Foundation Policies with Hilbert Representations

Seohong Park [1] Tobias Kreiman [1] Sergey Levine [1]

Abstract

Unsupervised and self-supervised objectives, such as next token prediction, have enabled pre-training generalist models from large amounts of unlabeled data. In reinforcement learning (RL), however, finding a truly general and scalable unsupervised pre-training objective for *generalist policies* from offline data remains a major open question. While a number of methods have been proposed to enable generic self-supervised RL, based on principles such as goal-conditioned RL, behavioral cloning, and unsupervised skill learning, such methods remain limited in terms of either the diversity of the discovered behaviors, the need for high-quality demonstration data, or the lack of a clear adaptation mechanism for downstream tasks. In this work, we propose a novel unsupervised framework to pre-train generalist policies that capture diverse, optimal, long-horizon behaviors from unlabeled offline data such that they can be quickly adapted to any arbitrary new tasks in a zero-shot manner. Our key insight is to learn a structured representation that preserves the temporal structure of the underlying environment, and then to span this learned latent space with directional movements, which enables various zero-shot policy "prompting" schemes for downstream tasks. Through our experiments on simulated robotic locomotion and manipulation benchmarks, we show that our unsupervised policies can solve goal-conditioned and general RL tasks in a zero-shot fashion, even often outperforming prior methods designed specifically for each setting. Our code and videos are available at `https://seohong.me/projects/hilp/`.

1. Introduction

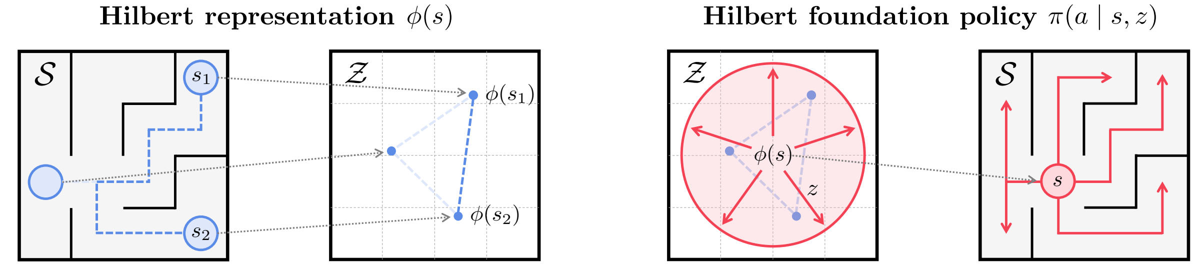
Generalist models that can utilize large amounts of weakly labeled data provide an appealing recipe: pre-train via self-supervised or unsupervised objectives on large and diverse datasets without ground truth labels, and then adapt efficiently via prompting, few-shot learning, or fine-tuning to downstream tasks. This strategy has proven to be extremely effective in settings where simple self-supervised objectives can be used to train on Internet-scale data (Brown et al., 2020; Ramesh et al., 2022), leading to models that can quickly adapt to new tasks for pattern recognition (Kirillov et al., 2023), question answering (Ouyang et al., 2022), and even diverse AI-assistant applications (Chen et al., 2021b). Motivated by this observation, a number of works have recently sought to propose self-supervised objectives to pre-train generalist *policies* for reinforcement learning (RL) and control (Reed et al., 2022; Padalkar et al., 2024). We can broadly refer to the resulting models as *foundation policies*: general-purpose policies that can rapidly adapt to solve a variety of downstream tasks.

However, unlike natural language processing, where next token prediction has become the standard pre-training objective (Brown et al., 2020), finding the best *policy pre-training objective* from data remains a major open question in RL. Prior works have proposed several ways to pre-train generalist policies based on diverse objectives, such as behavioral cloning (BC) (Ajay et al., 2021; Reed et al., 2022; Padalkar et al., 2024), offline goal-conditioned RL (GCRL) (Chebotar et al., 2021; Eysenbach et al., 2022; Park et al., 2023), and unsupervised skill discovery (Gregor et al., 2016; Machado et al., 2017; Eysenbach et al., 2019; Park et al., 2024). However, none of these objectives is ideal: behavioral cloning requires expert demonstrations, which limits the availability of data, goal-conditioned RL can only yield goal-reaching behaviors, and unsupervised skill discovery methods, though general and principled, can present major challenges in terms of scalability, optimization, and offline learning.

In this work, we propose a general offline pre-training objective for foundation policies that capture diverse, optimal "long-horizon" behaviors from unlabeled data to facilitate downstream task learning. Our main idea is to discover the *temporal structure* of states through offline data, and to represent this structure in such a way that we can quickly and accurately obtain optimal policies for any arbitrary new tasks from relatively concise "prompts" (*e.g.*, a small number of states annotated with rewards, target goals, etc.). We begin by learning a geometric abstraction of the dataset, where dis-

[1] University of California, Berkeley. Correspondence to: Seohong Park <seohong@berkeley.edu>.

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



Distance-preserving mapping $d^*(s_1, s_2) = \|\phi(s_1) - \phi(s_2)\|$

Directional state-spanning policies $r = \langle \phi(s') - \phi(s), z \rangle$

Figure 1. **Illustration of HILPs.** (*left*) We first train a *distance-preserving* mapping $\phi : \mathcal{S} \to \mathcal{Z}$ that maps temporally similar states to spatially similar latent states (d^* denotes the temporal distance). (*right*) We then train a latent-conditioned policy $\pi(a \mid s, z)$, which we call a Hilbert foundation policy, that *spans* that latent space with directional movements. This policy captures diverse long-horizon behaviors from unlabeled data, which can be directly used to solve a variety of downstream tasks efficiently, even in a zero-shot manner.

tances between representations of states correspond to their long-horizon global relationships. Specifically, we train a representation $\phi : \mathcal{S} \to \mathcal{Z}$ that maps the state space $\mathcal{S}$ into a *Hilbert* space $\mathcal{Z}$ (*i.e.*, a metric space with a well-defined *inner product*) such that

$$d^*(s, g) = \|\phi(s) - \phi(g)\| \tag{1}$$

holds for every $s, g \in \mathcal{S}$, where d^* denotes the temporal distance (*i.e.*, the minimum number of time steps needed for an optimal policy to transition between them). Then, we train a latent-conditioned policy $\pi(a \mid s, z)$ that *spans* the learned latent space using offline RL, with the following "directional" intrinsic reward based on the inner product:

$$r(s, z, s') = \langle \phi(s') - \phi(s), z \rangle. \tag{2}$$

Intuitively, by learning to move in every possible *direction* specified by a unit vector $z \in \mathcal{Z}$, the policy learns diverse long-horizon behaviors that optimally *span* the latent space as well as the state space (Figure 1).

The resulting multi-task policy $\pi(a \mid s, z)$ has a number of attractive properties. **First**, it captures a variety of diverse behaviors, or *skills*, from offline data. These behaviors can be hierarchically combined or fine-tuned to solve downstream tasks efficiently. **Second**, we can train this policy with offline RL (as opposed to BC), and thus can utilize suboptimal data, unlike BC-based policy pre-training methods. Moreover, the learned behaviors are provably optimal for solving goal-reaching tasks (under some assumptions), which makes our method subsume *goal-conditioned RL* as a special case, while providing for much more diverse behaviors. **Third**, thanks to our inner product parameterization, this multi-task policy provides a very efficient way to adapt to any arbitrary reward function, enabling *zero-shot RL*. **Fourth**, this pre-training procedure yields a highly structured Hilbert representation ϕ, which enables efficient test-time *planning* without training an additional model. Given the above versatility of our multi-task policy $\pi(a \mid s, z)$, we call it a **Hilbert foundation policy** (**HILP**).

Our main contribution of this work is to introduce HILPs, a new objective to pre-train diverse policies from offline data that can be adapted efficiently to various downstream tasks. Through our experiments, we empirically demonstrate that HILPs capture diverse behaviors that can be directly used to solve goal-conditioned RL and zero-shot RL without any additional training. We also show that our single general HILP framework often outperforms previous offline policy pre-training methods specifically designed for individual problem statements (*e.g.*, zero-shot RL, goal-conditioned RL, and hierarchical RL) on seven robotic locomotion and manipulation environments.

2. Related Work

Representation learning for sequential decision making. HILPs are based on a distance-preserving state representation ϕ, and are related to prior work in representation learning for RL and control. Previous methods have proposed various representation learning objectives based on visual feature learning (Shah & Kumar, 2021; Parisi et al., 2022; Xiao et al., 2022), contrastive learning (Sermanet et al., 2018; Nair et al., 2022), dynamics modeling (Seo et al., 2022; Lamb et al., 2022; Brandfonbrener et al., 2023), and goal-conditioned RL (Ma et al., 2023; Ghosh et al., 2023). In particular, several previous methods (Sermanet et al., 2018; Nair et al., 2022; Ma et al., 2023) employ the same ℓ^2 parameterization as HILPs to obtain temporal distance-based representations. However, unlike these prior works, which focus only on pre-training representations, our focus is on unsupervised pre-training of diverse *behaviors* (*i.e.*, foundation *policies*). This enables solving downstream tasks in a *zero-shot* manner by simply "prompting" the foundation policy.

Unsupervised policy pre-training. Prior works have proposed various unsupervised (*i.e.*, task-agnostic) objectives to pre-train diverse policies that can be used to accelerate downstream task learning. Online unsupervised RL methods pre-train policies with exploration (Pathak et al., 2017; 2019; Mendonca et al., 2021; Rajeswar et al., 2023) or skill discovery objectives (Gregor et al., 2016; Eysenbach et al., 2019; Sharma et al., 2020; Klissarov & Machado, 2023; Park et al., 2024). Unlike these works, we focus on the offline setting, where we aim to learn diverse policies purely from an offline dataset of unlabeled trajectories.

For offline policy pre-training, behavioral cloning (Pertsch et al., 2020; Ajay et al., 2021; Padalkar et al., 2024) and trajectory modeling (Chen et al., 2021a; Janner et al., 2021; Reed et al., 2022; Liu et al., 2022; Wu et al., 2023) approaches train foundation policies via supervised learning. However, these supervised learning-based methods share a limitation in that they assume demonstrations of high quality. Among offline RL-based policy pre-training approaches, offline goal-conditioned RL methods train goal-conditioned policies to reach any goal state from any other state (Eysenbach et al., 2022; Ma et al., 2022; Yang et al., 2023; Wang et al., 2023; Park et al., 2023). These methods, however, only learn goal-reaching behaviors and thus have limited behavioral diversity. In contrast, our method subsumes goal-conditioned RL as a special case while learning much more diverse behaviors, which can be used to maximize arbitrary reward functions in a zero-shot manner.

Another line of work pre-trains multi-task policies with offline RL based on successor features and other generalized value function designs (Dayan, 1993; Barreto et al., 2017; Borsa et al., 2019; Ma et al., 2020; Touati & Ollivier, 2021; Touati et al., 2023; Chen et al., 2023; Hu et al., 2023). Our work is closely related to these approaches as our inner product reward function resembles the linear structure in the successor feature framework. Any successor feature or generalized value function approach operating as an unsupervised pre-training method needs to make a key decision about which tasks to learn, since any finite task representation needs to trade off some tasks for others. Some prior methods make this decision simply based on random reward functions or random features (Zheng et al., 2021; Farebrother et al., 2023; Chen et al., 2023; Hu et al., 2023), while the others employ hand-crafted state features (Barreto et al., 2017; Borsa et al., 2019), off-the-shelf representation learning (*e.g.*, autoencoders), or low-rank approximation of optimal successor representations (Touati et al., 2023), to specify and prioritize which tasks to capture. In this work, we prioritize *long-term temporal structure*, training state representation ϕ to capture the temporal distances between states by geometrically abstracting the state space. In our experiments, we show that this leads to significantly better performance and scalability than prior successor feature- or generalized value function-based offline unsupervised RL methods.

Finally, our method is closely related to METRA (Park et al., 2024), a recently proposed online unsupervised skill discovery method. METRA also learns to span a temporal distance-based abstraction of the state space based on a similar directional objective with online rollouts. However, METRA cannot be directly applied to the offline setting as it assumes on-policy rollouts to train the representation ϕ. Unlike METRA, we decouple representation learning and policy learning to enable *offline* policy pre-training from unlabeled data.

3. Preliminaries and Problem Setting

Markov decision process (MDP). An MDP $\mathcal{M}$ is defined as a tuple $(\mathcal{S}, \mathcal{A}, r, \mu, p)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \to \mathbb{R}$ is the reward function, $\mu : \Delta(\mathcal{S})$ is the initial state distribution, and $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics kernel. In this work, we assume a deterministic MDP unless otherwise stated, following prior works in offline RL and representation learning (Ma et al., 2023; Ghosh et al., 2023; Wang et al., 2023).

Hilbert space. A Hilbert space $\mathcal{Z}$ is a complete vector space equipped with an inner product $\langle x, y\rangle$, the induced norm $\|x\| = \sqrt{\langle x, x\rangle}$, and the induced metric $d(x, y) = \|x - y\|$ for $x, y \in \mathcal{Z}$. **Intuitively**, a Hilbert space can roughly be thought of as a "stricter" version of a metric space, where there exists an *inner product* that is consistent with the metric. For example, a Euclidean space with the ℓ^1- or ℓ^∞-norm is a metric space but not a Hilbert space, whereas a Euclidean space with the ℓ^2-norm is a Hilbert space, as $\|x\|_2 = \sqrt{x^\top x}$ for $x \in \mathbb{R}^D$. In our experiments, we will mainly employ Euclidean spaces (with the ℓ^2-norm) as Hilbert spaces, but the theorems in the paper can be applied to any arbitrary real Hilbert space.

Problem setting. We assume that we are given unlabeled trajectory data $\mathcal{D}$, which consists of state-action trajectories $\tau = (s_0, a_0, s_1, \dots, s_T)$. We do not make any assumptions about the quality of these unlabeled trajectories: they can be optimal for some unknown tasks, suboptimal, completely random, or even a mixture of these.

Our goal is to pre-train a versatile latent-conditioned policy $\pi(a \mid s, z)$, where $z \in \mathcal{Z}$ denotes a latent vector (which we call a *task* or a *skill*), purely from the unlabeled offline data $\mathcal{D}$, without online interactions. For the evaluation of the pre-trained policy, we consider three evaluation settings. (**1**) **Zero-shot RL**[1]: Given a reward function $r(s)$, we aim to find the best latent vector z that maximizes the reward function, without additional training. (**2**) **Offline goal-conditioned RL**: Given a target goal $g \in \mathcal{S}$, we aim to find the best latent vector z of the policy $\pi(a \mid s, z)$ that leads to the goal as quickly as possible, without additional training. The goal is specified at test time. (**3**) **Hierarchical RL**: Given a reward function $r(s)$, we train a high-level policy $\pi^h(z \mid s)$ that sequentially combines pre-trained skills to maximize the reward function using offline RL. In all three settings, we only allow online interaction with the environ-

[1] We use the term "zero-shot RL" following Touati et al. (2023).

ment during the final evaluation, and assume that the state space and environment dynamics remain the same at evaluation time.

