STRESS-TESTING OFFLINE REWARD-FREE REINFORCEMENT LEARNING: A CASE FOR PLANNING WITH LATENT DYNAMICS MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) has enabled significant progress in controlling embodied agents. While online RL can learn complex behaviors, it is usually costly and limiting as it requires direct interactions between an agent and its environment. On the other hand, offline RL has promised to use pre-collected data to solve tasks without any direct environment interaction. In particular, zero-shot and goal-conditioned offline RL methods are even able to handle reward-free data. However, how the properties of the offline dataset influence the performance of offline RL for reward-free data remains unclear. In this work, we study how well offline RL methods for reward-free data generalize using controlled offline datasets of varying quality. We find that when given a large amount of high-quality data, model-free approaches excel but that model-based planning achieves superior performance when there is variability in the environment layouts, when solving the task requires stitching suboptimal trajectories, or when the dataset is small. Given the scarcity of high-quality, task-specific data and the abundance of suboptimal, task-agnostic trajectories in real-world scenarios, our results suggest that planning with a dynamics model is an appealing choice for zero-shot generalization from suboptimal data.

1 INTRODUCTION

How can we train agents to generalize to previously unseen tasks and environments? Although online reinforcement learning (RL) enables learning to solve increasingly complex tasks—ranging from Atari games (Mnih, 2013) to Go (Silver et al., 2016) to controlling real robots (OpenAI et al., 2018)—it requires numerous environment interactions to do so. For example, OpenAI et al. (2018) used the equivalent of 100 years of real-time hand manipulation experience to train a robot to reliably handle a Rubik's cube. To address this sample complexity problem, offline RL methods (Kostrikov et al., 2021; Levine et al., 2020; Ernst et al., 2005) have been developed to learn behaviors from state–action trajectories with corresponding reward annotations. Unfortunately, conventional offline RL limits agents to one task, making it impossible to use a trained agent to solve another downstream task. To address this shortcoming, recently proposed methods learn desired behaviors from reward-free data (Park et al., 2024b; Touati & Ollivier, 2021; Kim et al., 2024; Park et al., 2024c). This reward-free paradigm is particularly appealing as it allows agents to learn from suboptimal data and use the learned policy to solve a variety of downstream tasks. For example, a system trained on a large dataset of low-quality robotic interactions with cloths can generalize to new tasks, such as folding laundry (Black et al., 2024).

Despite significant methodological advances, the question of how the quality of pre-training data affects the performance of reward-free offline RL methods remains unanswered. Prior works most commonly train methods on data collected either by an expert policy or using unsupervised RL (Fu et al., 2020; Yarats et al., 2022) and typically do not explore more than three types of datasets per task. In this work, we address this gap in the literature and study the strengths and weaknesses of the offline RL methods for reward-free data. We conduct a wide range of carefully designed experiments to determine how different learning paradigms handle data of varying quality, amount, and relevance.

Our contributions can be summarized as:

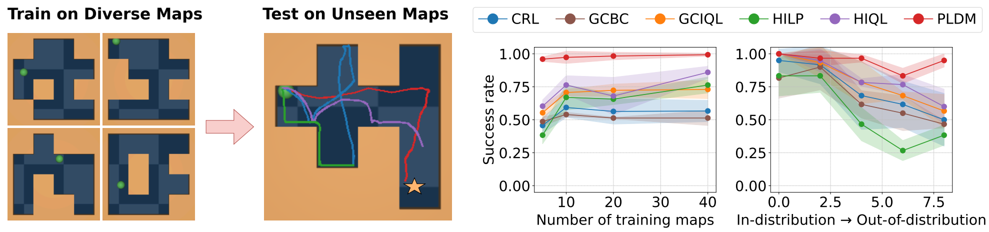


Figure 1: **Left:** Examples of training maze layouts in the offline data, and trajectories of different agents navigating an unseen maze layout towards goal at test time. **Right:** Success rates of various methods on held-out layouts, as a function of the number of training layouts, as well as success rates of models trained on data from five layouts, evaluating on held-out layouts ranging from similar to training layouts to out-of-distribution ones. See Figure 9 for more details.

- We propose two navigation environments with granular control over the data generation process, and generate a total of *23 datasets* of varying quality;
- We use these datasets to carefully investigate the behavior of existing offline RL methods for reward-free data. We test their ability to learn from random trajectories, to stitch short trajectories, to learn from small datasets, and to generalize to unseen environments and tasks;
- We demonstrate that learning a latent dynamics model and using it for planning is the most robust to data quality and achieves the highest level of generalization to environment variations;
- We present a list of guidelines to help practitioners choose between methods depending on available data and generalization requirements;

We release[1] code to construct the environments along with the used data to enable further research into the role of data quality in offline RL with reward-free data.

2 Related Work

Offline RL aims to learn behaviors purely from offline data without online interactions. There, a big challenge is preventing the policy from selecting trajectories that were not seen in the dataset. CQL (Kumar et al., 2020) relies on model conservatism to prevent the learned policy from being overly optimistic about trajectories not observed in the data. IQL (Kostrikov et al., 2021) introduces an objective that avoids evaluating the Q-function on state-action pairs not seen in the data to prevent value overestimation. MOPO (Yu et al., 2020) is a model-based approach to learning from offline data, and uses model disagreement to constrain the policy. See (Levine et al., 2020) for a more in-depth survey.

Reward-free offline RL proposes to learn from the offline data that does not contain rewards in a task-agnostic way. The goal is to extract general behaviors from the offline data to solve a variety of downstream tasks. One approach to this is to use goal-conditioned RL, with and sample goals using a technique proposed in Hindsight Experience Replay (Andrychowicz et al., 2017). Park et al. (2024b) show that this can be applied to learn a goal-conditioned policy using IQL, as well as to learn a hierarchical value function. Hatch et al. (2022) proposes using a small set of observations corresponding to the solved task to define the task and learn from reward-free data. Kim et al. (2024) study how to transition from offline to online RL, and uses HILP (Park et al., 2024c) for unsupervised pre-training, then fine-tunes it on online data. Yu et al. (2022); Hu et al. (2023) propose to use labeled data to train a reward function, than label the reward-free trajectories.

Zero-shot methods go beyond just goal-reaching from offline data, and aim to solve any possible task specified during test time. HILP (Park et al., 2024c) propose learning a distance-preserving representation space such that the distance in that space is proportional to the number of steps between two states, similar to Laplacian representations (Wu et al., 2018; Wang et al., 2021; 2022). Forward-Backward representations (Touati & Ollivier, 2021; Touati et al., 2022) tackle this with an approach akin to successor-features. Frans et al. (2024) propose to learn a transformer model to encode target task's state action sequences. Chen et al. (2023) propose to learn basis Q-functions, that implicitly model dynamics and enable better generalization.

[1]To be released upon completion of the review process.

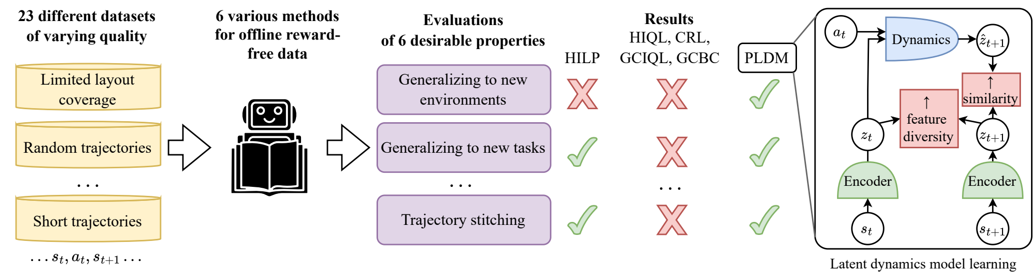


Figure 2: **Overview of our analysis.** We test six methods for learning from offline reward-free trajectories on 23 different datasets across two top-down navigation environments. We evaluate for six generalization properties required to scale to large offline datasets of suboptimal trajectories. We find that planning with a latent dynamics model (PLDM) demonstrates the highest level of generalization. For a full comparison, see Table 1. **Right:** diagram of PLDM. Circles represent variables, rectangles – loss components, half-ovals – trained models.

