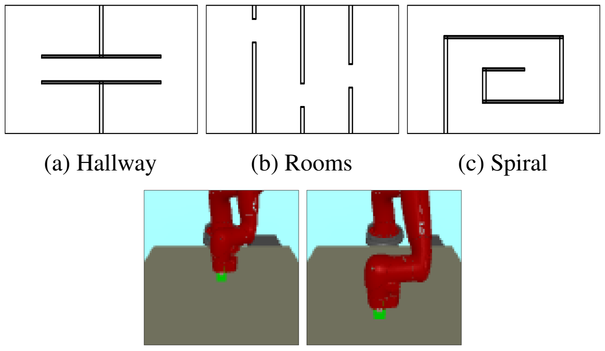


Figure 3. Environments: (a), (b) and (c) show different wall configurations of *Maze2d* environment for point-mass navigation task and (d) shows top-down view of robot-arm environment with the task of reaching various goal positions in 2D-planar space.

given as images of the robot arm in the goal state. Similar to maze2d environment, we generate an offline dataset of 20K transitions using rollouts from a random policy. Likewise to maze, it has dense and sparse reward variants.

Exogenous Noise Mujoco. We adopted control-tasks *"Cheetah-Run"* and *"Walker-walk"* from visual-d4rl (Lu et al., 2022) benchmark which provides offline transition datasets of various qualities. The datasets include high-dimensional agent tracking camera images, to which exogenous noise is added by concatenating randomly sampled images from another distribution as shown in Figure 4 and discussed further in Appendix D.2. We consider *"medium, medium-expert, and expert"* datasets and use *"random"* dataset of same domain as source of exogenous noise. The general objective in these tasks is to keep the agent alive and move forward based on images with exogenous noise.

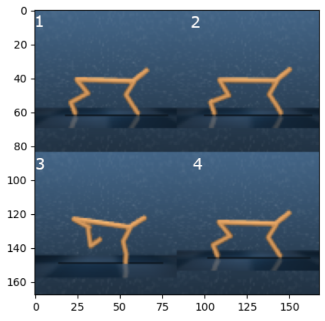


Figure 4. Illustration of an observation for *Cheetah-Run*, where the controllable environment image (1) is placed along with exogenous noise images (2-4) in a 4×4 grid. Numbers on images are for reference only. This grid of images is given as input to agent.

4.2. Impact of representation learning on goal-conditioned RL

We investigate the impact of different representations on performance in goal-conditioned model-free methods. First, we consider methods which use explicit-reward signal for representation learning. As part of this, we trained goal-conditioned variant of PPO (Schulman et al., 2017) on each environment with different current state and goal representation methods. This includes: (1) Image representation for end-to-end learning, (2) ACRO representation (Islam et al., 2022), and (3) PCLAST representation. For (1), we trained PPO for 1 million environment steps. For (2) and (3), we first trained representation using an offline dataset and then used frozen representation with PPO during online training for 100K environment steps only. In the case of *Sawyer-Reach*, we emphasize the effect of limited data and reserved experiments to 20K online environment steps. We also did similar experiment with offline CQL (Kumar et al., 2020) method with pre-collected dataset.

Secondly, we consider RL with Imagined Goals (RIG) (Nair et al., 2018), a method which *doesn't need an explicit reward signal* for representation learning and planning. It is an online algorithm which first collects data with simple exploration policy. Thereafter, it trains an embedding using VAE on images and fine-tunes it over the course of training. Goal-conditioned policy and value functions are trained over the VAE embedding of goal and current state. The reward function is the negative of ℓ_2 distance in the latent representation of current and goal observation. In our experiments, we consider pre-trained ACRO and PCLAST representation in addition to default VAE representation. Pre-training was done over the datasets collected in Section 4.1.

Our results in Table 1 show PPO and CQL have poor performance when using direct images as representations in maze environments. However, ACRO and PCLAST representations improve performance. Specifically, in PPO, PCLAST leads to significantly greater improvement compared to ACRO for maze environments; training curves for the same are shown in Figure 13 (Appendix). This suggests that enforcing a neighborhood constraint facilitates smoother traversal within the latent space, ultimately enhancing goal-conditioned planning. PCLAST in CQL gives significant performance gain for *Maze-Hallway* over ACRO; but they remain within standard error of each other in *Maze-Rooms* and *Maze-Spiral*. Generally, each method does well on *Sawyer-Reach* environment. We assume it is due to lack of obstacles which allows a linear path between any two positions easing representation learning and planning from images itself. In particular, different representations tend to perform slightly better in different methods such as ACRO does better in PPO (sparse), PCLAST does in CQL, and image itself does well in PPO (dense) and RIG.

Table 1. Impact of different representations on policy learning and planning. The numbers represent mean and standard error of the percentage success rate of reaching goal states, estimated over 5 random seeds. RIG and n-Level Planner do not use an external reward signal. In n-Level Planner, we use $n = 5$ abstraction levels. Best mean performance in each task is highlighted in bold.