4. Hilbert Foundation Policies (HILPs)

We now introduce our offline pre-training scheme for foundation policies that capture diverse long-horizon behaviors from unlabeled data. Our main strategy is to first learn a geometric state abstraction that preserves the temporal structure of the MDP (Section 4.1), and then to span the abstracted latent space with skills that correspond to directional movements in this space (Section 4.2).

4.1. Hilbert Representations

We begin by training a representation function $\phi : \mathcal{S} \to \mathcal{Z}$ that abstracts the state space into a latent space $\mathcal{Z}$. We have two desiderata for ϕ. First, ϕ should map *temporally* similar states to *spatially* similar latent states, so that it can abstract the dataset states while preserving their long-horizon global relationships. Second, ϕ should be well-structured such that it provides a way to train a versatile multi-task policy $\pi(a \mid s, z)$ that can be easily "prompted" to solve a variety of downstream tasks.

Based on these desiderata, we set $\mathcal{Z}$ to be a Hilbert space, which not only provides a proper *metric* to quantify the similarity between latent states, but also provides an *inner product* that enables several principled ways to prompt the policy, which we will describe in Section 5. In the latent space $\mathcal{Z}$, our desiderata for the representation function ϕ can be formalized as follows:

$$d^*(s, g) = \|\phi(s) - \phi(g)\|, \tag{3}$$

where d^* denotes the optimal temporal distance from s to g, *i.e.*, the minimum number of time steps to reach g from s. We refer to a representation ϕ that satisfies Equation (3) as a **Hilbert representation**. Intuitively, ϕ is a distance-preserving embedding function (*i.e.*, an *isometry* to a Hilbert space), where distances in the latent space correspond to the temporal distances in the original MDP. This enables ϕ to abstract the state space while maintaining the global relationships between states.

To train ϕ, we leverage the equivalence between temporal distances and optimal goal-conditioned value functions (Kaelbling, 1993; Wang et al., 2023), $V^*(s, g) = -d^*(s, g)$. Here, $V^*(s, g)$ is the optimal goal-conditioned value function for the state s and the goal g, *i.e.*, the maximum possible return (*i.e.*, the sum of rewards) for the reward function given by $r(s, g) = -\mathbb{1}(s \neq g)$ and the episode termination condition given by $\mathbb{1}(s = g)$. Based on this connection to goal-conditioned RL, we can train ϕ with any off-the-shelf offline goal-conditioned value learning algorithm (Park et al., 2023; Ma et al., 2023; Wang et al., 2023) with the value function being parameterized as

$$V(s, g) = -\|\phi(s) - \phi(g)\|. \tag{4}$$

Implementation. For practical implementation, we set $\mathcal{Z}$ to be the Euclidean space $\mathbb{R}^D$ with the ℓ^2-norm. To train $V(s, g)$ from offline data, we opt to employ the IQL-based (Kostrikov et al., 2022) goal-conditioned value learning scheme introduced by Park et al. (2023). This method minimizes the following temporal difference loss:

$$\mathbb{E}_{s,s',g}[\ell^2_\tau(-\mathbb{1}(s \neq g) + \gamma \bar{V}(s', g) - V(s, g))], \tag{5}$$

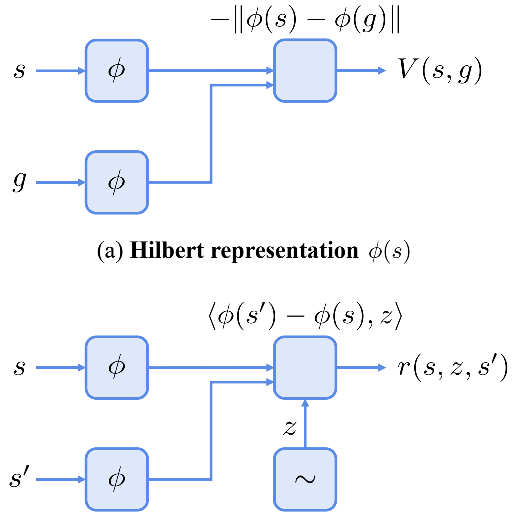
where γ denotes a discount factor, $\bar{V}$ denotes the target network (Mnih et al., 2013), and $\ell^2_\tau(x) = |\tau - \mathbb{1}(x < 0)|x^2$ denotes the expectile loss (Newey & Powell, 1987), an asymmetric ℓ^2 loss that approximates the $\max$ operator in the Bellman backup (Kostrikov et al., 2022). States, next states, and goals (*i.e.*, (s, s', g)) are sampled from the replay buffer with a hindsight relabeling strategy (Andrychowicz et al., 2017; Park et al., 2023), and episodes terminate upon goal reaching (see Appendix D for details). With the value function parameterization in Equation (4), our final objective for Hilbert representations ϕ becomes

$$\mathbb{E}[\ell^2_\tau(-\mathbb{1}(s \neq g) - \gamma\|\bar{\phi}(s') - \bar{\phi}(g)\| + \|\phi(s) - \phi(g)\|)], \tag{6}$$

where $\bar{\phi}$ denotes the target representation network.

Remarks. There exist three potential limitations with Equation (6). First, the distance metric in $\mathcal{Z}$ is symmetric, whereas temporal distances might be asymmetric. Second, even when the environment dynamics are symmetric, there might not exist an exact isometry between the MDP and the Hilbert space (Indyk et al., 2017; Pitis et al., 2020). Third, we use a discount factor γ in Equation (6), but temporal distances are undiscounted. In this regard, our objective might better be viewed as finding the best discounted Hilbert *approximation* of the MDP, rather than learning an exact Hilbert abstraction. While this might not be ideal in highly asymmetric environments, we note that we will *not* directly use the potentially erroneous parameterized value function $V(s, g)$ for policy learning. We will instead only take the representation ϕ, defining a new reward function as well as a new value function to pre-train an unsupervised latent-conditioned policy, as we will describe in Section 4.2. After all, our goal is to train a representation ϕ that captures the long-term temporal structure of the MDP, and we empirically found that even approximate Hilbert representations lead to diverse useful behaviors that can be directly used to solve downstream tasks in our experiments (Section 6). We refer to Appendix C for further discussions.

We also note that the parameterization in Equation (4) has been employed in several prior works in robotic representation learning (Sermanet et al., 2018; Nair et al., 2022; Ma et al., 2023). However, unlike these works, which mainly use ϕ only as a visual feature extractor, we pre-train a foun-



(a) **Hilbert representation** $\phi(s)$

(b) **Hilbert foundation policy** $\pi(a \mid s, z)$

Figure 2. **Diagram of HILPs.** (*a*) We train a Hilbert representation $\phi(s)$ using a goal-conditioned value learning objective with the value function parameterized as $V(s, g) = -\|\phi(s) - \phi(g)\|$. (*b*) We train a Hilbert foundation policy $\pi(a \mid s, z)$ using the intrinsic reward function $r(s, z, s')$ defined as the inner product between $\phi(s') - \phi(s)$ and a randomly sampled unit vector z.

dation *policy* with an intrinsic reward function based on an inner product involving ϕ.

4.2. Unsupervised Policy Training

After obtaining a Hilbert representation ϕ, our next step is to learn diverse skills that *span* the latent space $\mathcal{Z}$ with offline RL. Since a Hilbert space provides an inner product, we can train a state-spanning latent-conditioned policy $\pi(a \mid s, z)$ with the following inner product-based reward function:

$$r(s, z, s') = \langle\phi(s') - \phi(s), z\rangle, \quad (7)$$

where z is sampled uniformly from the set of unit vectors in $\mathcal{Z}$, $\{z \in \mathcal{Z} : \|z\| = 1\}$.

Since the latent-conditioned policy must maximize the reward in Equation (7) for all randomly sampled unit vectors z, the optimal set of skills represented by $\pi(a \mid s, z)$ should be able to travel as far as possible in every possible latent space direction. Consequently, we obtain a set of policies that optimally span the latent space, capturing diverse behaviors from the unlabeled dataset $\mathcal{D}$. We call the resulting policy $\pi(a \mid s, z)$ a **Hilbert foundation policy** (**HILP**). In Section 5, we will discuss why HILPs are useful for solving downstream tasks in a variety of scenarios.

Implementation. To train a HILP, we employ a standard off-the-shelf offline RL algorithm, such as IQL (Kostrikov et al., 2022), with the intrinsic reward defined as Equation (7). Latent vectors z are sampled from the uniform distribution over $\mathbb{S}^{D-1} = \{z \in \mathbb{R}^D : \|z\| = 1\}$. In practice, we also consider another variant of the reward function defined as $r(s, z, s') = \langle\phi(s) - \bar{\phi}, z\rangle$, where $\bar{\phi}$ is defined as $\mathbb{E}_{s\sim\mathcal{D}}[\phi(s)]$. This is also a state-spanning directional reward function, but the displacement is defined as the difference from the center instead of the difference between two adjacent states. We found this variant to perform better in the zero-shot RL setting in our experiments (Section 6.1). We illustrate the architecture of HILPs in Figure 2, summarize the full training procedure in Algorithm 1, and refer to Appendix D for the full experimental details.

Algorithm 1 Hilbert Foundation Policies (HILPs)

```
Initialize Hilbert representation $\phi(s)$, policy $\pi(a \mid s, z)$
while not converged do
    Sample $(s, s', g) \sim \mathcal{D}$
    Train $\phi(s)$ by minimizing Equation (6)
end while
while not converged do
    Sample $(s, a, s') \sim \mathcal{D}$
    Sample $z \sim \mathbb{S}^{D-1}$
    Compute intrinsic reward $r(s, z, s')$
    Train $\pi(a \mid s, z)$ with any off-the-shelf offline RL algorithm
end while
```

5. Why are HILPs useful?

HILPs, in combination with structured representations $\phi(s)$, provide a number of ways to solve downstream tasks efficiently, often even in a zero-shot fashion. In this section, we describe three test-time "prompting" strategies for HILPs for zero-shot RL (Section 5.1), offline goal-conditioned RL (Section 5.2), and test-time planning (Section 5.3).

5.1. Zero-Shot RL

First, HILPs can be used to quickly adapt to any arbitrary reward functions at test time. Our key observation is that the operand in our inner product reward function (Equation (7)) $\tilde{\phi}(s, a, s') := \phi(s') - \phi(s)$ can be viewed as a *cumulant* (Sutton & Barto, 2005) in the successor feature framework (Dayan, 1993; Barreto et al., 2017). This connection to successor features enables *zero-shot RL* (Touati et al., 2023): given an arbitrary reward function $r(s, a, s')$ at test time, we can find the latent vector z for the policy $\pi(a \mid s, z)$ that best solves the task via simple linear regression, without any additional training. Specifically, the optimal (unnormalized) latent vector z^* for the reward function $r(s, a, s')$ is given as

$$z^* = \underset{z\in\mathcal{Z}}{\arg\min}\, \mathbb{E}_{\mathcal{D}}\left[\left(r(s, a, s') - \langle\tilde{\phi}(s, a, s'), z\rangle\right)^2\right], \quad (8)$$

where (s, a, s') tuples are sampled from the unlabeled dataset $\mathcal{D}$. If $\mathcal{Z}$ is a Euclidean space, we have the closed-form solution $z^* = \mathbb{E}_{\mathcal{D}}[\tilde{\phi}\tilde{\phi}^\top]^{-1}\mathbb{E}_{\mathcal{D}}[r(s, a, s')\tilde{\phi}]$, where $\tilde{\phi}$ denotes $\tilde{\phi}(s, a, s')$.

In practice, we sample a small number of (s, a, s') tuples from the dataset, compute the optimal latent z^* with respect

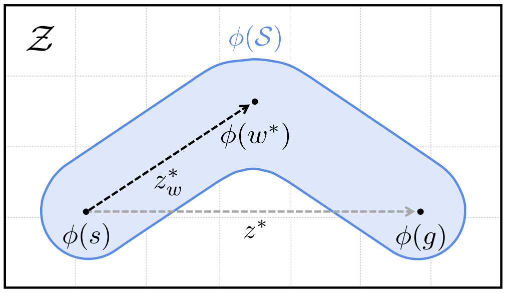


Figure 3. **Test-time midpoint planning.** In the presence of embedding errors, the direction toward the midpoint subgoal w^* can be more accurate than the direction toward the goal g.

to the test-time reward function using the above formula, and execute the corresponding policy $\pi(a \mid s, z^*)$ to perform zero-shot RL.

5.2. Offline Goal-Conditioned RL

For goal-reaching tasks, HILPs provide an even simpler way to solve *goal-conditioned RL* in a zero-shot manner, where the aim is to reach a goal state $g \in \mathcal{S}$ within the minimum number of steps. Intuitively, a Hilbert representation ϕ is learned in a way that the distance between $\phi(s)$ and $\phi(g)$ in the latent space corresponds to the optimal temporal distance $d^*(s, g)$. Hence, to reach the goal g, the agent just needs to move in the latent direction of $\phi(g) - \phi(s)$ so that it can monotonically decrease the distance toward $\phi(g)$ in the Hilbert space, which in turn decreases the temporal distances toward g in the original MDP. Since the HILP $\pi(a \mid s, z)$ is pre-trained to move in every possible direction, we can just set z to

$$z^* := \frac{\phi(g) - \phi(s)}{\|\phi(g) - \phi(s)\|}. \tag{9}$$

We theoretically prove this intuition as follows:

Theorem 5.1 (Directional movements in the latent space are optimal for goal reaching)**.** *If embedding errors are bounded as* $\sup_{s,g\in\mathcal{S}} |d^*(s, g) - \|\phi(s) - \phi(g)\|| \le \varepsilon_e$, *directional movement errors are bounded as* $\sup_{s,g\in\mathcal{S}} \|z'^*(s, g) - \hat{z}'(s, g)\| \le \varepsilon_d$, *and* $4\varepsilon_e + \varepsilon_d < 1$, *then* $\pi(a \mid s, z^*)$ *is an optimal goal-reaching policy.*

The formal definitions of $z'^*(s, g)$ and $z'(s, g)$ and the proof are provided in Appendix C. Intuitively, Theorem 5.1 states that if the embedding errors of ϕ are small enough, directional movements in the latent space are optimal for solving goal-reaching tasks. We refer to Appendix C for further discussion including limitations.