Training representations for RL. Another way to use large amounts of data to improve RL agents is using self-supervised learning (SSL). CURL (Laskin et al., 2020) introduce an SSL objective in addition to the standard RL objectives. Later works also explore using a separate pre-training stage (Schwarzer et al., 2021; Zhang et al., 2022; Nair et al., 2022). Zhou et al. (2024) show that pre-trained visual representations from DINO (Caron et al., 2021; Oquab et al., 2023) can be used to learn a word model for planning.

Investigating importance of offline data. ExORL (Yarats et al., 2022) show the importance of data for offline RL, and demonstrated that data collected using unsupervised RL enables off-policy RL algorithms to perform well in the offline setting; however, that study only compares using data collected by unsupervised RL and by task-specific agents, without giving a more fine-grained analysis on how different aspects of the data affect performance. Buckman et al. (2020) investigates the data importance for offline RL with rewards. Recently proposed OGBench (Park et al., 2024a) introduces multiple versions of offline data for a variety of goal-conditioned tasks; in contrast to that work, we focus on top-down navigation environments and build *23 different datasets* to perform a detailed study of methods' generalization, including to new tasks and environment layouts, as opposed to at most only three dataset versions for one task in OGBench and its focus on the single layout, goal-conditioned setting. Cobbe et al. (2018) investigate generalization in RL using variations in the environment akin to what we did in Section 4.7, although that study is using the online setting with rewards. Yang et al. (2023) also study generalization of offline GCRL, but focus on reaching out-of-distribution goals. Ghugare et al. (2024) study stitching generalization.

3 The Landscape of Reward-Free Offline RL

In this section, we formally introduce the setting of learning from state-action sequences without reward annotations and overview available approaches. We also introduce a method we call Planning with a Latent Dynamics Model (PLDM).

3.1 Problem Setting

We consider a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, p, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mu \in \mathcal{P}(\mathcal{S})$ denotes the initial state distribution, $p \in \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the transition dynamics, and $r \in \mathcal{S} \to \mathbb{R}$ denotes the reward function. We work in the offline setting, where we have access to a dataset of state-action sequences $\mathcal{D}$ which consists of transitions $(s_0, a_0, s_1, \ldots, a_{T-1}, s_T)$. We emphasize again that the offline dataset in our setting does not contain any reward information. In our experiments, we also consider only deterministic transition dynamics. The goal is, given $\mathcal{D}$, to find a policy $\pi \in \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$, to maximize cumulative reward r_z, where $\mathcal{Z}$ is the space of possible task definitions. Our goal is to make the best use of the offline dataset $\mathcal{D}$ to enable the agent to solve a variety of tasks in a given environment with potentially different layouts. During evaluation, unless otherwise specified, the agent is tasked to reach a goal state s_g, so the reward is defined as $r_g(s) = \mathbb{I}[s = s_g]$, and $\mathcal{Z}$ is equivalent to $\mathcal{S}$.

Common Solutions. In this work, we focus on methods that can learn to solve tasks from offline data without rewards. We do not consider methods that augment reward-labeled dataset with reward

Table 1: **Road-map of our generalization stress-testing experiments.** We test 4 offline goal-conditioned methods - HIQL, GCIQL, CRL, GCBC; a zero-shot RL method HILP, and a learned latent dynamics planning method PLDM. ★★★ denotes good performance in the specified experiment, ★★☆ denotes average performance, and ★☆☆ denotes poor performance. We see that HILP and PLDM are the best-performing methods, with PLDM standing out as the only method that does not completely fail in any of the settings.

Property (Experiment section)	**HILP**	**HIQL**	**GCIQL**	**CRL**	**GCBC**	**PLDM**
Transfer to new environments (4.7)	★☆☆	★☆☆	★☆☆	★☆☆	★☆☆	★★★
Transfer to a new task (4.6)	★★☆	★☆☆	★☆☆	★☆☆	★☆☆	★★★
Data efficiency (4.3)	★☆☆	★★☆	★★☆	★★☆	★★☆	★★★
Best-case performance (4.2)	★★★	★★★	★★★	★★★	★★☆	★★☆
Can learn from random trajectories (4.5)	★★★	★☆☆	★☆☆	★☆☆	★☆☆	★★☆
Can stitch suboptimal trajectories (4.4)	★★★	★☆☆	★☆☆	★☆☆	★☆☆	★★☆
Fail-proof in all settings	✗	✗	✗	✗	✗	✓

free data, as we believe that fully reward-free approach is more general. In offline RL, methods for learning without rewards fall into two broad categories: offline goal-conditioned RL and zero-shot RL methods that model the underlying task as a latent variable. In this work, we consider both categories, and select methods that we believe reflect the state of the art. We test all methods on goal-reaching, and test the zero-shot methods transfer to new tasks. The methods we investigate are:

GCIQL (Park et al., 2024b) – goal-conditioned version of Implicit Q-Learning (Kostrikov et al., 2021), a strong and widely-used method for offline RL.

HIQL (Park et al., 2024b) – a hierarchical GCRL method which trains two policies: one to place generate subgoals, and another one to reach the subgoals. Notably, both policies use the same value function.

HILP (Park et al., 2024c) – a method that learns state representations from the offline data such that the distance in the learned representation space is proportional to the number of steps between two states. A direction-conditioned policy is then learned to be able to move along any specified direction in the latent space.

CRL (Eysenbach et al., 2022) – uses contrastive learning to learn compatibility between states and possible reachable goals. The learned representation, which has been shown to be directly linked to goal-conditioned Q-function, is then used to train a goal-conditioned policy.

GCBC (Lynch et al., 2020; Ghosh et al., 2019) – Goal-Conditioned Behavior Cloning. This is the simplest baseline for goal-reaching.

3.2 Planning with a Latent Dynamics Model

Although the methods outlined in Section 3.1 cover a wide range of paradigms, none of them use the model-based approach, which achieves impressive performance in other settings (Deisenroth & Rasmussen, 2011; Silver et al., 2017; 2016; Rafailov et al., 2021). An easy way to use state-action sequences is to learn a dynamics model, making it a natural choice for our setting. For example, Nair et al. (2020); Pertsch et al. (2020) propose model-based methods for goal-reaching, and use image reconstruction objective. In this work, we choose to focus on just the dynamics learning objective, with added representation learning objectives to prevent collapse, and avoid using reconstruction. We introduce a model-based method named PLDM – Planning with a Latent Dynamics Model. We learn latent dynamics using a reconstruction-free self-supervised learning (SSL) objective, making this method a joint-embedding predictive architecture (JEPA) (LeCun, 2022). During evaluation, we use planning to optimize the goal-reaching objective. We opt for an SSL approach that involves predicting the latents as opposed to reconstructing the input observations (Hafner et al., 2018; 2019; 2020; 2023; Watter et al., 2015; Finn et al., 2016; Zhang et al., 2019; Banijamali et al., 2018) as recent work showed that reconstruction leads to suboptimal features (Balestriero & LeCun, 2024; Littwin et al., 2024), and that reconstruction-free representation learning can work well for control and RL (Shu et al., 2020; Hansen et al., 2022). More concretely, given agent trajectory sequence $(s_0, a_0, s_1, ..., a_{T-1}, s_T)$, we specify the PLDM world model as:

$$\text{Encoder}: \quad \hat{z}_0 = z_0 = h_\theta(s_0) \qquad \text{Predictor}: \qquad \hat{z}_t = f_\theta(\hat{z}_{t-1}, a_{t-1})$$