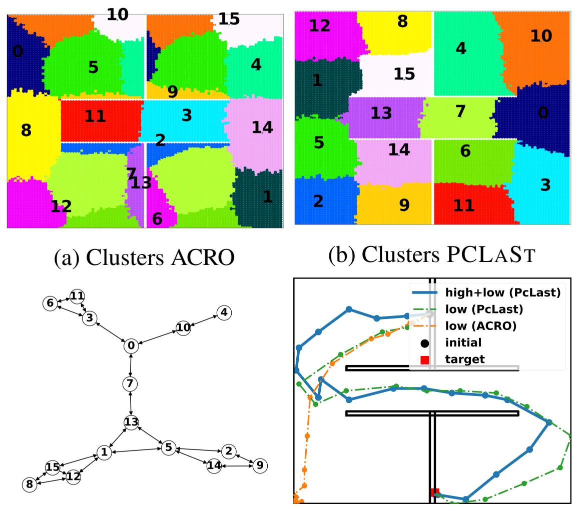
METHOD	REWARD TYPE	HALLWAY	ROOMS	SPIRAL	SAWYER-REACH
PPO	DENSE	6.7 ± 0.6	7.5 ± 7.1	11.2 ± 7.7	**86.00** $\pm$ **5.367**
PPO + ACRO	DENSE	10.0 ± 4.1	23.3 ± 9.4	23.3 ± 11.8	84.00 ± 6.066
PPO + PCLAST	DENSE	**66.7** $\pm$ **18.9**	**43.3** $\pm$ **19.3**	**61.7** $\pm$ **6.2**	78.00 ± 3.347
PPO	SPARSE	1.7 ± 2.4	0.0 ± 0.0	0.0 ± 0.0	68.00 ± 8.198
PPO + ACRO	SPARSE	21.7 ± 8.5	5.0 ± 4.1	11.7 ± 8.5	**92.00** $\pm$ **4.382**
PPO + PCLAST	SPARSE	**50.0** $\pm$ **18.7**	**6.7** $\pm$ **6.2**	**46.7** $\pm$ **26.2**	82.00 ± 5.933
CQL	SPARSE	3.3 ± 4.7	0.0 ± 0.0	0.0 ± 0.0	32.00 ± 5.93
CQL + ACRO	SPARSE	15.0 ± 7.1	**33.3** $\pm$ **12.5**	**21.7** $\pm$ **10.3**	68.00 ± 5.22
CQL + PCLAST	SPARSE	**40.0** $\pm$ **0.5**	23.3 ± 12.5	20.0 ± 8.2	**74.00** $\pm$ **4.56**
RIG	NONE	0.0 ± 0.0	0.0 ± 0.0	3.0 ± 0.2	**100.0** $\pm$ **0.0**
RIG + ACRO	NONE	**15.0** $\pm$ **3.5**	$4.0 \pm 1.$	**12.0** $\pm$ **0.2**	100.0 ± 0.0
RIG + PCLAST	NONE	10.0 ± 0.5	4.0 ± 1.8	10.0 ± 0.1	90.0 ± 5
LOW-LEVEL PLANNER + PCLAST	NONE	86.7 ± 3.4	69.3 ± 3.4	50.0 ± 4.3	$\pm$
n-LEVEL PLANNER + PCLAST	NONE	**97.78** $\pm$ **4.91**	**89.52** $\pm$ **10.21**	**89.11** $\pm$ **10.38**	95.0 ± 1.54

4.3. Impact of PCLAST on state abstraction

We now investigate the quality of learned latent representations by visualizing relationships created by them across true states. This is done qualitatively by clustering the learned representations of observations using k-means. Distance-based planners use this relationship when traversing in latent space. In Figures 5b and 6b, we show clustering of PCLAST representation of offline-observation datasets for *Maze-Hallway* and *Maze-Spiral* environment. We observe clusters having clear separation from the walls. This implies only states which are reachable from each other are clustered together. On the other hand, with ACRO representation in Figures 5a and 6a, we observe disjoint sets of states are categorized as single cluster such as in cluster-10 (orange) and cluster-15 (white) of *Maze-Hallway* environment. Further, in some cases, we have clusters which span across walls such as cluster-14 (light-pink) and cluster-12 (dark-pink) in *Maze-Spiral* environment. These disjoint sets of states violate a planner's state-reachability assumption, leading to infeasible plans.

4.4. Multi-Level Abstraction and Hierarchical Planning

In Section 4.2, we found PCLAST embedding improves goal-conditioned policy learning. However, reactive policies tend to generally have limitations for long-horizon planning. This encourages us to investigate the suitability of PCLAST for n-*level* state abstraction and hierarchical planning with Algorithm 1 which holds promise for long-horizon planning. Abstractions for each level are generated using k-means with varying k over the PCLAST embedding as done in Section 4.3. We do not consider ACRO embedding due to poor clustering behavior shown in Section 4.3.

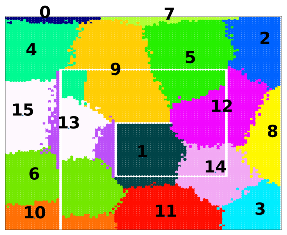


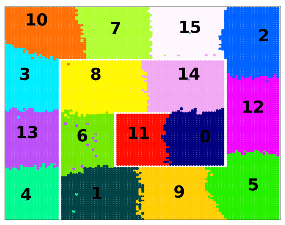
(a) Clusters ACRO (b) Clusters PCLAST

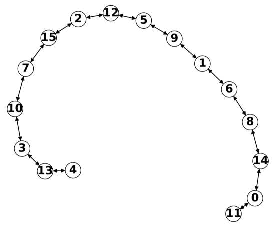
(c) State-transitions PCLAST (d) Planning Trajectories

Figure 5. Clustering, Abstract-MDP, and Planning are shown for *Maze-Hallway* environment. In (a) and (b), we show k-means ($k = 16$) clustering of latent states learned by PCLAST and ACRO, respectively. In (c), we show the abstract transition model of the discrete states learned by PCLAST (b) which captures the environment's topology. Finally, in (d), we show maze configuration and the executed trajectories of the agent from the initial location (black) to the target location (red) using *n-Level* ($n = 2$) planner (blue) with PCLAST and just low-level planner with ACRO (orange) and PCLAST (green) representation for cost minimization .

For simplicity, we begin by considering 2-level abstractions and refer to them as *high* and *low* levels. In Figures 5b and 6b, we show the learned high-level clusters. The corresponding transitions models between the abstract discrete states are shown in Figures 5c and 6c. Note that they match


(a) Clusters ACRO


(b) Clusters PCLAST


(c) State-transitions PCLAST

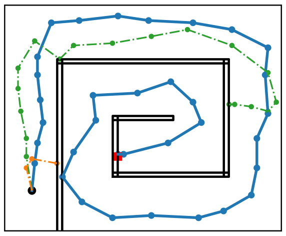

(d) Planning Trajectories

Figure 6. Clustering, Abstract-MDP, and Planning are shown for *Maze-Spiral* environment. Details same as Figure 5.

the true topology of the corresponding mazes. Using this discrete state representation, MPC is applied with the planner implemented in Algorithm 1. The planned trajectories are shown in Figures 5d and 6d and agent's performance is reported in Table 1 (last row). It suggests, *n-Level planner* ($n = 2$) generates feasible and shortest plans (blue line) in all cases. As a baseline, we directly evaluate our *low-level* planner with PCLAST map (ψ) (green line) representation for cost minimization (Equation (5)) which struggles due to long-horizon planning demand of the task and complex navigability of the environment and leads to suboptimal results. At the same time, we observe *low-level* planner with ACRO representation (orange line) for cost minimization fails in all the cases. This is simply due to lack of neighborhood structure in ACRO representation.