5.3. Test-Time Planning

Another benefit of our HILP framework is that it naturally enables efficient *test-time planning* for goal-conditioned RL based on the structured state representation $\phi(s)$. While Section 5.2 introduces a simple method to query the policy $\pi(a \mid s, z)$ to reach a goal g (Equation (9)), this might not be perfect in practice due to potential embedding errors in ϕ and thus may potentially lead to suboptimal goal-reaching performance, as stated in Theorem 5.1. In particular, when the image of the mapping ϕ is distorted (Figure 3), the straight line from the current latent state $\phi(s)$ to the latent goal $\phi(g)$ in $\mathcal{Z}$ might not represent a feasible path between s and g in the MDP, potentially making the agent struggle to reach the goal.

Algorithm 2 Test-time planning with HILPs

```
Input: unlabeled offline dataset $\mathcal{D}$, Hilbert representation
    $\phi(s)$, HILP $\pi(a \mid s, z)$, goal $g$
Sample $w_1, w_2, \ldots, w_N \sim \mathcal{D}$
Pre-compute $\phi(w_1), \phi(w_2), \ldots, \phi(w_N)$
Observe $s_0 \sim \mu(s_0)$
for $t \leftarrow 0$ to $T - 1$ do
    $s, u \leftarrow s_t, g$
    for $i \leftarrow 1$ to (# recursions) do
        $w^* \leftarrow \arg\min_{w\in\{w_1\ldots,w_N\}} \max(\|\phi(s) - \phi(w)\|,$
            $\|\phi(w) - \phi(u)\|)$
        $u \leftarrow w^*$
    end for
    Compute $z_w^*$ with Equation (11)
    Sample $a_t \sim \pi(a_t \mid s_t, z_w^*)$
    Observe $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
end for
```

To resolve this issue, we introduce a way to further *refine* the latent vector z to mitigate such approximation errors by finding the optimal *feasible subgoal* between s and g. Specifically, we aim to find a midpoint state that is equidistant from both s and g and still makes progress towards the goal. This can be formalized as the following minimax objective (Chane-Sane et al., 2021):

$$\begin{aligned} w^* &:= \arg\min_{w\in\mathcal{S}} [\max(d^*(s, w), d^*(w, g))] \\ &\approx \arg\min_{w\in\mathcal{S}} [\max(\|\phi(s) - \phi(w)\|, \|\phi(w) - \phi(g)\|)]\,. \end{aligned} \tag{10}$$

After finding w^*, we refine the latent vector z for the policy $\pi(a \mid s, z)$ as follows:

$$z_w^* = \frac{\phi(w^*) - \phi(s)}{\|\phi(w^*) - \phi(s)\|}. \tag{11}$$

As the distance between s and w^* is smaller than that between s and g, z_w^* is likely more accurate than z^* in Equation (9). We illustrate this refinement procedure in Figure 3.

Thanks to our structured Hilbert representation ϕ, Equation (10) can be very efficiently computed in practice. At test time, we first randomly sample a small number (N) of states from the dataset $\mathcal{D}$, and pre-compute their Hilbert representations. Then, at every time step, we approximate w^* by finding the $\arg\min$ of Equation (10) over the N samples using the pre-computed representations. Since we can approximate d^* by measuring the distances between repre-

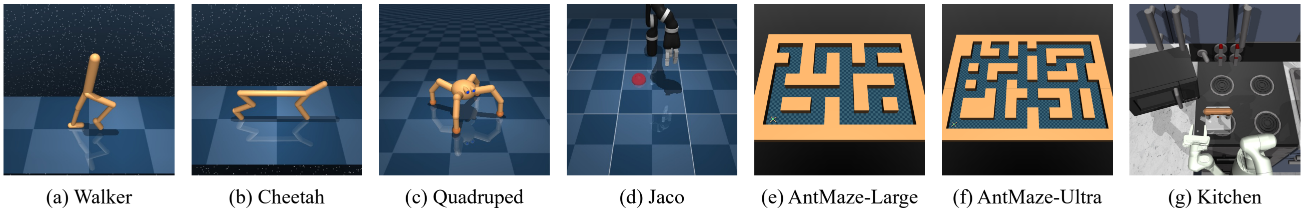


Figure 4. **Environments.** We evaluate HILPs on seven robotic locomotion and manipulation environments.

sentations, this procedure does not require any additional neural network queries.

To further refine the subgoal, we can recursively apply this midpoint planning procedure more than once. Namely, we can first find the midpoint w^* between s and g via Equation (10), and then find the midpoint between s and w^* (*i.e.*, the $1/4$-point between s and g) with the same minimax objective, and so on. Again, thanks to our Hilbert representation, this recursive planning can also be done only with elementary algebraic operations, without additionally querying neural networks. In our experiments, we empirically found that iterative refinements do improve performance on long-horizon tasks. We summarize the full test-time planning procedure in Algorithm 2 in Appendix D.

6. Experiments

In our experiments, we empirically evaluate the performance of HILPs on various types of downstream tasks. In particular, we consider three different experimental settings: zero-shot RL, offline goal-conditioned RL, and hierarchical RL (see Section 3 for the problem statements), in which we aim to answer the following questions: (1) Can HILPs be prompted to solve goal-conditioned and reward-based tasks in a zero-shot manner? (2) Can a single general HILP framework outperform prior state-of-the-art approaches specialized in each setting? (3) Does test-time planning improve performance? In our experiments, we use 8 random seeds unless otherwise stated, and report 95% bootstrap confidence intervals in plots and standard deviations in tables. Our code is available at this repository.

6.1. Zero-Shot RL

We first evaluate HILPs in the *zero-shot RL* setting, where the aim is to maximize an arbitrary reward function given at test time without additional training. For benchmarks, following Touati et al. (2023), we use the Unsupervised RL Benchmark (Laskin et al., 2021) and ExORL datasets (Yarats et al., 2022), which consist of trajectories collected by various unsupervised RL agents. We consider four environments (Walker, Cheetah, Quadruped, and Jaco) (Figure 4) and four datasets collected by APS (Liu & Abbeel, 2021a), APT (Liu & Abbeel, 2021b), Proto (Yarats et al., 2021), and RND (Burda et al., 2019) in each environment. These datasets correspond to the four most performant unsupervised RL algorithms in Figure 2 in the work by Yarats et al. (2022). In addition to these original state-based environments, we also employ their *pixel-based* variants with $64 \times 64 \times 3$-sized observation spaces to evaluate the scalability of the methods to complex observations. With these environments and datasets, we train HILPs and eight previous zero-shot RL methods in three different categories: (1) forward-backward (FB) representations (Touati et al., 2023), a state-of-the-art zero-shot RL method, (2) successor features (SFs) with six different feature learners (forward dynamics model (FDM), Laplacian (Lap) (Wu et al., 2019), autoencoder (AE), inverse dynamics model (IDM), random features, and contrastive learning (CL)), which includes the best feature learner (Laplacian) in Touati et al. (2023), and (3) goal-conditioned RL (GC-TD3). We use TD3 (Fujimoto et al., 2018) as the base offline RL algorithm to train these methods, as it is known to perform best in ExORL (Yarats et al., 2022). After finishing unsupervised policy training, we evaluate these methods on four test tasks in each environment (*e.g.*, Flip, Run, Stand, and Walk for Walker) with their own zero-shot adaption strategies. For HILPs, we use the zero-shot prompting scheme introduced in Section 5.1. In the Jaco domain, we additionally consider the goal-conditioned prompting scheme (Section 5.2) for HILPs (HILP-G), since it is a goal-oriented environment (however, we do *not* use this variant for overall performance aggregation). For FB and SF methods, we use the zero-shot schemes introduced by Touati et al. (2023) based on reward-weighted expectation or linear regression. For the goal-conditioned RL baseline (GC-TD3), we find the state with the highest reward value from the offline dataset and use it as the goal.

Figure 5 shows the overall and per-environment comparison results, where we use the interquartile mean (IQM) metric for overall aggregation, following the suggestion by Agarwal et al. (2021). The results suggest that HILPs achieve the best overall zero-shot RL performance among the nine methods, while achieving the best or near-best scores in each environment. Notably, HILPs achieve significantly better performance than GC-TD3, which indicates that HILPs capture more diverse behaviors than ordinary goal-reaching policies, and these diverse behaviors can be directly used to maximize a variety of reward functions at test time. Figure 6 shows the overall comparison results on pixel-based ExORL environments. Due to high computational costs, we

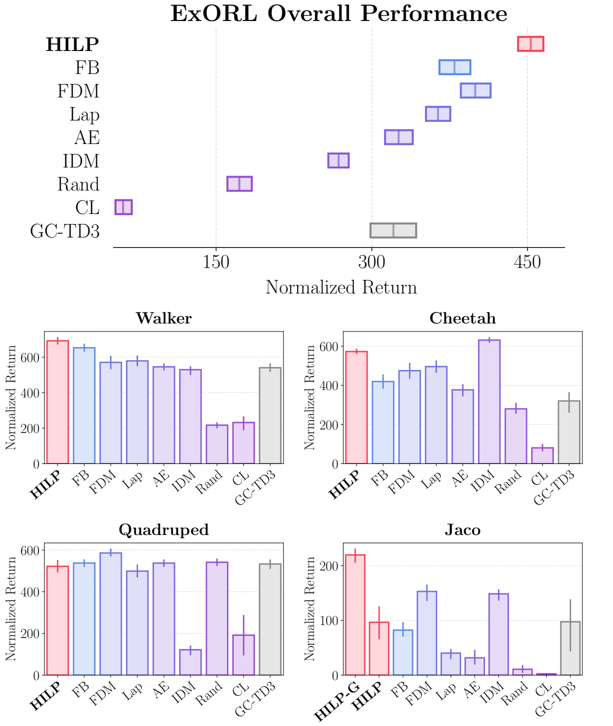


Figure 5. **Zero-shot RL performance.** HILP achieves the best zero-shot RL performance in the ExORL benchmark, outperforming previous state-of-the-art approaches. The overall results are aggregated over 4 environments, 4 tasks, 4 datasets, and 4 seeds (*i.e.*, 256 values in total).

compare HILPs with the two most performant baselines in Figure 5 (FB and FDM). The results indicate that HILPs achieve the best IQM in pixel-based environments as well, outperforming the previous best approaches. We refer to Appendix B for the full results.

6.2. Offline Goal-Conditioned RL

Next, we evaluate HILPs on *goal-reaching* tasks. For benchmarks, we consider the goal-conditioned variants of AntMaze and Kitchen tasks (Figure 4) from the D4RL suite (Fu et al., 2020; Park et al., 2023). In the AntMaze environment, we are given a dataset consisting of 1000 trajectories of a quadrupedal Ant agent navigating through a maze. We employ the two most challenging datasets with the largest maze ("antmaze-large-{diverse, play}") from the D4RL benchmark, and two even larger settings ("antmaze-ultra-{diverse, play}") introduced by Jiang et al. (2023) to further challenge the long-horizon reasoning ability of the agent. In the Kitchen manipulation environment (Gupta et al., 2019), we are given a dataset consisting of trajectories of a robotic arm manipulating different kitchen objects (*e.g.*, a kettle, a microwave, cabinets, etc.) in various orders. We employ two datasets ("kitchen-{partial, mixed}") from the D4RL benchmark. Additionally, we use *pixel-based* variants of these datasets ("visual-kitchen-{partial, mixed}") to evaluate the visual reasoning capacity of the agent, where the agent must learn to manipulate the robot arm purely from $64 \times 64 \times 3$ camera images.

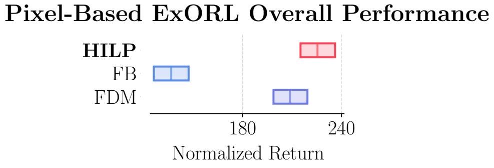


Figure 6. **Pixel-based zero-shot RL performance.** HILP exhibits the best performance in the *pixel-based* ExORL benchmark as well, outperforming the two most performant baselines among FB and SF methods. The results are aggregated over 4 environments, 4 tasks, 4 datasets, and 4 seeds (*i.e.*, 256 values in total).

In these environments, we compare HILPs against four general unsupervised policy pre-training methods as well as three methods that are specifically designed to solve goal-conditioned RL. For the former group, we consider the three best zero-shot RL methods from the previous section (FB, FDM, Lap) and random successor features (Rand). For goal-conditioned approaches, we consider goal-conditioned behavioral cloning (GC-BC) (Ding et al., 2019; Ghosh et al., 2021), goal-conditioned IQL (GC-IQL) (Kostrikov et al., 2022; Park et al., 2023), and goal-conditioned CQL (GC-CQL) (Kumar et al., 2020). To adapt HILPs to goal-conditioned tasks, we employ the zero-shot prompting scheme (Section 5.2) as well as the test-time midpoint planning scheme with three recursions (Section 5.3, "HILP-Plan"). For FB, we use the backward representation of the goal state to obtain the latent vector, and for SF methods, we perform linear regression with respect to the original reward function (*i.e.*, they use privileged reward information), as there is no clear way to adapt them to goal-conditioned tasks.

Table 1 shows the results, which suggest HILPs can solve challenging long-horizon goal-conditioned tasks in a zero-shot manner, and significantly outperform previous general unsupervised policy learning methods (FB and three SF methods). This is likely because our method prioritizes learning long-horizon behaviors that span the state space, unlike previous successor features or forward-backward methods. Table 1 also shows that HILPs with test-time planning often even outperform GCRL-dedicated methods (*e.g.*, GC-IQL). We note that this planning procedure can be done only with elementary algebraic operations based on pre-computed representations (Section 5.3), thanks to our structured Hilbert representations. To further study the effect of test-time planning, we compare the performances of HILPs with different numbers (0-3) of midpoint recursions. We report results in Figure 7, which shows that iterative refinement of prompts via our test-time planning approach improves performance consistently. We refer to Appendix B for further analyses, including an **ablation study**, **embedding error analysis**, and **latent space visualization**.

Table 1. **Offline goal-conditioned RL performance (8 seeds).** HILP achieves the best performance among general offline unsupervised policy learning methods, while being comparable to methods that are specifically designed to solve offline goal-conditioned RL. With our efficient test-time planning procedure based on Hilbert representations, HILPs often even outperform offline goal-conditioned RL methods.