Where $\hat{z}_t$ is the predicted latent state and z_t is the encoder output at time index t. The training objective involves minimizing the distance between predicted and encoded latents summed over all timesteps. Given latents $Z \in \mathbb{R}^{T \times N \times D}$, where T is the time dimension, N is the batch dimension,

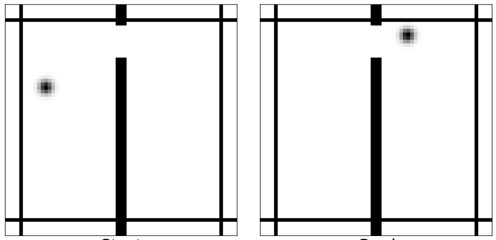


Figure 3

Method	Good-quality data	No door-passing trajectories
CRL	0.893 ± 0.012	0.147 ± 0.070
GCBC	0.860 ± 0.045	0.084 ± 0.037
GCIQL	0.936 ± 0.009	0.220 ± 0.041
HILP	1.000 ± 0.000	1.000 ± 0.000
HIQL	0.964 ± 0.030	0.263 ± 0.138
PLDM	0.860 ± 0.028	0.568 ± 0.031

Table 2

Figure 3: The Two-Rooms environment. The environment consists of two rooms separated by a wall with one door. The locations of the wall and the door remain fixed. The agent starts at a random location and is tasked with reaching the goal at another randomly sampled location in the other room within 200 steps. The evaluation episode is considered successful if the distance between the goal and the agent is below 1.4 pixels, with the environment being 64×64 pixels. **Table 2**: Performance of tested methods on good-quality data and on data with no trajectories passing through the door. Numbers are average success rates ($\pm$std) across 3 seeds. The dataset size is kept constant at 3M.

and D the feature dimension, the similarity objective between predictions and encodings is:

$$\mathcal{L}_{\text{sim}} = \sum_{t=0}^{T} \frac{1}{N} \sum_{b=0}^{N} \|\hat{Z}_{t,b} - Z_{t,b}\|_2^2$$

To prevent representation collapse, we use a VICReg-inspired (Bardes et al., 2021) objective, as well as inverse dynamics modeling (Lesort et al., 2018). We show a diagram of PLDM in Figure 2. For more details, see Appendix E.1.1.

Goal-conditioned planning with PLDM. At test time, given the current observation s_0, goal observation s_g, pretrained encoder h_θ and predictor f_θ, planning horizon H, our planning objective is:

$$\hat{z}_0 = z_0 = h_\theta(s_0),\ \hat{z}_t = f_\theta(\hat{z}_{t-1}, a_{t-1})$$

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} \text{Cost}(\mathbf{a}, s_0, s_g), \quad \text{Cost}(\mathbf{a}, s_0, s_g) = \sum_{t=0}^{H-1} \|h_\theta(s_g) - f_\theta(\hat{z}_t, a_t)\|$$

Following the Model Predictive Control framework (Morari & Lee, 1999), our model re-plans at every k_{th} interaction with the environment. Unless stated otherwise, we use $k = 1$. In all our experiments with PLDM, we use MPPI (Williams et al., 2015) for planning.

4 Every Method Can Excel but Few Generalize

We conduct thorough experiments testing a range of offline GCRL and zero-shot methods we outlined in Section 3.1 and Section 3.2. We test all methods on navigation tasks where the agent is a point mass. We generate datasets of varying size and quality and test how a specific data type affects a given method. We design our experiments to test the following properties of methods (see Table 1 for experiment overview): **P1** sample efficiency, **P2** ability to learn from random trajectories, **P3** ability to stitch together suboptimal trajectories, **P4** generalization to new environments variations, **P5** zero-shot generalization to a different task.

4.1 Environment

All our experiments are done with top-down navigation tasks with a point-mass agent. First, we introduce a two-rooms navigation task. Each observation x_t is a top-down view of the two-rooms environment, $x_t \in \mathbb{R}^{2\times64\times64}$, shown in Figure 3. The first channel in the image is the agent, the second channel is the walls. Actions $a \in \mathbb{R}^2$ denote the displacement vector of the agent position from one time step to the next one. The norm of the actions is restricted to be less than 2.45. The goal is to reach another randomly sampled state within 200 environment steps. See Appendix C.2 for more details. This environment makes control of the data generation process very easy, enabling

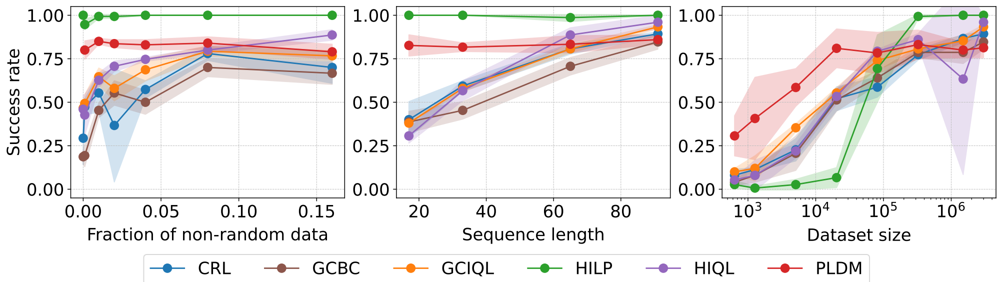


Figure 5: **Testing the selected methods' performance under different dataset constraints.** Values and shaded regions are means and standard deviations over 3 seeds, respectively. **Left**: tests the importance of the dataset quality, we mix the fully random trajectories and better quality trajectories (see Figure 6). As the amount of good quality data goes to 0, methods begin to fail, with PLDM and HILP being the most robust ones. **Center**: measures methods' performance when trained with different sequence lengths. We find that many goal-conditioned methods fail when train trajectories are short, which causes far-away goals to become out-of-distribution for the resulting policy. **Right**: measures methods' performance with datasets of varying sizes. We see that PLDM is the most sample efficient, and manages to get almost 50% success rate even with a few thousand transitions.

us to conduct our experiments efficiently and thoroughly, while not being so trivial that any method can solve it with even a little bit of data. Movement and navigation is a big part of virtually every real-world robotic environment, making this a useful testbed for development.

Offline data. To generate offline data, we place the agent in a random location within the environment, and execute a sequence of actions for T steps, where T denotes the episode length. The actions are generated by first picking a random direction, then using Von Mises distribution with concentration 5 to sample action directions. The step size is uniform from 0 to 2.45. When sampling low-quality data, we do not bias the action directions using Von Mises, and instead sample the direction completely uniformly. Unless otherwise specified, the episodes' length is $T = 91$, and the total number of transitions in the data is 3 million.

4.2 What is the Best-Case Performance?

We test all the methods in a setting with a large amount of data, with good state coverage, and with non-random trajectories. This should serve as top line performance in the perfect scenario. With 3 million transitions, corresponding to around 30 thousand trajectories, all methods reach their best-case performance in this environment. We report the results in Table 2. On the goal-reaching task in the two-rooms environment, all methods achieve impressive performance, with HIQL and HILP nearing perfect 100% success rate. PLDM fails to achieve perfect performance here. We hypothesize that because PLDM's training objective is not to learn a policy but to learn dynamics, PLDM does not make use of high-quality trajectories like other model-free methods.

> **Takeaway**: model-free approaches perform better than the model-based approach when the data is plentiful and high-quality.