Increasing Abstraction Levels. We investigate planning with multiple abstraction levels and consider $n \in \{2, 3, 4, 5\}$. Performance scores for $n = 5$ are reported in Table 1 (bottom row). These abstractions help us create a hierarchy of graphs that describes the environment. In Figure 7, we use $k = \{32, 16, 8, 4\}$ for $n = \{2, 3, 4, 5\}$ abstraction levels, respectively, and show graph-path for each abstraction-level for planning between two locations. This multi-level planning gives a significant boost to planning performance as compared to our model-free baselines. *At the same time, we observe* $3.8\times$ *computational time efficiency improvement in planning with* $n = 5$ *(0.07 ms) as compared to* $n = 2$ *(0.265 ms) abstraction levels*. However, no significant performance gains were observed by increasing levels. We assume this is due to the good quality of temporal abstraction at just $n = 2$ levels which leads to the shortest plans and increasing the levels just helps to save on computation time. However, for more complex tasks, increasing the abstraction levels may further increase the quality of plans.

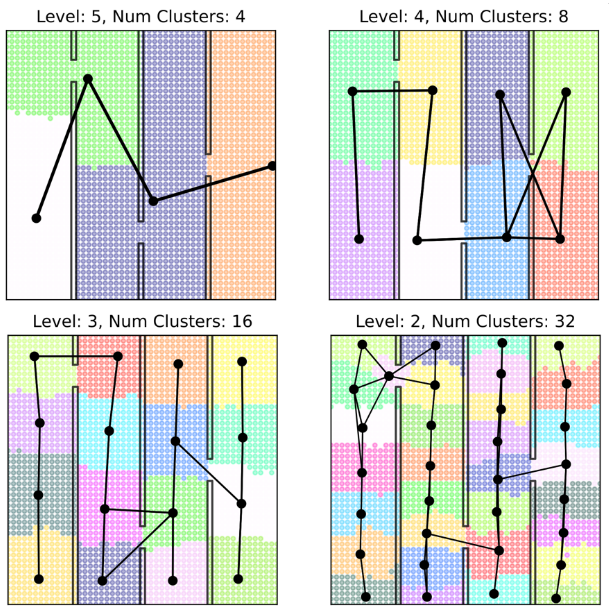


Figure 7. Visualization of hierarchical graphs in the *Maze-Hallway* environment. At every level, num clusters is the k used for clustering. The transition graph (in black) constructed from the cluster centers is superimposed over the environment.

4.5. Exogenous-Noise Offline RL Experiments

Here, we evaluate PCLAST exclusively on exogenous noised control environments described in Section 4.1. We follow the same experiment setup as done by Islam et al. (2022) and consider ACRO (Islam et al., 2022), DRIML (Mazoure et al., 2020), HOMER (Misra et al., 2020), CURL (Laskin et al., 2020) and 1-step inverse model (Pathak et al., 2017) as our baselines. We share results for *"Cheetah-Run"* with *"expert, medium, and medium-expert"* dataset in Figure 8. It shows PCLAST helps gain significant performance over the baselines (Islam et al., 2022). Extended results for *"Walker-Walk"* show similar performance trends as shown in Figure 12 (Appendix). These results along with results in Section 4.2 suggest PCLAST *to be suitable for environments with non-linear dynamics* such as caused by presence of obstacles/walls.

4.6. PCLAST Ablations

In Appendix D.3, we detail our network architecture and hyperparameters. Here, we investigate critical hyperparameters in PCLAST. These include: 1) the impact of K_{max} on multi-step inverse dynamics, 2) cluster sizes in the n-Level planner, and 3) the importance of training a separate map ψ instead of ϕ (ACRO) with contrastive loss (Equation (3d)).

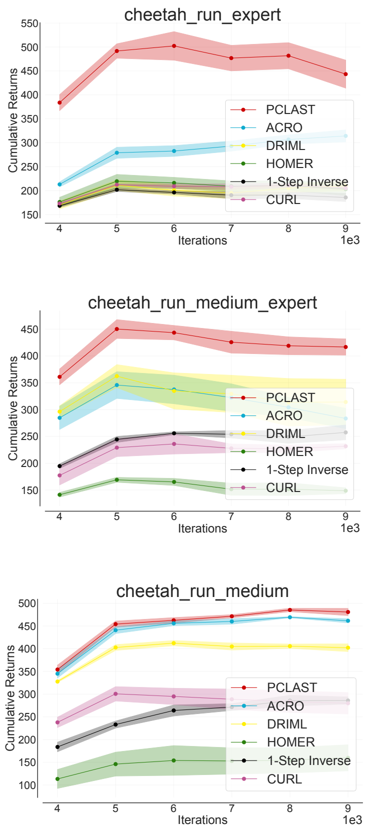


Figure 8. Comparisons of PCLAST in *Cheetah-Run* exogenous-noise environment with several other baselines.

These ablation studies are conducted in *Maze* environments.

K_{max}. In Figure 9, we see that agent performance improves with increasing K_{max} but drops significantly when K_{max} is too large. This is likely due to higher variance in action prediction in multi-step inverse dynamics (f_{AC}) for large values of K_{max}.

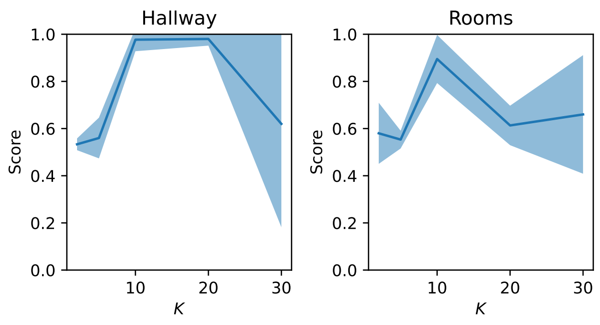


Figure 9. Effect of K_{max} over agents performance in *Maze* environments. The graph shows the normalized mean and standard deviation of scores over 3 seeds.

Cluster-Size (C). In Figure 10, we observe that increasing the number of clusters in n-Level (n=2) planner improves performance, as it makes clusters more robust to errors in the ψ space, reducing planning errors.

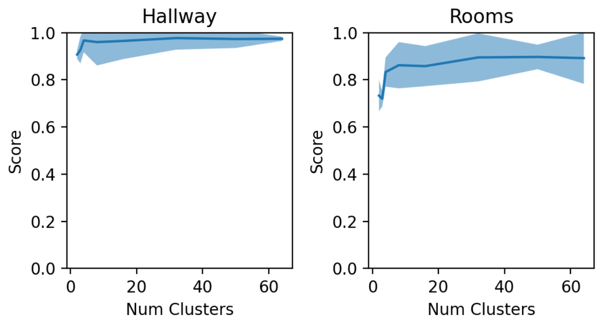


Figure 10. Effect of cluster sizes (C) for abstractions in n-Level planner over agent's performance in *Maze* environments. The graph shows the normalized mean and standard deviation of scores over 3 seeds.