	GCRL-dedicated methods			General unsupervised policy learning methods					
Dataset	GC-BC	GC-IQL	GC-CQL	FB	FDM	Lap	Rand	**HILP (ours)**	**HILP-Plan (ours)**
antmaze-large-diverse	15.0 ±9.3	56.0 ±6.0	36.2 ±19.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	46.0 ±12.7	**64.5** ±10.2
antmaze-large-play	12.0 ±5.9	**56.0** ±25.7	32.0 ±25.8	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	49.0 ±8.8	**58.8** ±11.2
antmaze-ultra-diverse	30.5 ±10.1	40.8 ±11.1	14.2 ±13.5	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	21.2 ±11.2	**59.2** ±12.7
antmaze-ultra-play	26.5 ±6.2	41.8 ±9.0	16.5 ±14.3	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	22.2 ±11.4	**50.8** ±9.6
kitchen-partial	52.7 ±11.0	56.4 ±8.4	31.2 ±16.6	0.2 ±0.7	39.2 ±4.2	41.2 ±9.3	44.6 ±8.9	**63.9** ±5.7	59.7 ±5.1
kitchen-mixed	**58.8** ±8.0	**59.5** ±3.8	15.7 ±17.6	0.7 ±1.9	42.9 ±7.1	40.9 ±3.0	46.6 ±2.3	55.5 ±9.5	51.9 ±8.3
visual-kitchen-partial	**63.2** ±3.7	**63.6** ±4.2	-	-	-	43.3 ±2.2	39.7 ±3.3	59.9 ±4.0	52.1 ±5.3
visual-kitchen-mixed	**57.5** ±4.2	52.9 ±4.7	-	-	-	50.9 ±3.6	49.6 ±5.7	**55.9** ±9.7	54.1 ±13.4

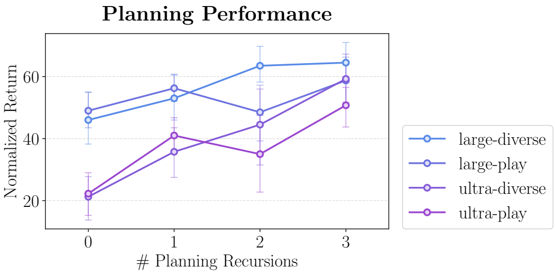


Figure 7. **Planning performance in AntMaze (8 seeds).** Test-time midpoint planning (Section 5.3) improves performance as the number of recursive refinements increases.

Table 2. **Hierarchical RL performance (8 seeds).** HILP outperforms OPAL, a previous VAE-based offline skill learning method.

Dataset	OPAL	**HILP (ours)**
antmaze-large-diverse	**59.2** ±12.5	56.0 ±9.4
antmaze-large-play	**58.0** ±9.6	**58.0** ±11.1
antmaze-ultra-diverse	21.0 ±10.1	**43.8** ±8.6
antmaze-ultra-play	22.8 ±7.6	**47.8** ±13.2
kitchen-partial	**54.8** ±13.3	**54.1** ±3.9
kitchen-mixed	**58.2** ±7.5	48.2 ±10.1
Average	45.7	**51.3**

6.3. Hierarchical RL

Finally, to evaluate the effectiveness of skills learned by HILPs compared to previous BC-based skill learning methods, we compare HILPs with OPAL (Ajay et al., 2021), a previous offline skill extraction method based on a trajectory variational autoencoder (VAE). For benchmarks, we use the AntMaze and Kitchen datasets from D4RL (Fu et al., 2020). On top of skills learned by HILPs or OPAL in these environments, we train a high-level skill policy $\pi^h(z \mid s)$ that uses learned skills as high-level actions with IQL (Kostrikov et al., 2022). We use the same IQL-based high-level policy training scheme to ensure a fair comparison. Table 2 shows the results, which suggest that HILPs achieve better performances than OPAL, especially in the most challenging AntMaze-Ultra tasks, likely because HILPs capture optimal, long-horizon behaviors. Additionally, we note that, unlike OPAL or similar VAE-based approaches, HILPs provide multiple zero-shot prompting schemes to query the learned policy to solve downstream tasks without a separate high-level policy, as shown in previous sections.

7. Conclusion

In this work, we introduced Hilbert foundation policies (HILPs), a general-purpose offline policy pre-training scheme based on the idea of spanning a structured latent space that captures the temporal structure of the MDP. We showed that structured Hilbert representations enable zero-shot prompting schemes for zero-shot RL and goal-conditioned RL as well as test-time planning to adapt pre-trained HILPs to downstream tasks. Through our experiments, we demonstrated that our single HILP framework often outperforms previous specialized methods for zero-shot RL, goal-conditioned RL, and hierarchical RL.

Final remarks. Offline unsupervised policy pre-training is all about determining and prioritizing the right behaviors to capture from offline data. Prior works have proposed simply cloning dataset actions, capturing goal-reaching behaviors, or learning to maximize linear combinations of state features. In this work, we proposed capturing long-horizon, state-spanning behaviors. This is desirable because such global behaviors are usually harder to learn than local behaviors, and thus are worth capturing during pre-training. However, one may still wonder: "What if the environment is stochastic or partially observable, where an isometric embedding does not exist?" "Are directional latent movements sufficient?" "Is zero-shot task adaptation (without fine-tuning) the right way to *use* learned behaviors?" These are all important and valuable questions, and finding satisfying answers to these questions would lead to exciting future work. For example, we may learn a *local isometry* (Lee, 2018) instead of a global isometry to handle partially observable environments, learn more diverse latent movements to enhance expressivity, or explore fine-tuning or few-shot learning for better task adaptation. We further discuss limitations and future work in Appendix A. Nonetheless, we hope that this work represents a step toward ideal offline unsupervised policy pre-training.

Impact Statement

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

Acknowledgments

We would like to thank Oleg Rybkin for an informative discussion, and Qiyang Li and Yifei Zhou for helpful feedback on earlier drafts of this work. This work was partly supported by the Korea Foundation for Advanced Studies (KFAS), Berkeley Chancellor's Fellowship, ONR N00014-21-1-2838, AFOSR FA9550-22-1-0273, and Qualcomm. This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at UC Berkeley.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 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. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Van Hasselt, H., and Silver, D. Successor features for transfer in reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D. J., Munos, R., Hasselt, H. V., Silver, D., and Schaul, T. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

Brandfonbrener, D., Nachum, O., and Bruna, J. Inverse dynamics pretraining learns good representations for multitask imitation. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

Chane-Sane, E., Schmid, C., and Laptev, I. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning (ICML)*, 2021.

Chebotar, Y., Hausman, K., Lu, Y., Xiao, T., Kalashnikov, D., Varley, J., Irpan, A., Eysenbach, B., Julian, R. C., Finn, C., and Levine, S. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In *International Conference on Machine Learning (ICML)*, 2021.

Chen, B., Zhu, C., Agrawal, P., Zhang, K., and Gupta, A. Self-supervised reinforcement learning that transfers using random features. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems (NeurIPS)*, 2021a.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D. W., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Babuschkin, I., Balaji, S. A., Jain, S., Carr, A., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M. M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021b.

Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993.

Ding, Y., Florensa, C., Phielipp, M., and Abbeel, P. Goal-conditioned imitation learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

Eysenbach, B., Zhang, T., Salakhutdinov, R., and Levine, S. Contrastive learning as goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Farebrother, J., Greaves, J., Agarwal, R., Lan, C. L., Goroshin, R., Castro, P. S., and Bellemare, M. G. Proto-value networks: Scaling representation learning with auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Geng, X. Jaxcql: a simple implementation of sac and cql in jax, 2022. URL https://github.com/young-geng/JaxCQL.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Ghosh, D., Bhateja, C., and Levine, S. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning (ICML)*, 2023.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *ArXiv*, abs/1611.07507, 2016.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Hu, H., Yang, Y., Ye, J., Mai, Z., and Zhang, C. Unsupervised behavior extraction via random intent priors. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Indyk, P., Matoušek, J., and Sidiropoulos, A. Low-distortion embeddings of finite metric spaces. In *Handbook of discrete and computational geometry*, pp. 211–231. Chapman and Hall/CRC, 2017.

Janner, M., Li, Q., and Levine, S. Reinforcement learning as one big sequence modeling problem. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktaschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. In *International Conference on Learning Representations (ICLR)*, 2023.

Kaelbling, L. P. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. B. Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Klissarov, M. and Machado, M. C. Deep laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning (ICML)*, 2023.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Lamb, A., Islam, R., Efroni, Y., Didolkar, A. R., Misra, D., Foster, D. J., Molu, L. P., Chari, R., Krishnamurthy, A., and Langford, J. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research (TMLR)*, 2022.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Lee, J. M. *Introduction to Riemannian manifolds*. Springer, 2018.

Liu, F., Liu, H., Grover, A., and Abbeel, P. Masked autoencoding for scalable and generalizable decision making. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Liu, H. and Abbeel, P. APS: Active pretraining with successor features. In *International Conference on Machine Learning (ICML)*, 2021a.

Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. In *Neural Information Processing Systems (NeurIPS)*, 2021b.

Ma, C., Ashley, D. R., Wen, J., and Bengio, Y. Universal successor features for transfer reinforcement learning. *ArXiv*, abs/2001.04025, 2020.

Ma, Y. J., Yan, J., Jayaraman, D., and Bastani, O. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Ma, Y. J., Sodhani, S., Jayaraman, D., Bastani, O., Kumar, V., and Zhang, A. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023.

Machado, M. C., Bellemare, M. G., and Bowling, M. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., and Pathak, D. Discovering and achieving goals via world models. In *Neural Information Processing Systems (NeurIPS)*, 2021.

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

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

Nakamoto, M., Zhai, Y., Singh, A., Mark, M. S., Ma, Y., Finn, C., Kumar, A., and Levine, S. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Newey, W. and Powell, J. L. Asymmetric least squares estimation and testing. *Econometrica*, 55:819–847, 1987.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L. E., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. J. Training language models to follow instructions with human feedback. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., Raffin, A., Wahid, A., Burgess-Limerick, B., Kim, B., Schölkopf, B., Ichter, B., Lu, C., Xu, C., Finn, C., Xu, C., Chi, C., Huang, C., Chan, C., Pan, C., Fu, C., Devin, C., Driess, D., Pathak, D., Shah, D., Büchler, D., Kalashnikov, D., Sadigh, D., Johns, E., Ceola, F., Xia, F., Stulp, F., Zhou, G., Sukhatme, G. S., Salhotra, G., Yan, G., Schiavi, G., Su, H., Fang, H., Shi, H., Amor, H. B., Christensen, H. I., Furuta, H., Walke, H., Fang, H., Mordatch, I., Radosavovic, I., Leal, I., Liang, J., Kim, J., Schneider, J., Hsu, J., Bohg, J., Bingham, J., Wu, J., Wu, J., Luo, J., Gu, J., Tan, J., Oh, J., Malik, J., Tompson, J., Yang, J., Lim, J. J., Silvério, J., Han, J., Rao, K., Pertsch, K., Hausman, K., Go, K., Gopalakrishnan, K., Goldberg, K., Byrne, K., Oslund, K., Kawaharazuka, K., Zhang, K., Majd, K., Rana, K., Srinivasan, K. P., Chen, L. Y., Pinto, L., Tan, L., Ott, L., Lee, L., Tomizuka, M., Du, M., Ahn, M., Zhang, M., Ding, M., Srirama, M. K., Sharma, M., Kim, M. J., Kanazawa, N., Hansen, N., Heess, N. M. O., Joshi, N. J., Suenderhauf, N., Palo, N. D., Shafiullah, N. M. M., Mees, O., Kroemer, O., Sanketi, P. R., Wohlhart, P., Xu, P., Sermanet, P., Sundaresan, P., Vuong, Q. H., Rafailov, R., Tian, R., Doshi, R., Mendonca, R., Shah, R., Hoque, R., Julian, R. C., Bustamante, S., Kirmani, S., Levine, S., Moore, S., Bahl, S., Dass, S., Song, S., Xu, S., Haldar, S., Adebola, S. O., Guist, S., Nasiriany, S., Schaal, S., Welker, S., Tian, S., Dasari, S., Belkhale, S., Osa, T., Harada, T., Matsushima, T., Xiao, T., Yu, T., Ding, T., Davchev, T., Zhao, T., Armstrong, T., Darrell, T., Jain, V., Vanhoucke, V., Zhan, W., Zhou, W., Burgard, W., Chen, X., Wang, X., Zhu, X., Li, X., Lu, Y., Chebotar, Y., Zhou, Y., Zhu, Y., Xu, Y., Wang, Y., Bisk, Y., Cho, Y., Lee, Y., Cui, Y., hua Wu, Y., Tang, Y., Zhu, Y., Li, Y., Iwasawa, Y., Matsuo, Y., Xu, Z., and Cui, Z. J. Open x-embodiment: Robotic learning datasets and rt-x models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Parisi, S., Rajeswaran, A., Purushwalkam, S., and Gupta, A. K. The unsurprising effectiveness of pre-trained vision models for control. In *International Conference on Machine Learning (ICML)*, 2022.

Park, S., Ghosh, D., Eysenbach, B., and Levine, S. Hiql: Offline goal-conditioned rl with latent states as actions. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Park, S., Rybkin, O., and Levine, S. Metra: Scalable unsupervised rl with metric-aware abstraction. In *International Conference on Learning Representations (ICLR)*, 2024.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Pathak, D., Gandhi, D., and Gupta, A. K. Self-supervised exploration via disagreement. In *International Conference on Machine Learning (ICML)*, 2019.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *ArXiv*, abs/1910.00177, 2019.

Pertsch, K., Lee, Y., and Lim, J. J. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

Pitis, S., Chan, H., Jamali, K., and Ba, J. An inductive bias for distances: Neural nets that respect the triangle inequality. In *International Conference on Learning Representations (ICLR)*, 2020.

Rajeswar, S., Mazzaglia, P., Verbelen, T., Pich'e, A., Dhoedt, B., Courville, A. C., and Lacoste, A. Mastering the unsupervised reinforcement learning benchmark from pixels. In *International Conference on Machine Learning (ICML)*, 2023.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *Transactions on Machine Learning Research (TMLR)*, 2022.

Seo, Y., Lee, K., James, S., and Abbeel, P. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning (ICML)*, 2022.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., and Levine, S. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Shah, R. and Kumar, V. Rrl: Resnet as representation for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

Sutton, R. S. and Barto, A. G. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16: 285–286, 2005.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., and Riedmiller, M. A. Deepmind control suite. *ArXiv*, abs/1801.00690, 2018.

Touati, A. and Ollivier, Y. Learning one representation to optimize all rewards. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Touati, A., Rapin, J., and Ollivier, Y. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations (ICLR)*, 2023.

Wang, T., Torralba, A., Isola, P., and Zhang, A. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023.