4.3 What Method is the Most Sample-Efficient?

We investigate how different methods perform when the dataset size varies. While our ultimate goal is to have a method that can make use of a large amount of suboptimal offline data, this experiment serves to distinguish which methods can glean the most information from available data. We tried ranges of dataset sizes all the way down to a few thousand transitions. In Figure 5 we see that the model-based method PLDM outperforms model-free methods when the data is scarce. HILP is more data-hungry than other model-free methods, although it achieves perfect performance with enough data. We hypothesize that learning dynamics provides more learning signal to the model as opposed to learning a goal-conditioned value function and policy, making the model-based approach perform better.

Takeaway: A model-based approach is better than model-free approaches when the data is scarce.

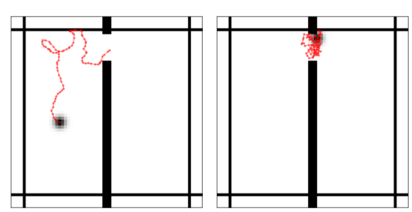
4.4 What Methods Can Stitch Suboptimal Trajectories?

Can we learn from short trajectories? In this experiment, we vary the episode length T when generating the data. This experiment aims to test the methods' ability to stitch together shorter trajectories in order to get to the goal. In real-life scenarios, collecting long episodes may be much more challenging than having a large set of shorter trajectories, especially when we scale to more open-ended environments. Therefore, the ability to learn and generalize effectively from shorter trajectories, even when the evaluation trajectory may be much longer, is essential. In our environment, successfully navigating from the bottom left corner to the bottom right corner requires around 90 steps. This means that successful trajectories for the hardest start goal pairings are never observed in a dataset with episodes of length 16. In order to successfully solve this task, the learning method has to be able to stitch together multiple offline trajectories. We generate several datasets, with episode length of 91, 64, 32, 16. We adjust the number of episodes to keep the total number of transitions close to 3 million. The results are shown in Figure 5 (center). We see that when the episode length is short, goal-conditioned methods fail. We hypothesize that because goal-conditioned methods sample state and goal pairs from a trajectory to train their policies, far away goals become out of distribution for the resulting policy. On the other hand, HILP performs well because instead of reaching goals, it learns to follow directions in the latent space, which can be learned even from short trajectories. Similarly, a model based method such as PLDM can learn an accurate model from short trajectories and stitch together a plan during test time.

Can we learn from data with imperfect coverage? We artificially constrain trajectories to always stay within one room within the episode, and never pass through the door. Without the constraint, around 35% trajectories pass through the door. During evaluation, the agent still needs to go through the door to reach the goal state. This also reflects possible constraints in real-life scenarios, as the ability to stitch offline trajectories together is essential to efficiently learn from offline data. The results are shown in Table 2. We see that HILP achieves perfect performance, while PLDM performs worse but does not fail completely. Similarly to the experiment with short trajectories, the GCRL methods fail. We hypothesize that the structure of the latent space allows HILP to stitch trajectories easily, while PLDM retains some performance due to the learned dynamics. Model-free GCRL methods fail because the goal in a different room from the current state is always out-of-distribution for the policy trained on trajectories staying in one room.

Takeaway: When solving the task requires 'stitching', HILP and the model-based approach are better than model-free GCRL.

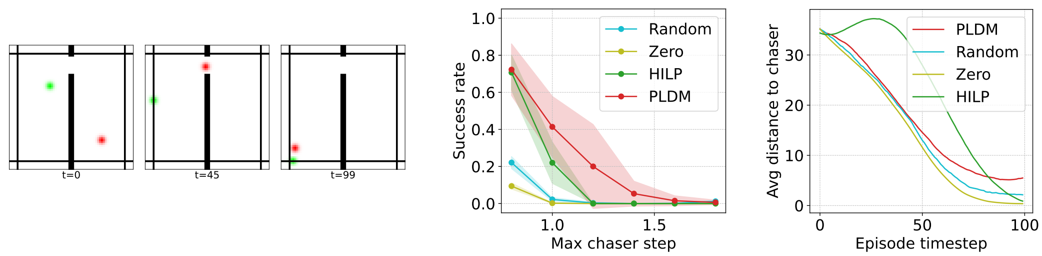
4.5 What Methods Can Learn From Random Trajectories?



(a) (b)

Figure 6: **Examples of trajectories used in the two-rooms environment. (a)** shows a trajectory where each step's direction is sampled from Von Mises distribution. **(b)** shows an example trajectory where each step is fully random, making the agent stay roughly in one place.

In this experiment, we evaluate how trajectory quality affects agent performance. In practice, collecting random trajectories is easy, but access to skilled demonstrations cannot always be assumed. Therefore, developing an algorithm that can learn from noisy or random trajectories is critical for leveraging all available data. We generate a dataset of noisy trajectories, where at each step the direction is sampled completely at random. This effectively makes the agent move randomly, and throughout the episode it mostly stays close to where it started. In this dataset, the average maximum distance between any two points in a trajectory is ~ 10 (when the whole environment is 64 by 64), while when using Von Mises to sample actions, it is ~ 28. Example trajectories from both types of action sampling are shown in Figure 6. Again, we see that HILP and the model-based PLDM perform better with noisy data, while the goal-conditioned RL methods struggle (Figure 5). Similarly



(a) (b) (c)

Figure 7: **Testing zero-shot generalization to the chasing task.** **(a)** Chase environment. The controlled agent (in green) is tasked with avoiding the chaser agent (red). The chaser agent follows the shortest path to the agent. The observations of the agent remain unchanged: we pass the chaser state as the goal state. The agent has to avoid the specified state instead of reaching it. **(b)** Performance of the tested methods on the chasing task across different chaser speeds, with faster chaser making the task harder. We baseline against policies that always take zero actions (labeled as 'Zero') and policies that always take random actions (labeled as 'Random'). **(c)** Average distance between the controlled agent and the chaser agent throughout the episode when chaser speed is 1.0.

to the experiment with shorter trajectories, we hypothesize that because trajectories on average do not go far, the sampled state and goal pairs during training are close to each other, making faraway goals out of distribution. On the other hand, PLDM uses the data only to learn the dynamics model, and random trajectories are still suitable for the purpose. HILP uses the data to learn the latent space and how to traverse it in various dimensions, and can also use the random trajectories effectively.

Takeaway: When the dataset quality is low, HILP and the model-based method perform better than offline GCRL.

4.6 How Well Do Zero-Shot Methods Generalize to a New Task?

In order to build a system that can learn from offline data effectively, we need a learning algorithm that can generalize to different tasks. So far, we compared all the methods on goal-reaching tasks. In this experiment, we test whether the selected methods are able to generalize to a different task in the same environment. In this section, we compare the performances of PLDM and HILP on the task of avoiding another agent that is 'chasing' the controlled agent. We evaluate models trained on optimal data from the experiment in Section 4.2 without any additional training. In this task, the chasing agent follows an expert policy that moves toward the agent along the shortest path. To vary the difficulty of the task, we vary the speed of the chasing agent. The goal of the controlled agent is to avoid the chasing agent. We note that the goal-conditioned methods can only reach specified goals, and by definition are unable to avoid a given state. Therefore, we only test PLDM and HILP. At each step, the agent is given the state of the chaser agent, and has to choose actions to avoid the chaser. To achieve that, in PLDM we simply invert the sign of the planning objective, making planning maximize the distance in representation space to the goal state. In HILP, we invert the skill direction. To compare the two methods, we evaluate the success rate of the controlled agent avoiding the chaser agent. The episode is considered successful if the agent manages to stay away from the chaser by at least 1.4 pixels for the whole episode lasting 100 steps. The results are shown in Figure 7b. To further analyze the results, we also investigate average distance between the agents throughout the episode, and plot the average, see Figure 7c. We see that PLDM performs better than HILP, and is able to evade the chaser more efficiently, keeping a larger distance between the agents at the end of the episode.

Takeaway: planning with a latent dynamics model generalizes better to a new task compared to HILP.