Contrastive Loss. In PCLAST, we learn a representation ψ with the contrastive loss (Equation (3d)). Here, we examine the effect on agent's performance if we use ACRO representation (ϕ) trained via contrastive loss with n-Level planner instead of using ψ. The results are reported in Table 2. As motivated in Section 3.3, PCLAST tends to perform better than just applying contrastive loss on the ACRO (ϕ).

Table 2. Comparing agent's performance when applying contrastive loss to ACRO (ϕ) versus using the PCLAST map ψ. Scores are reported over 3 seeds with $K_{max} = 10$.

ENV.	SCORE	
	ϕ + CONTRASTIVE LOSS	PCLAST
HALLWAY	89.3 ± 8.1	$\mathbf{97.8 \pm 4.9}$
ROOMS	79.3 ± 9.2	$\mathbf{89.6 \pm 10.2}$
SPIRAL	81.3 ± 10	$\mathbf{89.2 \pm 10.3}$

5. Summary

Learning competent agents to plan in environments with complex sensory inputs, exogenous noise, non-linear dynamics, along with limited sample complexity requires learning compact latent representations which maintain state affordances. Our work introduces an approach that learns a representation via a multi-step inverse model and temporal contrastive loss objective. This makes the representation robust to exogenous noise as well as retains local neighborhood structure. Our diverse experiments suggest the learned representation is better suited for reactive policy learning, latent-space planning as well as multi-level abstraction for computationally efficient hierarchical planning.

Impact Statement

Decision making agents for real-world problems face challenges of rich high-dimensional observations embedded with significant exogenous elements. These noisy rich observations limit the usage of agent-training algorithms. It is addressed either by hand-crafting relevant feature extraction or controlling observable world elements such as in factories; both of which require additional human-engineering/effort and are still erroneous. Neural Networks have also been used to learn compact representations of the world to improve sample efficiency of agent-training algorithms. However, when planning in compact representation space, agents still tend to underperform. Further, agents are generally trained for a specific task and struggle to adapt to unseen similar tasks; which requires more data collection and training, increasing economic cost.

In our work, we build on task-agnostic representation methods which is robust to exogenous noise element of the observations; thereby increasing adaptability of trained agents in more realistic environments as well as reducing human-engineering effort to control the environment. Further, the learned representation maintains navigability relation to underlying true state space; easing the generalization and adaptability of agents to unseen interpolated states. On top of that, our framework enables efficient zero-shot planning for new tasks in the environment by decomposing the world into multiple levels of abstraction. This opens up the door to efficient and feasible solutions of complex problems with long horizon solutions. It also reduces human/computational effort spend in collecting data for new tasks as well as reduces economic/carbon costs for training new agents.

In terms of applications, we believe our approach would benefit any decision making agent which is deployed in real-world and/or requires quick adaptability to new tasks such as last-mile delivery robots, house robots, autonomous driving, and others. Our approach takes a step towards robustifying agents to real-world elements and can potentially help with safer interaction with the real-world.

References

Amin, S., Gomrokchi, M., Satija, H., van Hoof, H., and Precup, D. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Bellemare, M., Dabney, W., Dadashi, R., Ali Taiga, A., Castro, P. S., Le Roux, N., Schuurmans, D., Lattimore, T., and Lyle, C. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Bharadhwaj, H., Xie, K., and Shkurti, F. Model-predictive control via cross-entropy and gradient-based optimization. In *Learning for Dynamics and Control*, pp. 277–286. PMLR, 2020.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Brown, N. and Sandholm, T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959. doi: 10.1007/bf01386390. URL `https://doi.org/10.1007/bf01386390`.

Du, S., Krishnamurthy, A., Jiang, N., Agarwal, A., Dudik, M., and Langford, J. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pp. 1665–1674. PMLR, 2019.

Durrett, R. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Efroni, Y., Misra, D., Krishnamurthy, A., Agarwal, A., and Langford, J. Provably filtering exogenous distractors using multistep inverse dynamics. In *International Conference on Learning Representations*, 2021.

Efroni, Y., Misra, D., Krishnamurthy, A., Agarwal, A., and Langford, J. Provably filtering exogenous distractors using multistep inverse dynamics. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=RQLLzMCefQu`.

Fang, M., Zhou, C., Shi, B., Gong, B., Xu, J., and Zhang, T. Dher: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations*, 2018.

Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning, corr abs/1710.02298. *arXiv preprint arXiv:1710.02298*, 2017.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Islam, R., Tomar, M., Lamb, A., Efroni, Y., Zang, H., Didolkar, A., Misra, D., Li, X., van Seijen, H., Combes, R. T. d., et al. Agent-controller representations: Principled offline rl with rich exogenous information. *arXiv preprint arXiv:2211.00164*, 2022.

Jinnai, Y., Park, J. W., Machado, M. C., and Konidaris, G. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2019.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Kaelbling, L. P. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8. Citeseer, 1993.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Lamb, A., Islam, R., Efroni, Y., Didolkar, A., Misra, D., Foster, D., Molu, L., Chari, R., Krishnamurthy, A., and Langford, J. Guaranteed discovery of controllable latent states with multi-step inverse models. *arXiv preprint arXiv:2207.08229*, 2022.

Lan, C. L., Tu, S., Oberman, A., Agarwal, R., and Bellemare, M. G. On the generalization of representations in reinforcement learning. *arXiv preprint arXiv:2203.00543*, 2022.

Lan, C. L., Tu, S., Rowland, M., Harutyunyan, A., Agarwal, R., Bellemare, M. G., and Dabney, W. Bootstrapped representations in reinforcement learning. *arXiv preprint arXiv:2306.10171*, 2023.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Lu, C., Ball, P. J., Rudner, T. G. J., Parker-Holder, J., Osborne, M. A., and Teh, Y. W. Challenges and opportunities in offline reinforcement learning from visual observations. *CoRR*, abs/2206.04779, 2022. doi: 10.48550/arXiv.2206.04779. URL https://doi.org/10.48550/arXiv.2206.04779.