Wu, P., Majumdar, A., Stone, K., Lin, Y., Mordatch, I., Abbeel, P., and Rajeswaran, A. Masked trajectory models for prediction, representation, and control. In *International Conference on Machine Learning (ICML)*, 2023.

Wu, Y., Tucker, G., and Nachum, O. The laplacian in rl: Learning representations with efficient approximations. In *International Conference on Learning Representations (ICLR)*, 2019.

Xiao, T., Radosavovic, I., Darrell, T., and Malik, J. Masked visual pre-training for motor control. *ArXiv*, abs/2203.06173, 2022.

Yang, R., Lin, Y., 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 (ICML)*, 2023.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning (ICML)*, 2021.

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*, abs/2201.13425, 2022.

Zheng, Z., Veeriah, V., Vuorio, R., Lewis, R. L., and Singh, S. Learning state representations from random deep action-conditional predictions. In *Neural Information Processing Systems (NeurIPS)*, 2021.

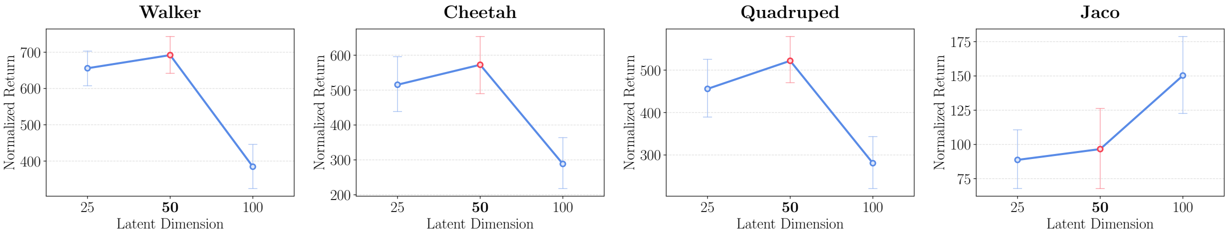


Figure 8. **Ablation study of latent dimensions for zero-shot RL.** In the ExORL benchmark, $D = 50$ generally leads to the best performance across the environments. The results are aggregated over 4 tasks, 4 datasets, and 4 seeds (*i.e.*, 64 values in total).

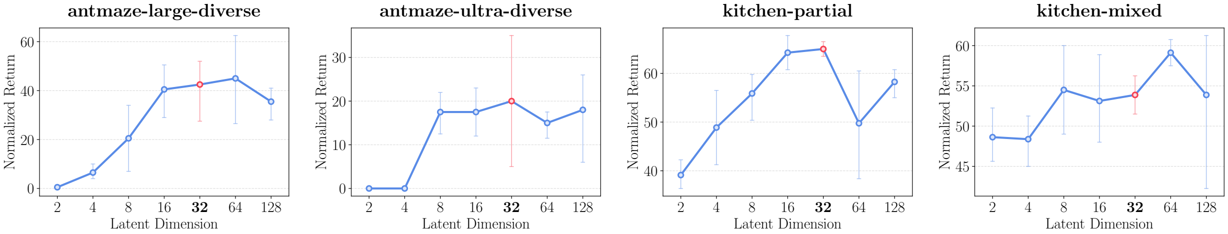


Figure 9. **Ablation study of latent dimensions for offline goal-conditioned RL (4 seeds).** In D4RL tasks, $D = 32$ generally leads to the best performance across the environments.

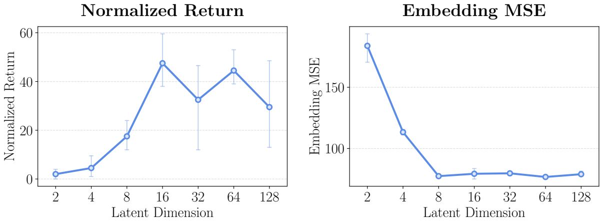
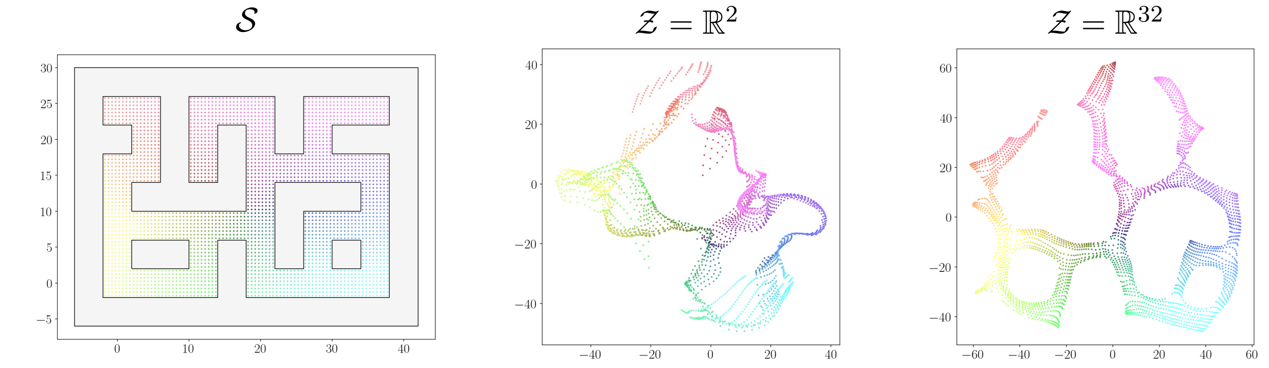


Figure 10. **Embedding error analysis (4 seeds).** We show the relationship between performance and Hilbert embedding errors on the antmaze-large-diverse task. In general, lower embedding errors lead to better goal-reaching performance.

A. Limitations

As mentioned in Section 4.1, one limitation of our Hilbert representation objective is that a symmetric Hilbert space might not be expressive enough to capture arbitrary MDPs. Although we empirically demonstrate that HILPs exhibit strong performances in various complex, long-horizon simulated robotic environments, they might struggle in highly asymmetric or disconnected MDPs (*e.g.*, environments in which gravity plays a significant role), where there might not exist a reasonable approximate isometry to a Hilbert space. We believe this limitation may be resolved by combining the notion of an inner product with a universal quasimetric embedding (Pitis et al., 2020; Wang et al., 2023), which we leave for future work. Another limitation is that our value-based representation learning objective (Equation (6)) might be optimistically biased in stochastic or partially observable environments (Park et al., 2023). We believe combining our method with history-conditioned policies or recurrent world models may be one possible solution to deal with such MDPs. We also note that our experiments assume the state space and environment dynamics to be the same at evaluation time. We leave applying HILPs to multi-environment or transfer learning settings for future work. Finally, we use Euclidean spaces as Hilbert spaces for our experiments in this work. We believe applying HILPs to more general Hilbert spaces, such as the L^2 function space or reproducing kernel Hilbert spaces, may significantly enhance the expressivity of Hilbert representations.



(a) AntMaze states (b) 2-D embedding $\phi(s)$ (c) 32-D embedding $\phi(s)$ (t-SNE)

Figure 11. **Visualization of Hilbert representations.** We visualize Hilbert representations learned on the antmaze-large-diverse dataset. Since Hilbert representations are learned to capture the temporal structure of the MDP, they focus on the global layout of the maze even when we use a two-dimensional latent space ($D = 2$), and accurately capture the maze layout with a 32-dimensional latent space ($D = 32$).

B. Additional Results

Ablation study. Figures 8 and 9 show how the dimension D of the latent space $\mathcal{Z} = \mathbb{R}^D$ affects the performances of zero-shot RL and offline goal-conditioned RL (without planning), where we use $D = 50$ for zero-shot RL and $D = 32$ for goal-conditioned RL in our main experiments. The results suggest that a latent dimension between 25 and 64 generally leads to the best performance in both cases.

Embedding error analysis. To understand the relationship between Hilbert embedding errors and goal-reaching performance, we compare the mean squared errors (MSEs) of Hilbert representations (*i.e.*, $\mathbb{E}[(d^*(s,g) - \|\phi(s) - \phi(g)\|)^2]$) and the final performances with different embedding dimensions on the antmaze-large-diverse task. To approximate the ground-truth temporal distance $d^*(s,g)$ in practice, we employ a monolithic goal-conditioned value function $V(s,g) \approx -d^*(s,g)$ trained with a separate goal-conditioned IQL objective. We use the same IQL expectile of 0.9 for both value functions in this analysis. Figure 10 shows the results, suggesting that low embedding errors generally lead to better goal-reaching performances, as predicted by Theorem 5.1.

Visualization of Hilbert representations. We train Hilbert representations with two different latent dimensions ($D \in \{2, 32\}$) on the antmaze-large-diverse dataset, and visualize the learned latent spaces in Figure 11. We use a t-distributed stochastic neighbor embedding (t-SNE) to visualize 32-dimensional latent states. Since Hilbert representations are learned to preserve the temporal structure of the underlying environment, they focus on the global layout of the maze even when we use a very low-dimensional latent space ($D = 2$, Figure 11b), and accurately capture the layout with $D = 32$ (Figure 11c).

Full results on the ExORL benchmark. Tables 3 and 4 present the full results (*unnormalized* returns) on the state- and pixel-based ExORL benchmarks. Figures 12 to 17 show plots with three different aggregation criteria (per-environment, per-dataset, and per-task) on the state- and pixel-based ExORL benchmarks.

Table 3. **Full results on the state-based ExORL benchmark (4 seeds).** The table shows the *unnormalized* return averaged over four seeds in each setting.

Dataset	Environment	Task	GC-TD3	CL	Rand	IDM	AE	Lap	FDM	FB	**HILP (ours)**	**HILP-G (ours)**
APS	Walker	Flip	406 ±153	137 ±66	92 ±41	354 ±181	289 ±37	390 ±128	426 ±68	334 ±178	573 ±37	-
		Run	274 ±59	75 ±33	79 ±18	357 ±78	140 ±18	217 ±72	248 ±23	388 ±27	348 ±14	-
		Stand	853 ±78	438 ±64	372 ±76	846 ±113	694 ±54	637 ±150	865 ±77	824 ±54	883 ±42	-
		Walk	627 ±201	185 ±112	221 ±68	596 ±70	323 ±131	495 ±84	634 ±91	842 ±105	862 ±31	-
	Cheetah	Run	133 ±95	8 ±8	165 ±82	430 ±24	119 ±52	263 ±28	116 ±116	250 ±135	373 ±72	-
		Run Backward	156 ±137	38 ±24	192 ±31	466 ±30	335 ±30	240 ±18	360 ±17	251 ±39	316 ±21	-
		Walk	695 ±334	32 ±45	633 ±295	988 ±1	414 ±161	964 ±26	396 ±287	683 ±267	939 ±55	-
		Walk Backward	930 ±33	197 ±121	905 ±62	986 ±1	973 ±14	984 ±1	982 ±2	980 ±3	985 ±2	-
	Quadruped	Jump	732 ±62	78 ±67	780 ±14	144 ±97	740 ±58	696 ±68	707 ±30	757 ±52	623 ±149	-
		Run	420 ±37	77 ±90	486 ±3	87 ±36	481 ±11	483 ±12	481 ±4	474 ±33	411 ±62	-
		Stand	938 ±32	131 ±128	965 ±4	177 ±106	944 ±19	914 ±65	961 ±20	949 ±30	797 ±117	-
		Walk	486 ±53	91 ±106	513 ±51	99 ±40	600 ±131	550 ±53	578 ±145	584 ±12	605 ±75	-
	Jaco	Reach Bottom Left	89 ±53	1 ±1	5 ±6	37 ±20	1 ±0	16 ±6	12 ±18	14 ±12	88 ±41	78 ±11
		Reach Bottom Right	121 ±80	0 ±0	5 ±9	31 ±14	6 ±5	10 ±4	28 ±20	24 ±7	48 ±24	84 ±14
		Reach Top Left	71 ±48	1 ±1	3 ±3	20 ±16	5 ±6	51 ±40	21 ±16	23 ±17	49 ±18	75 ±3
		Reach Top Right	71 ±74	1 ±1	7 ±7	24 ±21	12 ±20	26 ±15	34 ±40	17 ±15	51 ±32	80 ±13
APT	Walker	Flip	413 ±44	162 ±61	99 ±14	480 ±48	486 ±53	575 ±37	519 ±80	526 ±89	714 ±89	-
		Run	187 ±19	92 ±27	88 ±4	328 ±26	284 ±22	301 ±30	338 ±69	386 ±13	440 ±39	-
		Stand	757 ±277	531 ±121	593 ±139	862 ±29	866 ±85	791 ±104	873 ±44	884 ±10	877 ±68	-
		Walk	673 ±256	138 ±52	232 ±156	640 ±160	775 ±91	685 ±94	719 ±145	891 ±41	843 ±57	-
	Cheetah	Run	101 ±126	68 ±19	51 ±17	398 ±107	79 ±26	172 ±60	194 ±75	141 ±91	269 ±69	-
		Run Backward	38 ±19	26 ±10	30 ±7	331 ±86	146 ±41	157 ±81	188 ±105	49 ±9	157 ±98	-
		Walk	475 ±350	325 ±76	218 ±42	875 ±131	326 ±139	703 ±235	684 ±215	439 ±307	808 ±142	-
		Walk Backward	265 ±66	128 ±49	149 ±50	946 ±59	455 ±114	642 ±238	646 ±199	208 ±43	779 ±229	-
	Quadruped	Jump	608 ±101	281 ±166	686 ±49	167 ±67	593 ±56	743 ±61	722 ±40	738 ±45	686 ±9	-
		Run	389 ±70	196 ±130	458 ±22	120 ±37	415 ±41	452 ±21	454 ±37	424 ±73	461 ±16	-
		Stand	918 ±77	379 ±246	914 ±44	155 ±48	835 ±82	879 ±75	919 ±34	891 ±50	930 ±30	-
		Walk	441 ±43	191 ±104	465 ±18	94 ±23	417 ±42	433 ±75	513 ±64	427 ±35	468 ±22	-
	Jaco	Reach Bottom Left	1 ±1	0 ±0	2 ±2	34 ±12	5 ±9	3 ±4	75 ±30	7 ±8	12 ±17	30 ±16
		Reach Bottom Right	0 ±1	0 ±0	1 ±1	47 ±14	3 ±3	9 ±13	56 ±22	8 ±3	29 ±44	37 ±14
		Reach Top Left	0 ±0	0 ±0	9 ±15	19 ±5	7 ±13	1 ±1	13 ±12	28 ±17	5 ±4	30 ±14
		Reach Top Right	0 ±0	0 ±0	1 ±1	40 ±10	3 ±6	2 ±3	12 ±4	13 ±10	21 ±24	34 ±11
Proto	Walker	Flip	494 ±153	216 ±61	159 ±41	432 ±62	553 ±105	531 ±104	433 ±64	560 ±94	675 ±62	-
		Run	324 ±67	141 ±26	68 ±36	251 ±42	312 ±47	352 ±17	296 ±43	425 ±49	402 ±28	-
		Stand	796 ±179	629 ±108	407 ±110	903 ±73	916 ±26	897 ±65	900 ±42	844 ±103	930 ±27	-
		Walk	866 ±33	383 ±57	190 ±214	588 ±195	851 ±37	864 ±50	873 ±63	905 ±29	905 ±32	-
	Cheetah	Run	135 ±89	4 ±3	123 ±92	370 ±59	160 ±121	282 ±37	363 ±24	223 ±42	227 ±27	-
		Run Backward	173 ±34	5 ±3	177 ±33	383 ±23	254 ±38	218 ±10	309 ±28	151 ±38	234 ±13	-
		Walk	923 ±81	23 ±18	556 ±362	939 ±63	563 ±363	982 ±9	944 ±60	949 ±40	973 ±12	-
		Walk Backward	560 ±368	28 ±17	820 ±136	987 ±1	916 ±106	981 ±6	984 ±1	737 ±160	985 ±2	-
	Quadruped	Jump	298 ±60	27 ±20	176 ±59	69 ±39	289 ±35	214 ±27	287 ±61	231 ±75	282 ±105	-
		Run	176 ±103	17 ±15	127 ±29	51 ±24	187 ±27	181 ±52	243 ±37	126 ±26	227 ±42	-
		Stand	436 ±45	35 ±28	307 ±35	117 ±60	403 ±48	287 ±52	453 ±63	262 ±16	425 ±156	-
		Walk	237 ±39	16 ±15	120 ±38	58 ±11	229 ±25	175 ±60	238 ±56	224 ±123	136 ±50	-
	Jaco	Reach Bottom Left	10 ±12	1 ±2	1 ±1	53 ±19	11 ±8	3 ±4	60 ±28	15 ±24	3 ±4	60 ±14
		Reach Bottom Right	1 ±1	0 ±0	2 ±3	57 ±28	29 ±35	2 ±2	80 ±31	16 ±31	7 ±12	61 ±4
		Reach Top Left	5 ±3	2 ±2	4 ±6	37 ±11	9 ±8	1 ±0	17 ±17	11 ±5	13 ±12	60 ±5
		Reach Top Right	17 ±13	1 ±1	0 ±0	34 ±16	22 ±21	2 ±3	32 ±27	39 ±40	12 ±9	61 ±4
RND	Walker	Flip	298 ±110	115 ±16	252 ±43	453 ±21	499 ±33	563 ±40	416 ±20	548 ±94	563 ±136	-
		Run	166 ±27	59 ±19	75 ±47	205 ±69	259 ±18	311 ±29	295 ±70	409 ±15	401 ±30	-
		Stand	812 ±133	309 ±100	397 ±50	778 ±52	806 ±44	844 ±35	821 ±84	866 ±120	800 ±61	-
		Walk	703 ±275	99 ±35	149 ±145	401 ±160	676 ±135	811 ±77	476 ±259	811 ±52	855 ±34	-
	Cheetah	Run	65 ±66	49 ±34	49 ±24	139 ±67	116 ±12	124 ±32	107 ±26	183 ±83	262 ±53	-
		Run Backward	50 ±35	20 ±6	28 ±10	298 ±27	143 ±86	113 ±37	177 ±62	153 ±41	187 ±55	-
		Walk	281 ±116	210 ±123	231 ±121	607 ±181	500 ±114	571 ±111	494 ±122	636 ±291	823 ±141	-
		Walk Backward	146 ±51	130 ±62	151 ±66	956 ±16	528 ±296	536 ±92	652 ±274	677 ±85	843 ±184	-
	Quadruped	Jump	639 ±106	402 ±363	737 ±48	174 ±48	698 ±94	508 ±182	758 ±98	642 ±36	556 ±101	-
		Run	435 ±38	289 ±229	458 ±27	103 ±38	446 ±57	346 ±81	491 ±7	436 ±26	393 ±42	-
		Stand	910 ±44	555 ±477	934 ±26	240 ±68	831 ±113	681 ±191	971 ±11	797 ±72	810 ±97	-
		Walk	470 ±21	303 ±221	541 ±98	92 ±41	492 ±198	441 ±88	601 ±82	642 ±202	542 ±32	-
	Jaco	Reach Bottom Left	1 ±1	0 ±0	1 ±1	60 ±18	1 ±2	18 ±25	53 ±20	18 ±11	19 ±20	46 ±14
		Reach Bottom Right	1 ±1	0 ±0	2 ±2	45 ±24	1 ±1	7 ±9	38 ±14	29 ±12	18 ±17	55 ±23
		Reach Top Left	0 ±0	0 ±0	0 ±0	34 ±4	6 ±8	5 ±8	36 ±19	45 ±16	8 ±10	47 ±15
		Reach Top Right	0 ±0	1 ±2	1 ±0	22 ±6	6 ±3	8 ±10	44 ±24	22 ±7	5 ±4	38 ±18