4.7 What Methods Can Generalize to Unseen Environments?

In this experiment we test the ability of the tested methods to generalize to new environments. Generalization to new environment variations is a requirement for any truly general RL agent, as it is impossible to collect data for every scenario. To test this, we introduce another navigation environment featuring more complex dynamics and configurable layouts, see Figure 1 for an example. We utilize the Mujoco PointMaze environment (Todorov et al., 2012) and generate various maze layouts by randomly permuting wall locations. The data is collected by initializing the agent at a random location and sampling actions randomly at every step. The observation space contains the top down view of the maze in RGB image format and the velocity of the agent, while the action is the 2D acceleration vector. The goal is to reach a randomly sampled goal state in the environment. For more details about the environment, see Appendix C.2. To study the generalization ability of our agents, we vary the number (5, 10, 20, 40) of pre-training maze layouts in the offline dataset and evaluate the trained agents on a held-out set of unseen layouts. Furthermore, for agents trained on 5 layouts, we analyze how their performance is affected by the degree to which the test layouts differ from the training layouts in distribution. We show the results in Figure 1, and more details in Figure 9. PLDM demonstrates the best performance, generalizing to unseen environments, even when trained on as few as five maps, while other methods fail. In particular, as the test layouts move out of distribution from train layouts, all methods except PLDM suffer in performance as a consequence. To make sure all methods are able to solve the task, we also evaluate the methods on a fixed layout, and see that all of them are able to reach 100% success rate, see Table 5. Figure 8 and 9 show the plans inferred by PLDM at test time, as well as the different agents' trajectories.

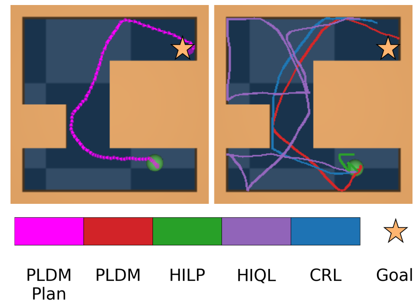


Figure 8: **Left**: Plans generated by PLDM at test time. **Right**: Actual agent trajectories for the tested methods. PLDM is the only method that reliably reaches the goal on held-out mazes.

Takeaway: The model-based approach enables better generalization to unseen environment variations than model-free methods.

5 Conclusion

In this work, we conducted a comprehensive study of existing RL methods for learning from offline data without rewards, aiming to identify the most promising approaches for leveraging large datasets of suboptimal trajectories. Our findings highlight HILP and PLDM as the strongest candidates, with PLDM demonstrating the highest robustness to variations in data quality. We aggregate our results in Table 1. Overall, we draw 3 main conclusions:

- **C1** The model-based approach exhibits robustness to data quality, superior data efficiency, and the best generalization to new layouts and tasks;
- **C2** Learning a well-structured latent-space (e.g. using HILP) enables trajectory stitching and robustness to data quality, although it is more data-hungry than other methods;
- **C3** Model-free GCRL methods are a great choice when the data is plentiful and good quality.

Moving forward. PLDM works well across different dataset settings, and is able to learn from poor data and generalize to novel environments and tasks. Therefore, we believe that learning latent dynamics models is a promising candidate for pre-training on large datasets of suboptimal trajectories. Dynamics learning can also be extended to data without actions by modeling them as a latent variable (Seo et al., 2022; Ye et al., 2022). We believe that other non-generative objectives for latent representation learning (Oquab et al., 2023; Bardes et al., 2024) can be used to further improve performance. Another promising direction of research is planning. Dynamics learning and planning bring their own set of issues, including accumulating prediction errors (Lambert et al., 2022) and increased computational complexity during inference. In our case, we used MPPI (Williams et al., 2015) for planning with a learned dynamics model, which takes a considerable amount of time, making evaluation with PLDM about 100 times slower compared to model-free methods (see Appendix F for details). Further research into making planning more efficient by e.g. backpropagating through the forward model is needed (Bharadhwaj et al., 2020). In domains where inference speed is important, planning can be also used as the target to train a policy (Liu et al., 2022).

REFERENCES

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.

Balestriero, R. and LeCun, Y. Learning by reconstruction produces uninformative features for perception. *arXiv preprint arXiv:2402.11337*, 2024.

Banijamali, E., Shu, R., Ghavamzadeh, M., Bui, H., and Ghodsi, A. Robust locally-linear controllable embedding. (arXiv:1710.05373), February 2018. doi: 10.48550/arXiv.1710.05373. URL `http://arxiv.org/abs/1710.05373`. arXiv:1710.05373 [cs].

Bardes, A., Ponce, J., and LeCun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., and Ballas, N. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

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.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Buckman, J., Gelada, C., and Bellemare, M. G. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Chen, B., Zhu, C., Agrawal, P., Zhang, K., and Gupta, A. Self-supervised reinforcement learning that transfers using random features. (arXiv:2305.17250), May 2023. doi: 10.48550/arXiv.2305.17250. URL `http://arxiv.org/abs/2305.17250`. arXiv:2305.17250 [cs].

Cho, K. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. arxiv preprint arxiv: 181202341. 2018.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

Eysenbach, B., Zhang, T., Levine, S., and Salakhutdinov, R. R. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 35603–35620, 2022.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Deep spatial autoencoders for visuomotor learning. (arXiv:1509.06113), March 2016. doi: 10.48550/arXiv.1509.06113. URL `http://arxiv.org/abs/1509.06113`. arXiv:1509.06113 [cs].

Frans, K., Park, S., Abbeel, P., and Levine, S. Unsupervised zero-shot reinforcement learning via functional reward encodings. *arXiv preprint arXiv:2402.17135*, 2024.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Ghugare, R., Geist, M., Berseth, G., and Eysenbach, B. Closing the gap between td learning and supervised learning–a generalisation point of view. *arXiv preprint arXiv:2401.11237*, 2024.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent 423 dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 424, 2018.

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

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Hatch, K., Yu, T., Rafailov, R., and Finn, C. Example-based offline reinforcement learning without rewards. *Proceedings of Machine Learning Research vol*, 144:1–17, 2022.

Hu, H., Yang, Y., Zhao, Q., and Zhang, C. The provable benefits of unsupervised data sharing for offline reinforcement learning. *arXiv preprint arXiv:2302.13493*, 2023.

Kim, J., Park, S., and Levine, S. Unsupervised-to-online reinforcement learning. *arXiv preprint arXiv:2408.14785*, 2024.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020. URL `https://arxiv.org/abs/2006.04779`.

Lambert, N., Pister, K., and Calandra, R. Investigating compounding prediction errors in learned dynamics models. (arXiv:2203.09637), March 2022. doi: 10.48550/arXiv.2203.09637. URL `http://arxiv.org/abs/2203.09637`. arXiv:2203.09637 [cs].

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.

LeCun, Y. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Littwin, E., Saremi, O., Advani, M., Thilak, V., Nakkiran, P., Huang, C., and Susskind, J. How jepa avoids noisy features: The implicit bias of deep linear self distillation networks. *arXiv preprint arXiv:2407.03475*, 2024.

Liu, I.-C. A., Uppal, S., Sukhatme, G. S., Lim, J. J., Englert, P., and Lee, Y. Distilling motion planner augmented policies into visual control policies for robot manipulation. In *Conference on Robot Learning*, pp. 641–650. PMLR, 2022.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. PMLR, 2020.

Mnih, V. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Morari, M. and Lee, J. H. Model predictive control: past, present and future. *Computers & chemical engineering*, 23(4-5):667–682, 1999.

Nair, S., Savarese, S., and Finn, C. Goal-aware prediction: Learning to model what matters. In *International Conference on Machine Learning*, pp. 7207–7219. PMLR, 2020.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

OpenAI, M. A., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J. W., Pachocki, J., Petron, A., Plappert, M., Powell, G., et al. Learning dexterous in-hand manipulation. corr abs/1808.00177 (2018). *arXiv preprint arXiv:1808.00177*, 2018.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Park, S., Frans, K., Eysenbach, B., and Levine, S. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024a.