Lyle, C., Rowland, M., Ostrovski, G., and Dabney, W. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pp. 1–9. PMLR, 2021.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017.

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

Mattingley, J., Wang, Y., and Boyd, S. Receding horizon control. *IEEE Control Systems Magazine*, 31(3):52–65, 2011.

Mazoure, B., Tachet des Combes, R., Doan, T. L., Bachman, P., and Hjelm, R. D. Deep reinforcement and infomax learning. *Advances in Neural Information Processing Systems*, 33:3686–3698, 2020.

Mezghani, L., Sukhbaatar, S., Bojanowski, P., Lazaric, A., and Karteek, A. Learning goal-conditioned policies offline with self-supervised reward shaping. *Conference on Robot Learning*, 2023. doi: 10.48550/arXiv.2301.02099. URL https://arxiv.org/abs/2301.02099v1.

Mhammedi, Z., Foster, D. J., and Rakhlin, A. Representation learning with multi-step inverse kinematics: An efficient and optimal approach to rich-observation rl. *arXiv preprint arXiv:2304.05889*, 2023.

Misra, D., Henaff, M., Krishnamurthy, A., and Langford, J. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pp. 6961–6971. PMLR, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Moravčík, M., Schmid, M., Burch, N., Lisỳ, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation, 2022.

Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems*, 32, 2019.

Paster, K., McIlraith, S. A., and Ba, J. Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419*, 2020.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 2017. URL http://proceedings.mlr.press/v70/pathak17a.html.

Pinneri, C., Sawant, S., Blaes, S., Achterhold, J., Stueckler, J., Rolinek, M., and Martius, G. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021.

Rubinstein, R. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

Rueckert, F. L. et al. Cr-vae: Contrastive regularization on variational autoencoders for preventing posterior collapse. *arXiv preprint arXiv:2309.02968*, 2023.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pp. 4732–4741. PMLR, 2018.

Tian, S., Nair, S., Ebert, F., Dasari, S., Eysenbach, B., Finn, C., and Levine, S. Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*, 2020.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Wang, T. and Ba, J. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

Wang, T., Torralba, A., Isola, P., and Zhang, A. Optimal goal-reaching reinforcement learning via quasimetric learning. *arXiv preprint arXiv:2304.01203*, 2023.

Wang, X. and Gupta, A. Unsupervised learning of visual representations using videos, 2015.

Wu, L., Evans, B., Islam, R., Seraj, R., Efroni, Y., and Lamb, A. Agent-centric state discovery for finite-memory POMDPs. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*, 2023. URL https://openreview.net/forum?id=nVgjCpz5mG.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T., et al. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2(5):6, 2018.

A. Limitations

PCLAST representation and n-Level hierarchical planning enables us to do computationally efficient plans as well as achieves higher success rate in goal conditioned tasks. However, we observe PCLAST is primarily effective in non-linear dynamic environments (such as with obstacles), and conventional methods (such as PPO) tend to have similar performance in linear environments. When training PCLAST, one needs domain knowledge to have good candidate of K_{max} value, which is used for learning multi-step inverse dynamics. Further, when clustering the representation with PCLAST map for the lowest level of hierarchy, one re-needs the domain knowledge to determine good number of clusters. In practice, we examine the environment to determine cluster count such that it leads to near-linear dynamics within the state-action space of a cluster at the lowest level. We also assume the presence of offline dataset with significant coverage of task-relevant state-action space. Though, our method just relies on the coverage and not on the quality of behavior policy; implying even a random policy data is sufficient for our method.

B. Intuitive Argument: Learning Local Neighborhood Structure via Contrastive Learning

In Section 3.3, we introduced a contrastive learning objective which allows us to learn the underlying local neighborhood structure in scalable and differentiable way (see equations (3a)-(3d)). Here, we elaborate on the intuition that resulted in this objective. We show that the newly introduced temporal contrastive loss can be derived assuming a natural diffusion process on the underlying dynamics.

Consider a simple discrete time multi-dimensional Brownian motion in $\bar{S}$ state-space. The conditional probability to observe state $\bar{s}'$ after stepping for k steps from state $\bar{s}$, denoted as $\mathbb{P}_k(\bar{s}'|\bar{s})$, is given as (see Durrett (2019), Section 7):

$$\mathbb{P}_k(\bar{s}'|\bar{s}) \propto \exp\left(-\frac{||\bar{s}-\bar{s}'||^2}{\sigma_0 k}\right).$$

With this fact in mind, we can study the distribution of the contrastive learning process which outputs the tuple $(y, \bar{s}, \bar{s}')$. A simple way to define this process is as follows: (1) sample $\bar{s}$ uniformly from $\bar{S}$, (2) After k-steps, set $\bar{s}'$ by sampling $y \sim \text{Bernoulli}(0.5)$; if $y = 1$, set $\bar{s}' \sim \mathbb{P}_k(\cdot|\bar{s})$ otherwise set $\bar{s}'$ by uniformly sampling from $\bar{S}$. We refer to this process as the *Contrastive Learning(CL) generating process*. The following can be readily derived from these assumptions (see proof in Appendix C).

Proposition 1. *Assume the tuple $(y, \bar{s}, \bar{s}')$ is sampled via the CL generating process described above. Then,* $\mathbb{P}_k(y = 1 \mid \bar{s}, \bar{s}') = \text{sigmoid}(c - b||\bar{s}-\bar{s}'||^2)$, *where* $\text{sigmoid}(x) = \exp(x)/(\exp(x)+1)$.

Although this generating process does not take into account the geometry of the underlying space and subtle intricacies of the environment, for small time scales this process can capture the dynamics to a reasonable degree. Further, the contrastive learning objective follows directly from Proposition 1. This objective can be interpreted as a log-likelihood learning procedure of the CL generating process.

C. Bayes Solution of Contrastive Loss

Proposition 1. *Assume the tuple $(y, \bar{s}, \bar{s}')$ is sampled via the CL generating process described above. Then,* $\mathbb{P}_k(y = 1 \mid \bar{s}, \bar{s}') = \mathrm{sigmoid}(c - b||\bar{s} - \bar{s}'||^2)$, *where* $\mathrm{sigmoid}(x) = \exp(x)/(\exp(x) + 1)$.