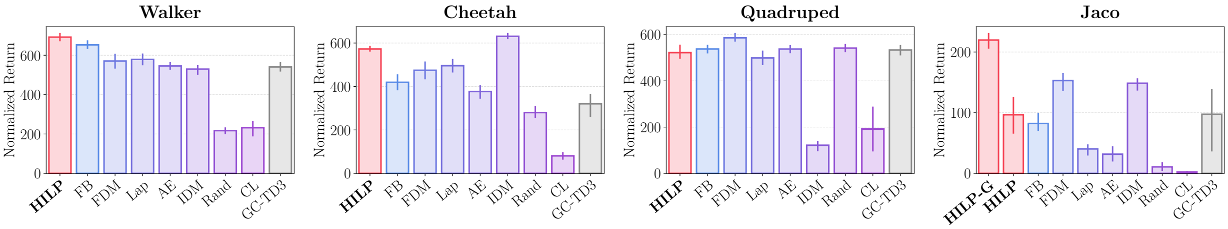


Figure 12. **Per-environment performances on the state-based ExORL benchmark.** The results are aggregated over 4 tasks, 4 datasets, and 4 seeds (*i.e.*, 64 values in total).

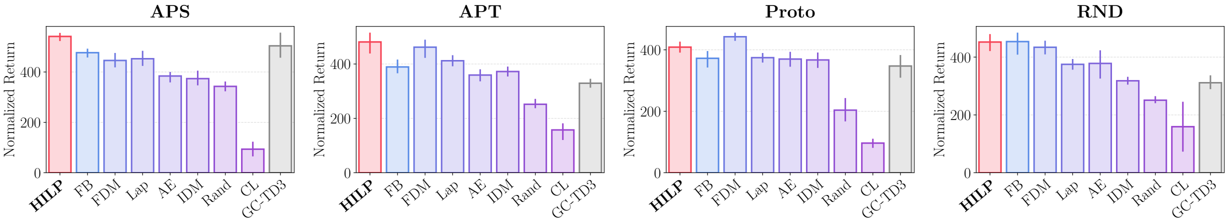


Figure 13. **Per-dataset performances on the state-based ExORL benchmark.** The results are aggregated over 4 environments, 4 tasks, and 4 seeds (*i.e.*, 64 values in total).

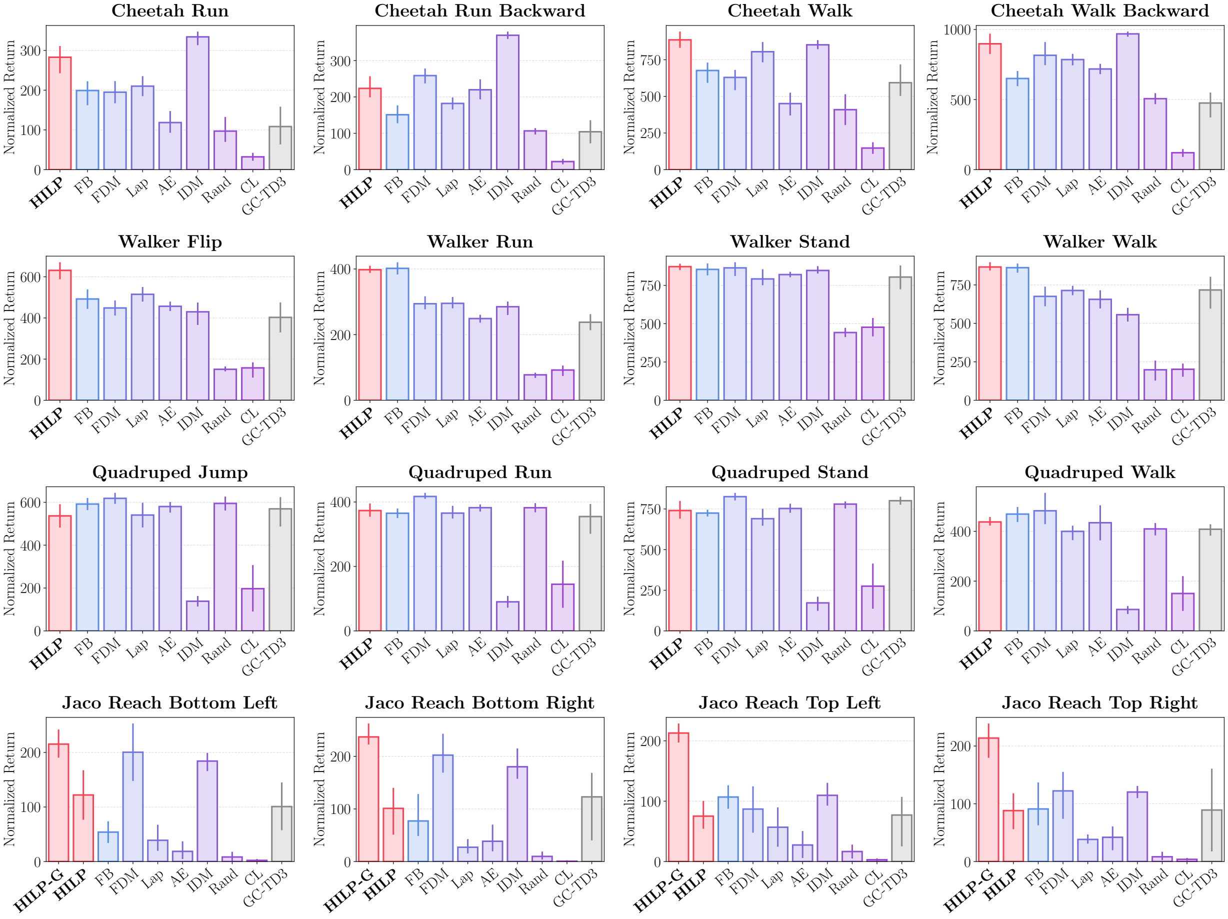


Figure 14. **Per-task performances on the state-based ExORL benchmark.** The results are aggregated over 4 datasets and 4 seeds (*i.e.*, 16 values in total).

Table 4. **Full results on the pixel-based ExORL benchmark (4 seeds).** The table shows the *unnormalized* return averaged over four seeds in each setting.

Dataset	Environment	Task	FDM	FB	**HILP (ours)**
APS	Walker	Flip	158 ±25	66 ±34	127 ±23
		Run	126 ±26	53 ±18	97 ±2
		Stand	608 ±90	340 ±122	520 ±30
		Walk	317 ±156	231 ±109	372 ±68
	Cheetah	Run	189 ±41	21 ±34	118 ±113
		Run Backward	59 ±73	24 ±28	248 ±46
		Walk	613 ±77	52 ±91	273 ±383
		Walk Backward	371 ±307	89 ±101	967 ±8
	Quadruped	Jump	224 ±23	291 ±34	301 ±31
		Run	172 ±27	231 ±23	204 ±39
		Stand	343 ±16	444 ±53	397 ±47
		Walk	176 ±25	230 ±22	195 ±16
	Jaco	Reach Bottom Left	12 ±2	52 ±45	63 ±27
		Reach Bottom Right	29 ±18	53 ±18	68 ±11
		Reach Top Left	21 ±8	31 ±21	62 ±35
		Reach Top Right	36 ±13	53 ±30	62 ±46
APT	Walker	Flip	353 ±37	58 ±28	280 ±26
		Run	239 ±34	47 ±17	160 ±25
		Stand	768 ±110	257 ±82	486 ±16
		Walk	504 ±79	66 ±30	514 ±71
	Cheetah	Run	251 ±18	28 ±13	326 ±67
		Run Backward	169 ±109	24 ±15	249 ±83
		Walk	737 ±54	107 ±49	880 ±60
		Walk Backward	578 ±137	109 ±67	839 ±99
	Quadruped	Jump	270 ±40	187 ±44	207 ±62
		Run	188 ±38	121 ±28	125 ±34
		Stand	375 ±112	242 ±49	267 ±60
		Walk	173 ±36	127 ±41	124 ±32
	Jaco	Reach Bottom Left	17 ±9	24 ±24	18 ±6
		Reach Bottom Right	31 ±32	22 ±20	23 ±4
		Reach Top Left	42 ±8	97 ±66	27 ±4
		Reach Top Right	64 ±27	37 ±32	39 ±24
Proto	Walker	Flip	267 ±65	85 ±56	140 ±65
		Run	212 ±44	48 ±19	108 ±35
		Stand	854 ±46	282 ±164	533 ±132
		Walk	563 ±268	88 ±61	347 ±71
	Cheetah	Run	87 ±67	11 ±12	116 ±31
		Run Backward	41 ±22	5 ±5	170 ±78
		Walk	150 ±126	32 ±51	410 ±303
		Walk Backward	384 ±232	26 ±31	743 ±267
	Quadruped	Jump	182 ±22	150 ±41	210 ±62
		Run	120 ±21	98 ±17	158 ±71
		Stand	226 ±65	181 ±29	293 ±134
		Walk	108 ±39	87 ±21	157 ±50
	Jaco	Reach Bottom Left	24 ±22	75 ±29	32 ±19
		Reach Bottom Right	26 ±21	31 ±31	24 ±18
		Reach Top Left	43 ±25	47 ±16	38 ±10
		Reach Top Right	48 ±21	60 ±37	66 ±19
RND	Walker	Flip	282 ±52	62 ±57	232 ±41
		Run	146 ±60	42 ±25	126 ±8
		Stand	557 ±99	172 ±111	496 ±73
		Walk	452 ±52	104 ±82	376 ±52
	Cheetah	Run	178 ±41	221 ±15	276 ±46
		Run Backward	126 ±9	171 ±123	297 ±46
		Walk	470 ±182	535 ±251	895 ±33
		Walk Backward	441 ±107	535 ±440	927 ±35
	Quadruped	Jump	273 ±66	224 ±149	244 ±122
		Run	192 ±29	158 ±82	148 ±19
		Stand	374 ±85	347 ±191	327 ±126
		Walk	199 ±63	162 ±92	163 ±45
	Jaco	Reach Bottom Left	20 ±14	62 ±16	25 ±3
		Reach Bottom Right	16 ±8	49 ±29	31 ±11
		Reach Top Left	40 ±16	53 ±7	29 ±4
		Reach Top Right	38 ±21	70 ±17	35 ±13

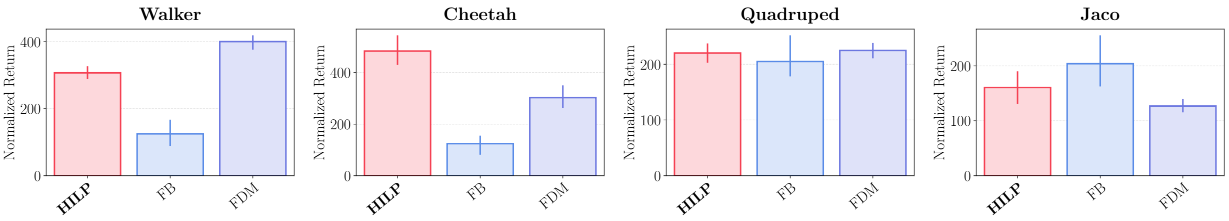


Figure 15. **Per-environment performances on the pixel-based ExORL benchmark.** The results are aggregated over 4 tasks, 4 datasets, and 4 seeds (*i.e.*, 64 values in total).