Park, S., Ghosh, D., Eysenbach, B., and Levine, S. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024b.

Park, S., Kreiman, T., and Levine, S. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024c.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pertsch, K., Rybkin, O., Ebert, F., Zhou, S., Jayaraman, D., Finn, C., and Levine, S. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

Rafailov, R., Yu, T., Rajeswaran, A., and Finn, C. Offline reinforcement learning from images with latent space models. In *Learning for dynamics and control*, pp. 1154–1168. PMLR, 2021.

Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, R. D., Bachman, P., and Courville, A. C. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Seo, Y., Lee, K., James, S. L., and Abbeel, P. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022.

Shu, R., Nguyen, T., Chow, Y., Pham, T., Than, K., Ghavamzadeh, M., Ermon, S., and Bui, H. H. Predictive coding for locally-linear control. (arXiv:2003.01086), March 2020. doi: 10.48550/arXiv.2003.01086. URL `http://arxiv.org/abs/2003.01086`. arXiv:2003.01086 [cs].

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.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017.

Sobal, V., SV, J., Jalagam, S., Carion, N., Cho, K., and LeCun, Y. Joint embedding predictive architectures focus on slow features. *arXiv preprint arXiv:2211.10831*, 2022.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL `http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12`.

Touati, A. and Ollivier, Y. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Touati, A., Rapin, J., and Ollivier, Y. Does zero-shot reinforcement learning exist? *arXiv preprint arXiv:2209.14935*, 2022.

Wang, K., Zhou, K., Zhang, Q., Shao, J., Hooi, B., and Feng, J. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pp. 11003–11012. PMLR, 2021.

Wang, K., Zhou, K., Feng, J., Hooi, B., and Wang, X. Reachability-aware laplacian representation in reinforcement learning. *arXiv preprint arXiv:2210.13153*, 2022.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. (arXiv:1506.07365), November 2015. doi: 10.48550/arXiv.1506.07365. URL `http://arxiv.org/abs/1506.07365`. arXiv:1506.07365 [cs].

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Wu, Y., Tucker, G., and Nachum, O. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Yang, R., Yong, L., Ma, X., Hu, H., Zhang, C., and Zhang, T. What is essential for unseen goal generalization of offline goal-conditioned rl? In *International Conference on Machine Learning*, pp. 39543–39571. PMLR, 2023.

Yarats, D., Brandfonbrener, D., Liu, H., Laskin, M., Abbeel, P., Lazaric, A., and Pinto, L. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Ye, W., Zhang, Y., Abbeel, P., and Gao, Y. Become a proficient player with limited data through watching pure videos. In *The Eleventh International Conference on Learning Representations*, 2022.

Yu, T., Thomas, G., Yu, L., Ma, T. X., Ermon, S., Zou, J., and Finn, C. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020. URL `https://arxiv.org/abs/2005.13239`.

Yu, T., Kumar, A., Chebotar, Y., Hausman, K., Finn, C., and Levine, S. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 25611–25635. PMLR, 2022.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M. J., and Levine, S. Solar: Deep structured representations for model-based reinforcement learning. (arXiv:1808.09105), June 2019. doi: 10.48550/arXiv.1808.09105. URL `http://arxiv.org/abs/1808.09105`. arXiv:1808.09105 [cs].

Zhang, W., GX-Chen, A., Sobal, V., LeCun, Y., and Carion, N. Light-weight probing of unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2208.12345*, 2022.

Zhou, G., Pan, H., LeCun, Y., and Pinto, L. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A IMPACT STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning and Reinforcement Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

B LIMITATIONS.

All our experiments were conducted in simple navigation environments, and it is unclear if these findings will translate to more complex environments, e.g. physical robots data. However, we argue that the conceptual understanding of the effects of data quality on the investigated methods will hold, as even in the relatively simple setting, we see many recent methods break down in surprising ways.

C ENVIRONMENTS AND DATASETS

C.1 TWO-ROOMS ENVIRONMENT

We build our own top-down navigation environment. It is implemented in PyTorch (Paszke et al., 2019), and supports GPU acceleration. The environment does not model momentum, i.e. the agent does not have velocity, and is moved by the specified action vector at each step. When the action takes the agent through a wall, the agent is moved to the intersection point between the action vector and the wall. We generate the specified datasets and save them to disk for our experiments. The datasets generation takes under 30 minutes.

C.2 DIVERSE POINTMAZE

Here, we build upon the Mujoco PointMaze environment (Todorov et al., 2012), which contains a point mass agent with a 4D state vector $(\text{global } x, \text{global } y, v_x, v_y)$, where v is the agent velocity. To allow our models to perceive the different maze layouts, we use as model input a top down view of the maze rendered as $(64, 64, 3)$ RGB image tensor instead of relying on $(\text{global } x, \text{global})$ directly.

Mujoco PointMaze allows for customization of the maze layout via a grid structure, where a grid cell can either be a wall or space. We opt for a 4×4 grid (excluding outer wall). Maze layouts are generated randomly. Only the following constraints are enforced: 1) all the space cells are interconnected, 2) percentage of space cells range from 50% to 75%.

We set action repeat to 4 for our version of the environment.

C.2.1 DATASET GENERATION

We produce four training datasets with the following parameters:

# Transitions	# layouts	# episodes per layout	episode length
1000000	5	2000	100
1000000	10	1000	100
1000000	20	500	100
1000000	40	250	100

Table 3: Details for Diverse PointMaze datasets

Each episode is collected by setting the $(\text{global } x, \text{global})$ at a random location in the maze, and agent velocity (v_x, v_y) by randomly sampling a 2D vector with $\|v\| \leq 5$, given that v_x and v_y are clipped within range of $[-5, 5]$ in the environment.

C.2.2 Evaluation

All the test layouts during evaluation are disjoint from the training layouts. For each layout, trials are created by randomly sampling a start and goal position guaranteed to be at least 3 cells away on the maze. The same set of layouts and trials are used to evaluate all agents for a given experimental setting.

We evaluate agents in two scenarios: 1) How agents perform on test layouts when trained on various number of train layouts; 2) Given a constant number of training layouts, how agents perform on test maps with varying degrees of distribution shift from the training layouts.

For scenario 1), we evaluate the agents on 40 randomly generated test layouts, 1 trial per layout.

For scenario 2), we randomly generate test layouts and partition them into groups of 5, where all the layouts in each group have the same degree of distribution shift from train layout as defined by metric D_{min} defined as the following:

Given train layouts $\{L^1_{\text{train}}, L^2_{\text{train}}, ...L^N_{\text{train}}\}$, test layout L_{test}, and let $d(L_1, L_2)$ represents the edit distance between two layouts L_1 and L_2's binary grid representation. We quantify the distribution shift of L_{test} as $D_{\min} = \min_{i \in \{1,2,\ldots,N\}} d(L_{\text{test}}, L^{(i)}_{\text{train}})$.

In this second scenario we evaluate 5 trials per layout, thus a total of $5 \times 5 = 25$ per group.