Proof. The proof following by direct analysis of the conditional probability distribution together with the assumption of the CL generating process, i.e., the underlying Brownian motion.

$$
\begin{aligned}
\mathbb{P}_k(y=1|\bar{s},\bar{s}') &= \frac{\mathbb{P}_k(\bar{s}|\bar{s}', y=1)\mathbb{P}_k(y=1|\bar{s})}{\mathbb{P}_k(\bar{s}'|\bar{s}, y=1)\mathbb{P}_k(y=1|\bar{s}) + \mathbb{P}_k(\bar{s}'|\bar{s}, y=0)\mathbb{P}_k(y=0|\bar{s})} && \because \text{Bayes' rule} \\
&= \frac{\mathbb{P}_k(\bar{s}|\bar{s}', y=1)}{\mathbb{P}_k(\bar{s}'|\bar{s}, y=1) + \mathbb{P}_k(\bar{s}'|\bar{s}, y=0)} && \because \mathbb{P}_k(y=1|\bar{s}) = \mathbb{P}_k(y=0|\bar{s}) = 0.5; \text{Bernoulli} \\
&= \frac{\mathbb{P}_k(\bar{s}|\bar{s}', y=1)}{\mathbb{P}_k(\bar{s}'|\bar{s}, y=1) + 1/\left|\bar{\mathcal{S}}\right|} && \because \mathbb{P}_k(\bar{s}'|\bar{s}, y=0) \approx 1/\left|\bar{\mathcal{S}}\right| \\
&= \frac{C\exp\left(-\frac{||\bar{s}-\bar{s}'||^2}{\sigma_0 k}\right)}{C\exp\left(\frac{-||\bar{s}-\bar{s}'||^2}{\sigma_0 k}\right) + 1/\left|\bar{\mathcal{S}}\right|} && \because \text{Assuming brownian motion (where } C \text{ is a positive constant)} \\
&= \frac{\exp\left(\log(C\left|\bar{\mathcal{S}}\right|) - \frac{||\bar{s}-\bar{s}'||^2}{\sigma_0 k}\right)}{\exp\left(\log(C\left|\bar{\mathcal{S}}\right|) - \frac{||\bar{s}-\bar{s}'||^2}{\sigma_0 k}\right) + 1} \\
&= \mathrm{sigmoid}(c - b||\bar{s}-\bar{s}'||^2) && \text{where } b = 1/(\sigma_0 k) \text{ and } c = \log(C\left|\bar{\mathcal{S}}\right|)
\end{aligned}
$$

□

D. Experiments

We discuss design of the considered environments as well as implementation details of our work.

D.1. Maze-2D point mass

Environment Setup. The state s_t of the point-mass experiment is the 2D position of a point and the action a_t is the position displacement, i.e., $s_{t+1} = s_t + a_t$. The action is bounded by $\|a_t\|_\infty \leq 0.2$. In the presence of obstacles, the point mass starting from s_t is moved along the direction of a_t until it collides with an obstacle. Further, we have two reward variants for each maze : 1) *Dense-reward* and 2) *Sparse-reward*. In the dense case, the agent receives a reward for the first time it crosses a particular distance threshold from the goal. Specifically, if d_g is the distance to the goal, the agent receives a reward r encouraging it to go closer to the goal. The thresholds for reaching the goal, as well as the corresponding reward values are given below.

$$
\begin{aligned}
r &= 0.25 && \text{if this is the first time } d_g < 0.1 \\
&= 0.5 && \text{if this is the first time } d_g < 0.05 \\
&= 1 && \text{if } d_g < 0.03
\end{aligned}
$$

In the sparse setting, the agent receives a reward of 1 when it's within a distance of 0.03 to the goal state. For evaluation, we randomize the start and goal states from across the maze so as to test the agent's ability to reach diverse goals.

As shown in Figures 5 and 6 in the main paper, we consider three environments with distinct layouts of obstacles. In each of these environments, a dataset of 500K samples is collected using random actions. An instance of the dataset has < *obs-image, state, action, next-state, next-obs-image* >, where *state* and *next-state* are the coordinates of the agent in the maze and represent the true environment state as the obstacles are fixed. We use the transaction data to train the latent

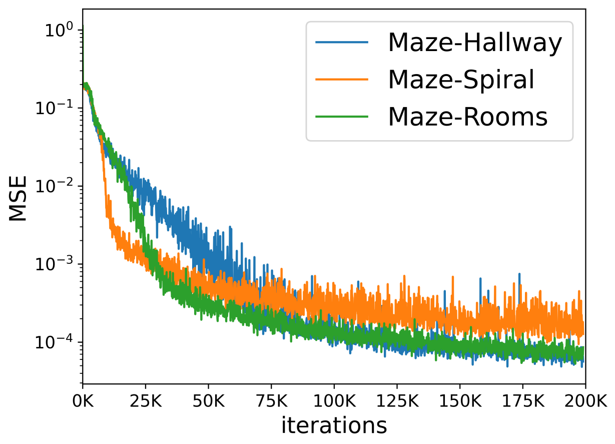


Figure 11. Mean square error of predicting the true state using the learned latent states during the training.

dynamics and extract an abstract transaction model using k-means clustering over the latent states with the continuous actions discretized with a resolution of 0.01 for identifying transitions of the latent states across different clusters.

Latent Representation Quality. Here we demonstrate the quality of the learned latent representation. Since, in the 2D point-mass experiment, we have access to the true state of the environment, we train a feed-forward network to predict the true state from the learned latent state. In Figure 11, we show the regression error of predicting the true state from the latent state. A significant low error implies that the learned latent state captures information about the true state.

Transition Model Generation. In constructing the transition models between the clusters of latent states, we filter out the infrequent transitions to avoid giving hard-to-reach goals to the low-level planner. For the cluster c_i, let $c_{i_1}, c_{i_2}, \cdots, c_{i_N}$ denote the N clusters such that there is at least a state transaction from c_i to c_{i_j} in the collected samples. This means it is feasible to move the point mass from c_i to c_{i_j}. Let $0 < p_{i_j} \leq 1$ for $j = 1, \cdots, N$ denote the ratio of the number of transactions from c_i to c_{i_j} to the total number of outward transactions from c_i. We observe that if p_{i_j} is small, the following issues may happen: (i) The transition from c_i to c_{i_j} is caused by the clustering errors and does not give a feasible transaction of the agent in practice. (ii) Even if such a transition is feasible, it is difficult for the low-level planner to find such a path as indicated by the sparsity of such transitions in the collected dataset. Therefore, in generating the transition models, we add the edge $c_i \to c_{i_j}$ to the graph only when p_{i_j} is large enough. Without loss of generality, assume that p_{i_j} for $j = 1, \cdots, N$ have been arranged in descending order. Motivated by the nucleus sampling (Holtzman et al., 2019), we choose $N^* = \arg\min_k \sum_{j=1}^{k} p_{i_j} \geq 0.9$ and only add the edges $c_i \to c_{i_j}$ for $j = 1, \cdots, N^*$ to the graph. In this way, the spurious transitions are filtered out.