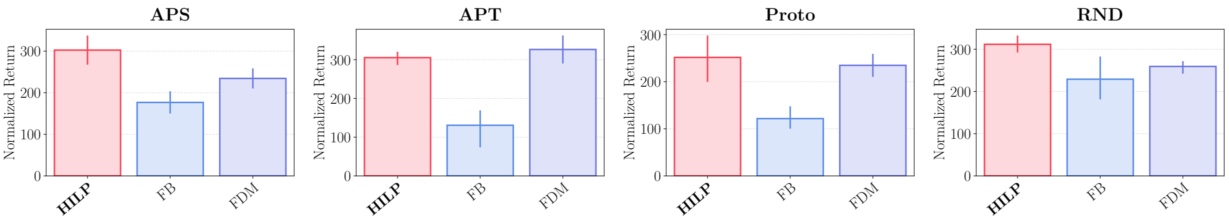


Figure 16. **Per-dataset performances on the pixel-based ExORL benchmark.** The results are aggregated over 4 environments, 4 tasks, and 4 seeds (*i.e.*, 64 values in total).

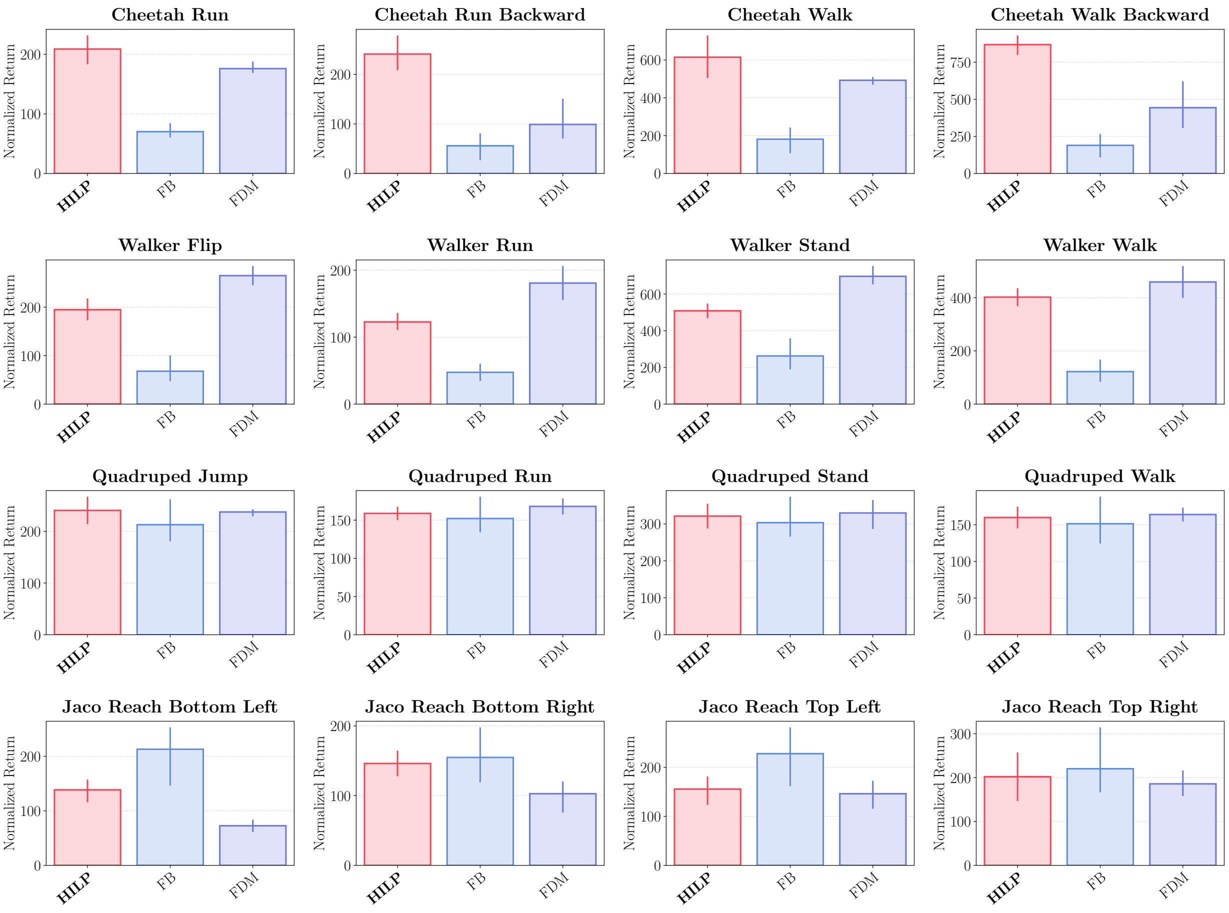


Figure 17. **Per-task performances on the pixel-based ExORL benchmark.** The results are aggregated over 4 datasets and 4 seeds (*i.e.*, 16 values in total).

C. Theoretical Results

Let $d^* : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ be the optimal temporal distance function. Let $\phi : \mathcal{S} \to \mathcal{Z}$ be a representation function that maps states into a real Hilbert space with an inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\|\cdot\|$. The role of ϕ is to embed temporal distances into the latent Hilbert space such that $d^*(s, g) \approx \|\phi(s) - \phi(g)\|$. Since we assume a deterministic MDP, we denote the transition dynamics function and policies as deterministic functions: $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $\pi : \mathcal{S} \to \mathcal{A}$. In this section, we will show that moving in the direction of $\phi(g) - \phi(s)$ is optimal to reach the goal g from the state s if embedding errors are sufficiently small.

For a state s and a goal g, we define the following functions:

$$z'^*(s, g) := \phi(s) + \frac{\phi(g) - \phi(s)}{\|\phi(g) - \phi(s)\|}, \tag{12}$$

$$\hat{\pi}(s, g) := \operatorname*{arg\,max}_{a \in \mathcal{A}} \left\langle \phi(s') - \phi(s), \frac{\phi(g) - \phi(s)}{\|\phi(g) - \phi(s)\|} \right\rangle \quad \text{s.t.} \quad s' = p(s, a),\ \|\phi(s) - \phi(s')\| \le 1, \tag{13}$$

$$\hat{z}'(s, g) := \phi(p(s, \hat{\pi}(s, g))). \tag{14}$$

We denote the neighborhood states of s as $N(s) := \{p(s, a) : a \in \mathcal{A}\}$. Intuitively, $z'^*(s, g)$ is the optimal latent point that is a unit distance away in the goal direction from the current latent state, and $\hat{z}'(s, g)$ is the optimal next latent state that maximizes the directional reward $\langle \phi(s') - \phi(s), (\phi(g) - \phi(s))/\|\phi(g) - \phi(s)\| \rangle$,

The following theorem states a condition for the policy $\hat{\pi}(s, g)$ to be optimal at (s, g).

Theorem C.1. *For a state-goal pair $(s, g) \in \mathcal{S} \times \mathcal{S}$, $s \neq g$, assume that the local embedding error is bounded as $\sup_{s' \in N(s) \cup \{s\}} |d^*(s', g) - \|\phi(s') - \phi(g)\|| \le \varepsilon_e$ and the directional movement error is bounded as $\|z'^*(s, g) - \hat{z}'(s, g)\| \le \varepsilon_d$. If $4\varepsilon_e + \varepsilon_d < 1$, $\hat{\pi}(s, g)$ is guaranteed to be an optimal action at (s, g).*

Proof. Define $\hat{s}' := p(s, \hat{\pi}(s, g))$. Since $s \neq g$, we know $d^*(s, g) \ge 1$. To show that $\hat{\pi}(s, g)$ is an optimal action, it suffices to show that the temporal distance toward the goal is reduced by 1 when the agent moves from s to $\hat{s}'$. We bound the difference between $d^*(\hat{s}', g)$ and $d^*(s, g) - 1$ as follows:

$$|d^*(\hat{s}', g) - (d^*(s, g) - 1)| \tag{15}$$

$$\le |d^*(\hat{s}', g) - \|\phi(\hat{s}') - \phi(g)\|| + |\|\phi(\hat{s}') - \phi(g)\| - (\|\phi(s) - \phi(g)\| - 1)| + |(\|\phi(s) - \phi(g)\| - 1) - (d^*(s, g) - 1)| \tag{16}$$

$$\le \varepsilon_e + |\|\phi(\hat{s}') - \phi(g)\| - (\|\phi(s) - \phi(g)\| - 1)| + \varepsilon_e \tag{17}$$

$$= 2\varepsilon_e + \|\phi(\hat{s}') - \phi(g)\| - (\|\phi(s) - \phi(g)\| - 1) \tag{18}$$

$$\le 2\varepsilon_e + \|\phi(\hat{s}') - z'^*(s, g)\| + \|z'^*(s, g) - \phi(g)\| - (\|\phi(s) - \phi(g)\| - 1), \tag{19}$$

where we use

$$\|\phi(\hat{s}') - \phi(g)\| + 1 \ge \|\phi(\hat{s}') - \phi(g)\| + \|\phi(s) - \phi(\hat{s}')\| \tag{20}$$

$$\ge \|\phi(s) - \phi(g)\| \tag{21}$$

for Equation (18). To bound Equation (19), we consider the following two cases.

Case #1: $\|\phi(g) - \phi(s)\| \ge 1$. In this case, we have

$$\|z'^*(s, g) - \phi(g)\| = \left\| \phi(s) + \frac{\phi(g) - \phi(s)}{\|\phi(g) - \phi(s)\|} - \phi(g) \right\| \tag{22}$$

$$= |\|\phi(g) - \phi(s)\| - 1| \tag{23}$$

$$= \|\phi(g) - \phi(s)\| - 1. \tag{24}$$

Case #2: $\|\phi(g) - \phi(s)\| < 1$. Similarly, we have

$$\|z'^*(s, g) - \phi(g)\| = 1 - \|\phi(g) - \phi(s)\|, \tag{25}$$

and thus

$$
\begin{aligned}
&\|z'^*(s,g)-\phi(g)\|-(\|\phi(s)-\phi(g)\|-1) \tag{26}\\
=&2(1-\|\phi(s)-\phi(g)\|) \tag{27}\\
\le&2(d^*(s,g)-\|\phi(s)-\phi(g)\|) \tag{28}\\
\le&2\varepsilon_e. \tag{29}
\end{aligned}
$$

As $\|z'^*(s,g)-\phi(g)\|-(\|\phi(s)-\phi(g)\|-1)$ is bounded by $2\varepsilon_e$ in both cases, we have

$$
\begin{aligned}
|d^*(\hat{s}',g)-(d^*(s,g)-1)| &\le 2\varepsilon_e+\|\phi(\hat{s}')-z'^*(s,g)\|+\|z'^*(s,g)-\phi(g)\|-(\|\phi(s)-\phi(g)\|-1) \tag{30}\\
&\le 4\varepsilon_e+\|\phi(\hat{s}')-z'^*(s,g)\| \tag{31}\\
&\le 4\varepsilon_e+\varepsilon_d. \tag{32}
\end{aligned}
$$

Since we have $4\varepsilon_e+\varepsilon_d<1$ and $|d^*(\hat{s}',g)-(d^*(s,g)-1)|$ is an integer, $|d^*(\hat{s}',g)-(d^*(s,g)-1)|$ must be zero and thus $\hat{\pi}(s,g)$ is an optimal action. □

A keen reader may notice that Theorem C.1 only depends on $z'^*(s,g)$ and $\hat{z}'(s,g)$, and is agnostic to the *objective* of the policy (Equation (13)), $\langle\phi(s')-\phi(s),(\phi(g)-\phi(s))/\|\phi(g)-\phi(s)\|\rangle$. The following theorem justifies this directional objective: namely, the policy objective in Equation (13) finds the optimal next latent state $z'^*(s,g)$ if it is a feasible point.