D Visualization of Plans and Trajectories for Diverse PointMaze

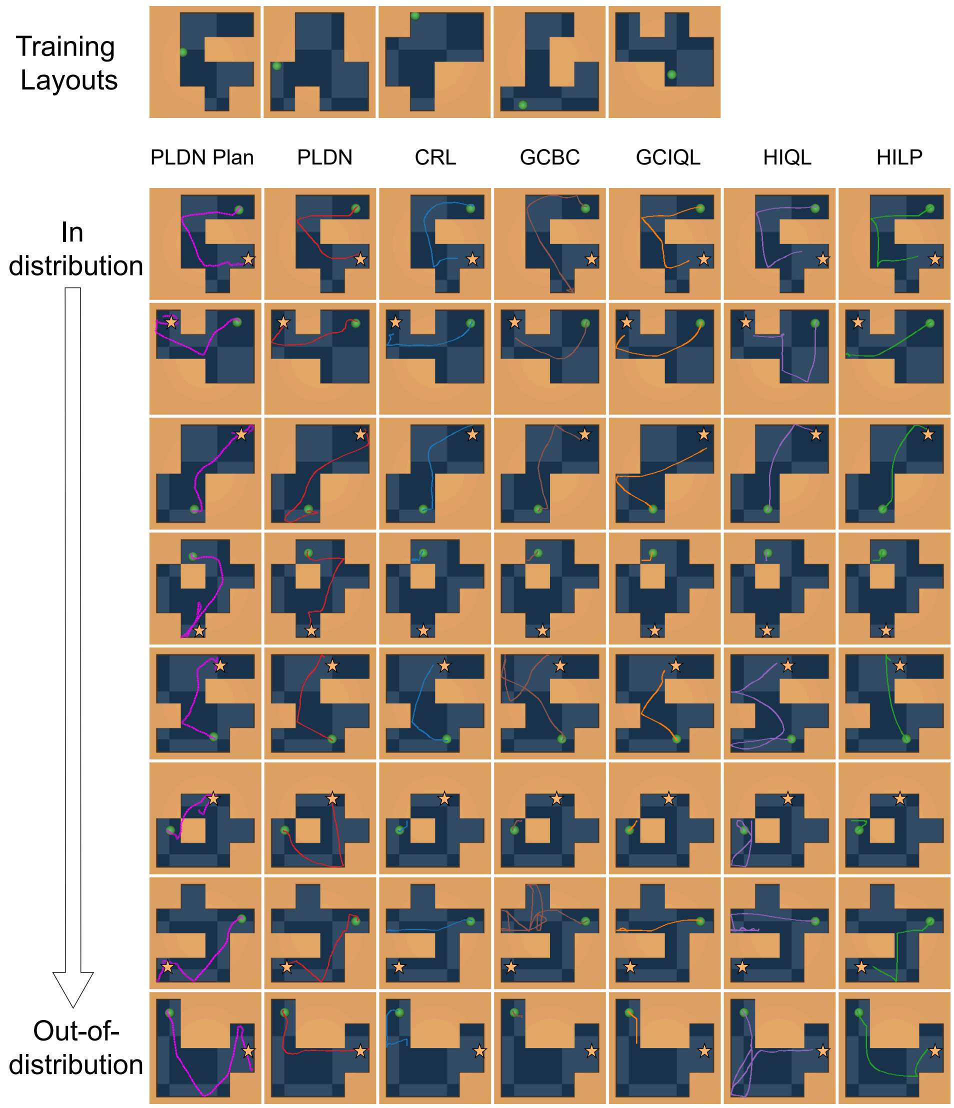


Figure 9: **Top**: The training layouts used in the 5 layout setting. **Middle**: Trajectories of different agents navigating an unseen maze layout towards goal at test time. As the layouts become increasingly out-of-distribution, only PLDN consistently succeeds. Layouts can be represented as a 4x4 array, with each value being either a wall or empty space. The distribution shift is quantified by the minimum edit distance between a given test layout and the closest training layout. The top row corresponds to an in-distribution layout with a minimum edit distance of 0, and with each subsequent row, the minimum edit distance increases by 1 incrementally.

E Models

For CRL, GCBC, GCIQL, and HIQL we use the implementations from the repository[2] of OGBench (Park et al., 2024a). Likewise, for HILP we use the official implementation[3] from its authors.

[2] https://github.com/seohongpark/ogbench
[3] https://github.com/seohongpark/HILP

For the Diverse PointMaze environment, to keep things consistent with our implementation of PLDM (E.1.3), instead of using frame stacking, we append the agent velocity directly to the encoder output.

E.1 PLDM

E.1.1 OBJECTIVE FOR COLLAPSE PREVENTION

To prevent collapse, we introduce a VICReg-based (Bardes et al., 2021) objective. We modify it to apply variance objective across the time dimension to encourage features to capture information that changes, as opposed to information that stays fixed (Sobal et al., 2022). The objective to prevent collapse is defined as follows:

$$\mathcal{L}_{\text{var}} = \frac{1}{TD} \sum_{t=0}^{T} \sum_{j=0}^{D} \max(0, \gamma - \sqrt{\text{Var}(Z_{t,:,j}) + \epsilon})$$

$$\mathcal{L}_{\text{time-var}} = \frac{1}{ND} \sum_{b=0}^{N} \sum_{j=0}^{D} \max(0, \gamma - \sqrt{\text{Var}(Z_{:,b,j}) + \epsilon})$$

$$C(Z_t) = \frac{1}{N-1}(Z_t - \bar{Z}_t)^\top (Z_t - \bar{Z}_t), \ \bar{Z} = \frac{1}{N} \sum_{b=1}^{N} Z_{t,b}$$

$$\mathcal{L}_{\text{cov}} = \frac{1}{T} \sum_{t=0}^{T} \frac{1}{D} \sum_{i \neq j} [C(Z_t)]_{i,j}^2$$

$$\mathcal{L}_{\text{IDM}} = \sum_{t=0}^{T} \frac{1}{N} \sum_{b=0}^{N} \|a_{t,b} - \text{MLP}(Z_{(t,b)}, Z_{(t+1,b)})\|_2^2$$

We also apply a tunable objective to enforce the temporal smoothness of learned representations:

$$\mathcal{L}_{\text{time-sim}} = \sum_{t=0}^{T-1} \frac{1}{N} \sum_{b=0}^{N} \|Z_{t,b} - Z_{t+1,b}\|_2^2$$

The combined objective is a weighted sum of above:

$$\mathcal{L}_{\text{JEPA}} = \mathcal{L}_{\text{sim}} + \alpha \mathcal{L}_{\text{var}} + \beta \mathcal{L}_{\text{cov}} + \lambda \mathcal{L}_{\text{time-var}} + \delta \mathcal{L}_{\text{time-sim}} + \omega \mathcal{L}_{\text{IDM}}$$

E.1.2 MODEL DETAILS FOR TWO-ROOMS

We use the same Impala Small Encoder used by the other methods from OGBench (Park et al., 2024a). For predictor, we use the a 2-layer Gated recurrent unit (Cho, 2014) with 512 hidden dimensions; the predictor input at timestep t is a 2D displacement vector representing agent action at timestep t; while the initial hidden state is $h_\theta(s_0)$, or the encoded state at timestep 0. A single layer normalization layer is applied to the encoder and predictor outputs across all timesteps. Parameter counts are the following:

```
total params: 2218672
encoder params: 1426096
predictor params: 793600
```

E.1.3 MODEL DETAILS FOR DIVERSE POINTMAZE ENVIRONMENT

For the Diverse PointMaze environment, we use convolutional networks for both the encoder and predictor. To fully capture the agent's state at timestep t, we first encode the top down view of the maze to get a spatial representation of the environment $h_\theta : \mathbb{R}^{3\times64\times64} \rightarrow \mathbb{R}^{16\times26\times26}, z^{env} = h_\theta(s^{env})$. We incorporate the agent velocity by first transforming it into planes $\text{Expander2D} : \mathbb{R}^2 \rightarrow \mathbb{R}^{2\times26\times26}, s^{vp} = \text{Expander2D}(s^v)$, where each slice $s^{vp}[i]$ is filled with $s^v[i]$. Then, we concatenate the expanded velocity tensor with spatial representation along the channel dimension to get our overall representation: $z = \text{concat}(s^{vp}, z^{env}, \dim = 0) \in \mathbb{R}^{18\times26\times26}$.