D.2. Exogenous Noise Offline RL Experiments

We add exogenous noise by sampling 3 observations from "random" quality dataset and adding them around the main observation as shown in Figure 4. This creates a 4×4 exogenous observation from the offline dataset. This is same setup as from (Islam et al., 2022). In our experiments, we have the controllable environment in one corner of the grid, and 3 other uncontrollable environments, taken from a random dataset, placed randomly in the 4×4 grid. Figure 12 shows additional results in the offline RL experimental setup.

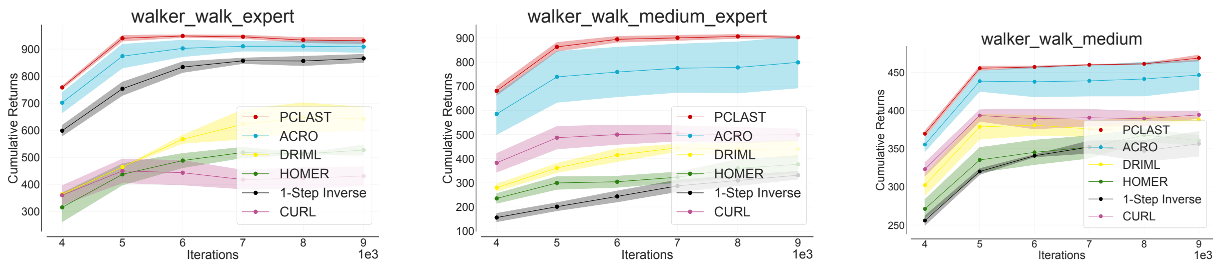


Figure 12. Extended results of PCLAST with several other baselines in *Walker-Walk* task, following the 4×4 exo-grid observation space offline RL setup from (Islam et al., 2022).

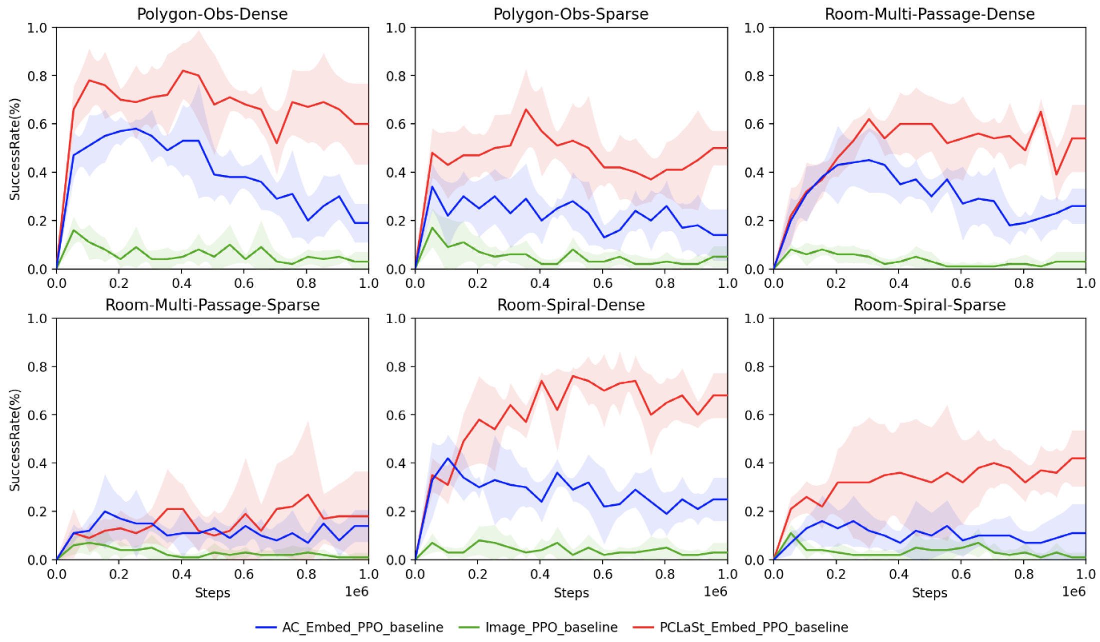


Figure 13. Performance comparison of PPO with ACRO and PCLAST Embedding. The graph shows mean and standard deviation over 3 seeds.

D.3. Network Architecture and Implementation Detail

PCLAST comprises of an encoder (ϕ), multi-step inverse dynamics (f_{AC}), PCLAST map (ψ) and one-step forward dynamics (δ). These networks are trained using Adam (Kingma & Ba, 2014) optimizer with learning rate of $1e^{-3}$ over batches of size 512. Unless specified otherwise, we sample transitions with $K_{max} = 10$. In the following, we discuss each of the networks.

- **Encoder.** It comprises of 2 layer MLP-Mixer Network (Tolstikhin et al., 2021) with an image patch size of 10, followed by a layer of BatchNorm (Ioffe & Szegedy, 2015) and GroupNorm (Wu & He, 2018). The output of GroupNorm is passed through a 2 layer network with intermediate Leaky ReLU activation. We use the same encoder output($\hat{s}$) dimension of 256 for all our experiments.

- **Multi-Step Inverse Dynamics.**. It operates over a batch of $< \hat{s}_t, \hat{s}_{t+k}, k >$. We use an Embedding layer to transform scalar k into an embedding of size 45 and concatenate it with $\hat{s}_t$ and $\hat{s}_{t+k}$ before feeding the network. The inverse dynamics network is a multi-layer network with the first linear layer of size 256. This is followed by a 2-layer residual unit with in-between Gelu activation. Finally, we have two output head layers. The first outputs deterministic continuous actions and is trained with Mean square error loss. The second layer outputs categorical distribution over continuous values by binning the continuous action space and is trained via cross entropy loss. We divide the action-space of considered environments into 20 bins. During backprop, mean-square loss and categorical loss are scaled by 10 and 0.01 respectively.

- **PCLAST map.** It's a 3-layer fully-connected network with Leaky ReLU as intermediate activations and hidden layers of size 512. For training, we sample positive examples using $d_m = 2$, and negative samples are drawn randomly. Gradients from PCLAST map are not propagated back to the encoder.