Theorem C.2. *If $z'^*(s,g)\in\{\phi(s'):s'\in N(s),\|\phi(s)-\phi(s')\|\le 1\}$, then $\hat{z}'(s,g)=z'^*(s,g)$.*

Proof. Recall that $\hat{\pi}(s,g)$ is defined as

$$
\hat{\pi}(s,g)=\operatorname*{arg\,max}_{a\in\mathcal{A}}\left\langle\phi(s')-\phi(s),\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\rangle\quad\text{s.t.}\quad s'=p(s,a),\ \|\phi(s)-\phi(s')\|\le 1. \tag{33}
$$

$$\tag{34}$$

We have

$$
\begin{aligned}
&\max_{s'\in N(s)}\left\langle\phi(s')-\phi(s),\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\rangle\quad\text{s.t.}\quad\|\phi(s)-\phi(s')\|\le 1 \tag{35}\\
\le&\max_{s'\in\mathcal{S}}\left\langle\phi(s')-\phi(s),\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\rangle\quad\text{s.t.}\quad\|\phi(s)-\phi(s')\|\le 1 \tag{36}\\
\le&\max_{s'\in\mathcal{S}}\|\phi(s')-\phi(s)\|\left\|\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\|\quad\text{s.t.}\quad\|\phi(s)-\phi(s')\|\le 1 \tag{37}\\
\le&1, \tag{38}
\end{aligned}
$$

by the Cauchy-Schwarz inequality. By the assumption, $z'^*(s,g)$ is a feasible point and

$$
\begin{aligned}
&\left\langle z'^*(s,g)-\phi(s),\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\rangle \tag{39}\\
=&\left\langle\phi(s)+\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}-\phi(s),\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\rangle \tag{40}\\
=&\left\langle\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|},\frac{\phi(g)-\phi(s)}{\|\phi(g)-\phi(s)\|}\right\rangle \tag{41}\\
=&1 \tag{42}
\end{aligned}
$$

holds. Hence, the maximum in Equation (35) is attainable by $z'^*(s,g)$, and thus $\hat{z}'(s,g)=z'^*(s,g)$ holds. □

Finally, as a corollary to Theorem C.1, we have the following:

Corollary C.3. *If embedding errors are bounded as $\sup_{s,g\in\mathcal{S}}|d^*(s,g)-\|\phi(s)-\phi(g)\||\le\varepsilon_e$, directional movement errors are bounded as $\sup_{s,g\in\mathcal{S}}\|z'^*(s,g)-\hat{z}'(s,g)\|\le\varepsilon_d$, and $4\varepsilon_e+\varepsilon_d<1$, then $\hat{\pi}(s,g)$ is an optimal goal-reaching policy.*

Intuitively, Corollary C.3 tells us that if the embedding error is small enough, directional movements in the latent space are optimal for solving goal-reaching tasks.

Limitations. One natural question to ask is whether it is always possible to embed any MDP into a Hilbert space up to arbitrary accuracy. Unfortunately, this is not always possible. First, temporal distances are asymmetric but the distance metric of the Hilbert space $\mathcal{Z}$ is symmetric. Second, even when the environment is completely symmetric, there exists a symmetric MDP that is not embeddable into a Hilbert space with an arbitrarily low approximation error (Indyk et al., 2017; Pitis et al., 2020). This is mainly because Hilbert spaces are highly structured, especially compared to metric or quasimetric spaces, which do not require a well-defined inner product. Nonetheless, the inner product structure of the Hilbert space naturally enables useful prompting strategies for zero-shot RL and goal-conditioned RL, and we empirically found that even MDPs that in principle do not have a lossless Hilbert representation can still be solved effectively via our method in our experiments. We believe finding a way to relax the Hilbert condition while having similar prompting and planning strategies is an exciting and important future research direction.

D. Experimental Details

We implement HILPs based on two different codebases: the official implementation of FB representations (Touati et al., 2023) for zero-shot RL experiments and that of HIQL (Park et al., 2023) for offline goal-conditioned RL and hierarchical RL experiments. Our implementations are publicly available at the following repository: `https://github.com/seohongpark/HILP`. We run our experiments on an internal cluster consisting of A5000 GPUs. Each run in this work takes no more than 28 hours.

D.1. Environments and Datasets

ExORL (Yarats et al., 2022). The ExORL benchmark consists of a set of datasets collected by unsupervised RL agents (Laskin et al., 2021) on the DeepMind Control Suite (Tassa et al., 2018). We use four environments (Walker (Figure 4a), Cheetah (Figure 4b), Quadruped (Figure 4c), and Jaco (Figure 4d)) and four datasets collected by APS (Liu & Abbeel, 2021a), APT (Liu & Abbeel, 2021b), Proto (Yarats et al., 2021), and RND (Burda et al., 2019) in each environment. Following Touati et al. (2023), we use the first 5M transitions from each dataset. Each environment has four test-time tasks: Walker has Flip, Run, Stand, and Walk; Cheetah has Run, Run Backward, Walk, and Walk Backward; Quadruped has Jump, Run, Stand, Walk; Jaco has Reach Bottom Left, Reach Bottom Right, Reach Top Left, and Reach Top Right. Among the four environments, Walker, Cheetah, and Quadruped have a maximum return of 1000, and Jaco has a maximum return of 250. As such, we multiply Jaco returns by 4 to normalize them for aggregation. For pixel-based ExORL experiments, we convert each state in the datasets into a $64 \times 64 \times 3$-sized camera image by rendering it.

AntMaze (Fu et al., 2020). The AntMaze datasets from D4RL (Fu et al., 2020) consist of trajectories of a quadrupedal robot navigating through a maze from random locations to other locations. We employ the two most challenging datasets with the largest maze ("antmaze-large-{diverse, play}-v2", Figure 4e) from the original D4RL benchmark, and two even larger settings ("antmaze-ultra-{diverse, play}-v0", Figure 4f) introduced by Jiang et al. (2023), where the "ultra" maze is twice the size of the "large" maze. For goal-conditioned RL experiments in Section 6.2, we use the same goal-conditioned evaluation setting as Park et al. (2023): we specify the test-time goal g by concatenating the x-y coordinates of the original target goal to the proprioceptive state dimensions of the first observation in the dataset. The agent gets a reward of 1 when it reaches the target goal. In Tables 1 and 2, we multiply the returns by 100 to normalize them. For hierarchical RL experiments in Section 6.3, we use the original non-goal-conditioned tasks.

Kitchen (Gupta et al., 2019; Fu et al., 2020). The Kitchen datasets from D4RL (Fu et al., 2020) consist of trajectories of a robotic arm manipulating different kitchen objects in various orders in the Kitchen environment (Gupta et al., 2019) (Figure 4g). We employ two datasets ("kitchen-{partial, mixed}-v0") from the original D4RL benchmark. For goal-conditioned RL experiments in Section 6.2, we use the same goal-conditioned evaluation setting as Park et al. (2023): we specify the test-time goal g by concatenating the proprioceptive state dimensions of the first observation in the dataset to the object states of the target goal given by the environment. The agent gets a reward of 1 whenever it achieves a subtask, where each task consists of a total of four subtasks. In Tables 1 and 2, we multiply the returns by 25 to normalize them. For hierarchical RL experiments in Section 6.3, we use the original non-goal-conditioned tasks. For pixel-based Kitchen experiments, we convert each state in the datasets into a $64 \times 64 \times 3$-sized camera image by rendering it. We use the same camera configuration as Mendonca et al. (2021); Park et al. (2024) (Figure 4g).

D.2. Implementation Details

Hilbert representations. In Equation (6), we use the same goal relabeling strategy as Park et al. (2023) except that we do not set $g = s$, since $V(s, s) = 0$ is always guaranteed in our parameterization. Namely, we sample g either from a geometric distribution over the future states within the same trajectory (with probability 0.625), or uniformly from the dataset (with probability 0.375). We note that the values 0.625 and 0.375 come from the original hyperparameters used by Park et al. (2023) (which use $g = s$ with probability 0.2, future states with probability 0.5, and random states with probability 0.3), where we redistribute the unnecessary probability mass of $g = s$ across the other two bins. To avoid numerical issues with gradient descent, we add a small value ($\varepsilon = 10^{-6}$) when computing $\|\phi(s) - \phi(g)\|$.

Zero-shot RL (Section 6.1). We evaluate HILPs and all baselines on the same codebase built on the official implementation of the work by Touati et al. (2023). For HILP, we use the centered reward function introduced in Section 4.2 and the zero-shot prompting scheme introduced in Section 5.1. For HILP-G in Jaco, we use the reward function in Equation (7) and the goal-conditioned prompting scheme introduced in Section 5.2, where the goal is specified as the state with the highest reward value from the offline dataset. For the FB, SF, and GCRL baselines, we follow the implementations provided by Touati et al. (2023). We use TD3 (Fujimoto et al., 2018) as the base (offline) RL algorithm to train these methods. Following Touati et al. (2023), for the HILP, FB, and SF methods, we either sample a latent vector z uniformly from the prior distribution (with probability 0.5) or set z to the latent vector that corresponds to a goal-reaching task (with probability 0.5). For successor feature losses, we use either the vector loss or the Q loss (Ma et al., 2020), depending on the environment. For hyperparameter tuning, we individually tune HILP, FB, Lap (the best SF method reported in the work by Touati et al. (2023)), and GC-TD3 in each environment with the RND dataset, and apply the found hyperparameters to the other datasets and to the other methods in the same category. We report the full list of the hyperparameters used in our zero-shot RL experiments in Table 5.

Offline goal-conditioned RL (Section 6.2). We implement HILP on top of the official codebase of the work by Park et al. (2023). For HILP, we use the reward function in Equation (7) and the goal-conditioned prompting scheme introduced in Section 5.2. We use IQL (Kostrikov et al., 2022) with AWR (Peng et al., 2019) as an offline algorithm to train policies. For HILP-Plan, at each evaluation epoch, we first randomly sample $N = 50000$ states $w_1, w_2, \ldots, w_N$ from the dataset $\mathcal{D}$, and pre-compute their representations $\phi(w_1), \phi(w_2), \ldots, \phi(w_N)$. Then, at every time step, we find the $\arg\min$ of Equation (10) over the N samples using the pre-computed representations. In practice, we use the average of the 50 $\arg\min$ representations, as we found this to lead to better performance. For GC-IQL and GC-BC, we use the implementations provided by Park et al. (2023). They are implemented on the same codebase as HILP. For GC-CQL, we modify the JaxCQL repository (Geng, 2022) to make it compatible with our goal-conditioned setting. We mostly follow the hyperparameters used by Nakamoto et al. (2023). We use the same goal relabeling strategy as Park et al. (2023) for all three goal-conditioned RL methods. For FB, we use the official implementation provided by Touati et al. (2023), where we additionally implement D4RL environments. For SF methods (FDM, Lap, and Rand), we re-implement IQL versions of them on the same codebase as HILP, as we found these versions to perform better than the original implementations by Touati et al. (2023). Among FB and SF methods, we only re-implement SF methods based on IQL, as FB in its current form is not directly compatible with IQL. We report the full list of the hyperparameters used in our offline goal-conditioned RL experiments in Table 6.

Hierarchical RL (Section 6.3). We implement hierarchical HILP and OPAL on top of the official codebase of the work by Park et al. (2023). To train a high-level policy $\pi^h(z \mid s)$ on top of our latent-conditioned (low-level) policy $\pi(a \mid s, z)$, we first sample (s_t, s_{t+k}) tuples from the dataset, label them with $z = (\phi(s_{t+k}) - \phi(s_t))/\|\phi(s_{t+k}) - \phi(s_t)\|$, and use z as high-level actions. k is a hyperparameter that determines the high-level action length. For OPAL, we use our own implementation on top of the same codebase as HILP, as we were unable to find the official implementation. We sample trajectory chunks $(s_{t:t+k}, a_{t:t+k-1})$ from the dataset and train a trajectory VAE consisting of three components: a trajectory encoder $p(z \mid s_{t:t+k}, a_{t:t+k-1})$ modeled by a bi-directional GRU (Cho et al., 2014), a decoder parameterized as $\pi(a_t \mid s_t, z)$, and a prior $p(z \mid s_t)$. For both HILP and OPAL, to ensure a fair comparison, we use the same offline RL algorithm (IQL) for high-level policy learning. We report the full list of the hyperparameters used in our zero-shot hierarchical RL experiments in Table 7.

[2] Following Touati et al. (2023), for policies $\pi(a \mid s, z)$ and TD3 values $Q(s, a, z)$, we process s (or (s, a)) and (s, z) separately with $(1024, 512)$-sized MLPs, concatenate them together, and then pass another (1024)-sized MLP.

[3] We found that $(256, 256)$-sized policy networks lead to better performance than $(512, 512, 512)$-sized ones for GC-BC and GC-IQL on AntMaze-Large.

[4] We found that OPAL works better with 8-dimensional latent spaces (as in the original work), compared to 32-dimensional ones (as in this work), especially in AntMaze tasks.

Table 5. **Hyperparameters for zero-shot RL.**

Hyperparameter	Value
# gradient steps	10^6 (state-based), 5×10^5 (pixel-based)
Learning rate	0.0005 (ϕ), 0.0001 (others)
Optimizer	Adam (Kingma & Ba, 2015)
Minibatch size	1024 (state-based), 512 (pixel-based)
MLP dimensions	$(512, 512)$ (ϕ), $(1024, 1024, 1024)$ (others)[2]
TD3 target smoothing coefficient	0.01
TD3 discount factor γ	0.98
Latent dimension	50
# state samples for latent vector inference	10000
Successor feature loss	Q loss ({HILP, SF} on {Quadruped, Jaco}), vector loss (others)
Hilbert representation discount factor	0.96 (Walker), 0.98 (others)
Hilbert representation expectile	0.5 (HILP), 0.9 (HILP-G)
Hilbert representation target smoothing coefficient	0.005

Table 6. **Hyperparameters for offline goal-conditioned RL.**

Hyperparameter	Value
# gradient steps	10^6 (AntMaze), 5×10^5 (Kitchen)
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2015)
Minibatch size	1024 (state-based), 256 (pixel-based)
Value MLP dimensions	$(512, 512, 512)$
Policy MLP dimensions	$(256, 256)$ ({GC-BC, GC-IQL} on AntMaze-Large)[3], $(512, 512, 512)$ (others)
Target smoothing coefficient	0.005
Discount factor γ	0.99
Latent dimension	32
Hilbert representation discount factor	0.99
Hilbert representation expectile	0.7 (Visual Kitchen), 0.95 (others)
Hilbert representation target smoothing coefficient	0.005
HILP IQL expectile	0.7 (Visual Kitchen), 0.9 (others)
HILP AWR temperature	10

Table 7. **Hyperparameters for hierarchical RL.**

Hyperparameter	Value
# gradient steps	5×10^5
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2015)
Minibatch size	1024 (HILP), 256 (OPAL, high-level IQL)
Value MLP dimensions	$(512, 512, 512)$
Policy MLP dimensions	$(512, 512, 512)$ (HILP), $(256, 256)$ (high-level IQL)
Target smoothing coefficient	0.005
Discount factor γ	0.99
Latent dimension	32 (HILP), 8 (OPAL)[4]
Hilbert representation discount factor	0.99
Hilbert representation expectile	0.95 (AntMaze), 0.7 (Kitchen)
Hilbert representation target smoothing coefficient	0.005
HILP IQL expectile	0.9
HILP AWR temperature	10
OPAL VAE MLP dimensions	$(256, 256)$
OPAL VAE # GRU layers	2
OPAL VAE KL coefficient	0.1
High-level action length k	10
High-level IQL discount factor	0.99
High-level IQL expectile	0.9 (AntMaze), 0.7 (Kitchen)
High-level AWR temperature	1
High-level value normalization	None (AntMaze), LayerNorm (Ba et al., 2016) (Kitchen)