For the predictor input, we concatenate the state $s_t \in \mathbb{R}^{18\times 26\times 26}$ with the expanded action $\mathrm{Expander2D}(a_t) \in \mathbb{R}^{2\times 26\times 26}$ along the channel dimension. The predictor output has the same dimension as the representation: $\hat{z} \in \mathbb{R}^{18\times 26\times 26}$. Both the encodings and predictions are flattened for computing the VicReg and IDM objectives.

We set the planning-frequency (Section 3.2) in MPPI to $k = 4$ for this environment.

The full model architecture is summarized using PyTorch-like notations.

```
total params: 53666
encoder params: 33296
predictor params: 20370

PLDM(
    (backbone): MeNet6(
        (layers): Sequential(
            (0): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1))
            (1): GroupNorm(4, 16, eps=1e-05, affine=True)
            (2): ReLU()
            (3): Conv2d(16, 32, kernel_size=(5, 5), stride=(2, 2))
            (4): GroupNorm(8, 32, eps=1e-05, affine=True)
            (5): ReLU()
            (6): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
            (7): GroupNorm(8, 32, eps=1e-05, affine=True)
            (8): ReLU()
            (9): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding
                =(1, 1))
            (10): GroupNorm(8, 32, eps=1e-05, affine=True)
            (11): ReLU()
            (12): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1))
        )
        (propio_encoder): Expander2D()
    )
    (predictor): ConvPredictor(
        (layers): Sequential(
            (0): Conv2d(20, 32, kernel_size=(3, 3), stride=(1, 1), padding
                =(1, 1))
            (1): GroupNorm(4, 32, eps=1e-05, affine=True)
            (2): ReLU()
            (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding
                =(1, 1))
            (4): GroupNorm(4, 32, eps=1e-05, affine=True)
            (5): ReLU()
            (6): Conv2d(32, 18, kernel_size=(3, 3), stride=(1, 1), padding
                =(1, 1))
        )
        (action_encoder): Expander2D()
    )
)
```

F Analyzing planning time of PLDM

In order to estimate how computationally expensive it is to run planning with a latent dynamics model, we evaluate PLDM and GCIQL on 25 episodes in the two-rooms environment. Each episode is 200 steps. We record the average time per episode and the standard deviation. We do not run HILP, GCBC, or CRL because the resulting policy architecture is the same, making the evaluation time identical to that of GCIQL. HIQL takes more time due to the hierarchy of policies. The results are shown below:

Table 4: Time of evaluation on one episode in two-rooms environment. PLDM is about 100 times slower than model-free methods. Time is calculated by running on 25 episodes.

Method	Time per episode (seconds)
PLDM	13.44 ± 0.11
GCIQL	0.12 ± 0.03
HIQL	0.16 ± 0.03

G Results for Single Maze Setting

Table 5: Results averaged over 3 seeds ± std

Method	Sucess rate)
PLDM	0.990 ± 0.001
CRL	0.980 ± 0.001
GCBC	0.970 ± 0.024
GCIQL	1.000 ± 0.000
HIQL	1.000 ± 0.000
HILP	1.000 ± 0.000

H Hyperparameters

H.1 CRL, GCBC, GCIQL, HIQL, HILP

Unless listed below, all hyperparameters remain consistent with default values from OGBench and HILP repositories. [2],[3]

H.1.1 Two-Rooms

For all methods, we used the learning rate of 3e-4. The rest of the hyperparameters were kept default.

Table 6: HILP hyperparameters

Hyperparam	Value
Expectile	0.7
Skill expectile	0.7

Table 7: HIQL hyperparameters

Hyperparam	Value
High-level AWR alpha	3.0
Low-level AWR alpha	3.0
Expectile	0.7

Table 8: GCIQL hyperparameters

Hyperparam	Value
Actor-loss	DDPG-BC
BC coefficient	0.3
Expectile	0.7

H.1.2 DIVERSE POINTMAZE

Table 9: Dataset specific hyperparameters of CRL, GCBC, GCIQL, HIQL, HILP for the Diverse PointMaze environment. For HILP, we set the same value for expectile and skill expectile.

Dataset	CRL	GCBC	GCIQL		HIQL		HILP	
	LR	LR	LR	Expectile	LR	Expectile	LR	Expectile
# map layouts = 5	0.0003	0.0003	0.0002	0.8	0.0001	0.7	0.0001	0.9
# map layouts = 10	0.0003	0.0001	0.0001	0.9	0.0001	0.7	0.0001	0.9
# map layouts = 20	0.0003	0.0001	0.0001	0.6	0.0003	0.7	0.0001	0.9
# map layouts = 40	0.0003	0.0001	0.0003	0.9	0.0001	0.9	0.0001	0.9

H.2 PLDM

H.2.1 TWO-ROOMS

The best case setting is sequence length = 91, dataset size = 3M, non-random $\%$ = 100, wall crossing $\% \approx 35$. For our experiments we vary each of the above parameters individually.

Table 10: Dataset-agnostic hyperparameters

Hyperparameter	Value
Batch Size	64
Optimizer	Adam
Scheduler	Cosine
MPPI noise σ	5
MPPI $\#$ samples	500
MPPI λ	0.005

For the dataset specific hyperparameters, we tune the following parameters from Appendix E.1.1:

Table 11: Dataset specific hyperparameters

Dataset	LR	α	β	λ	δ	ω
Sequence length = 91	0.0007	4.0	6.9	0.25	0.75	1.0
Sequence length = 65	0.0003	5.0	6.9	0.25	0.75	1.0
Sequence length = 33	0.0007	5.0	6.9	0.25	0.75	1.0
Sequence length = 17	0.0028	3.0	6.9	0.25	0.75	1.0
Dataset size = 634	0.0030	2.2	13.0	0.19	0.50	2.0
Dataset size = 1269	0.0010	2.2	13.0	0.19	0.50	2.0
Dataset size = 5078	0.0010	2.2	13.0	0.19	0.50	2.0
Dataset size = 20312	0.0030	2.2	13.0	0.19	0.50	2.0
Dataset size = 81250	0.0010	2.2	13.0	0.19	0.50	2.0
Dataset size = 325k	0.0010	4.0	6.9	0.25	0.75	1.0
Dataset size = 1500k	0.0010	4.0	6.9	0.25	0.75	1.0
Non-random % = 0.001	0.0007	5.5	9.7	0.42	0.38	1.4
Non-random % = 0.01	0.0007	3.9	6.5	0.27	0.19	0.6
Non-random % = 0.02	0.0007	3.9	6.5	0.31	0.72	0.5
Non-random % = 0.04	0.0007	3.9	6.9	0.25	0.75	1.0
Non-random % = 0.08	0.0007	3.9	6.5	0.31	0.24	1.5
Non-random % = 0.16	0.0007	3.0	6.5	0.40	0.27	1.4
Wall crossing % = 0	0.0007	4.0	6.9	0.25	0.75	1.0

H.2.2 DIVERSE POINTMAZE

Table 12: Dataset-agnostic hyperparameters

Hyperparameter	**Value**
Epochs	5
Batch Size	128
Optimizer	Adam
Scheduler	Cosine
MPPI noise σ	5
MPPI # samples	500
MPPI λ	0.0025

Table 13: Dataset specific hyperparameters

Dataset	**LR**	α	β	λ	δ	ω
# map layouts = 5	0.04	35.0	12.0	3.0	0.1	5.4
# map layouts = 5	0.04	35.0	12.0	3.0	0.1	5.4
# map layouts = 20	0.05	54.5	15.5	2.7	0.1	5.2
# map layouts = 40	0.05	54.5	15.5	2.7	0.1	5.2