- **Forward dynamics.** It's 4 layer fully-connected network with 512 as hidden layer size. It outputs a Gaussian distribution over the next state prediction. Again, gradients from forward dynamics are not propagated back to the encoder network.

E. Multi-Level Planner Implementation Details

E.1. High-level planner

Given the current latent state $\hat{s}_t$ and the target latent state $\hat{s}^*$, the high-level planner aims to find an intermediate waypoint $\tilde{\hat{s}}$ such that the low-level planner can effectively track $\tilde{\hat{s}}$. The search for the waypoint is based on the discrete abstraction of the environment which is given by the graph $\mathcal{G}$ as described in Section 3.4. We denote $\phi_d(\cdot)$ as the node (or cluster) membership function for the graph abstraction $\mathcal{G}$ such that $\phi_d(\psi(\hat{s}))$ returns the node in $\mathcal{G}$ for any latent state $\hat{s} \in \mathcal{S}$. The high-level planner is outlined in Algorithm 2

Algorithm 2 High-level planner

```
Require:
    Current latent state $\hat{s}_t$
    target latent state $\hat{s}^*$
    PCLAST map $\psi(.)$
    graph abstraction $\mathcal{G}$
    {The node membership function $\phi_d(\cdot)$ is given by $\mathcal{G}$.}
Ensure: Waypoint $\tilde{\hat{s}}$
 1: Find the discrete latent states $c = \phi_d(\psi(\hat{s}))$ and $c^* = \phi_d(\psi(\hat{s}^*))$.
 2: if $c \neq c^*$ then
 3:     $\tilde{c}$ = Dijkstra($c, c^*, \mathcal{G}$)
        {$\tilde{c}$ is the next node on the shortest path from discrete states $c$ to $c^*$.}
 4:     $\tilde{\hat{s}}$ = center of cluster $\tilde{c}$. { $\tilde{\hat{s}} \in S$}
 5: else
 6:     $\tilde{\hat{s}} = \hat{s}^*, \tilde{c} = c^*$
 7: end if
```

E.2. Low-level planner

Note that the latent forward dynamics is given by $\hat{s}_{t+1} = \delta(\hat{s}_t, a_t)$. Without loss of generality, we consider $\hat{s}_0$ as the current latent state and $\hat{s}^*$ as the target latent state. Given a horizon $T \geq 1$, our low-level planner generates actions $\{a_t\}_{t=0}^{T-1}$ that drive the latent state $\hat{s}_t$ to reach $\hat{s}^*$ by solving a trajectory optimization problem using PCLAST map ψ.

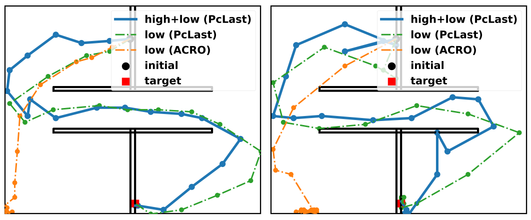
$$\begin{aligned} \underset{a_0,\cdots,a_{T-1}}{\text{minimize}} \quad & \sum_{t=0}^{T} \|\psi(\hat{s}_t) - \psi(\hat{s}^*)\|^2 \\ \text{subject to} \quad & \hat{s}_{t+1} = \delta(\hat{s}_t, a_t),\ t = 0, \cdots, T-1. \end{aligned} \tag{5}$$

In this work, we apply CEM to solve the low-level planning problem (5). CEM has been successfully applied in model-based reinforcement learning (Finn & Levine, 2017; Wang & Ba, 2019; Hafner et al., 2019b) and is outlined in Algorithm Algorithm 3. In the CEM, the actions $\{a_i\}_{i=0}^{T-1}$ are drawn from a multivariate Gaussian distribution whose parameters (mean and covariance matrix) are updated iteratively to approximate the optimal distribution of actions.

Algorithm 3 Cross-Entroy method

```
Require:
    Number of iteration $N$
    number of samples $M$ each iteration
    parameter $K$
Ensure: Action sequence $\{a_i^*\}_{i=0}^{T-1}$
1: Initialize a multivariate Gaussian distribution $\mathcal{N}(\mu^{(0)}, \Sigma^{(0)})$ with mean $\mu^{(0)} = \mathbf{0}$ and covariance matrix $\Sigma^{(0)} = I$.
2: for $j = 0, \cdots, N-1$ do
3:     Sample $M$ action sequences $\{a_i^{(m)}\}_{i=1}^{T-1}$ from $\mathcal{N}(\mu^{(j)}, \Sigma^{(j)})$ for $m = 1, \cdots, M$.
4:     For each action sequence $\{a_i^{(m)}\}_{i=1}^{T-1}$, evaluate the rendered cost $J^{(m)}$ of Problem (5).
5:     Select the $K$ smallest costs from $\{J^{(m)}\}_{m=1}^{M}$ and the corresponding action sequences $\{a_i^{*,(k)}\}_{i=1}^{T-1}$ for $k = 1, \cdots, K$.
6:     Update $\mu^{(j+1)}$ and $\Sigma^{(j+1)}$ as the mean and covariance of $\{a_i^{*,(k)}\}_{i=1}^{T-1}$, $k = 1, \cdots, K$.
7: end for
8: Sample the action sequence $\{a_i^*\}_{i=0}^{T-1}$ from $\mathcal{N}(\mu^{(N)}, \Sigma^{(N)})$ as the output.
```

Low-level planners comparison. In Figure 14, we evaluate the performances of the hierarchical planner and the low-level planner with two different methods, namely CEM and a first-order gradient descent-based method Adam (Kingma & Ba, 2014), to solve problem (5). In both cases, the hierarchical planner reaches the goal while solely using the low-level planner fails. This means directly using the ACRO representation $\hat{s}$ for planning faces significant difficulty. Figure 14 shows that the high-level planner (Algorithm 2) derived on $\psi(\hat{s})$ is a key enabler of successful planning in the latent space and over a long horizon.



(a) Cross-Entropy Method (b) Adam (Kingma & Ba, 2014)

Figure 14. Comparison of the CEM (Left) and Adam (Right) methods in solving the low-level planning problem (5) in the Maze-Hallway environment. 16 clusters are used for the high-level planner. It shows that the inherent complexity of problem (5) poses significant challenges for both low-level planners. A high-level planner shown in Algorithm 2 is a necessary enabler of goal reaching.