THE REWARD-AWARE LAPLACIAN REPRESENTATION IN ZERO-SHOT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Representations based on the reward-agnostic successor measure have been shown to be effective in a variety of applications, and, in particular, have led to many advances in zero-shot RL. In this paper, we consider the setting in which, across different tasks, there are shared reward structures that reflect inherent environmental properties, e.g., the negative rewards of wear and tear in robotics. We demonstrate that in this setting, standard zero-shot RL methods can fail to represent downstream reward functions well, as large approximation errors for the shared reward structures can dominate task-specific reward information during inference. We address this problem by learning a reward-aware representation that reduces the burden of representing shared reward structures during inference, enabling better reconstruction of task-specific rewards. We do so by proposing ReAL, a reward-aware generalization of the Laplacian representation. Empirically, we show that compared to reward-agnostic representations, the ReAL representation enables better downstream reward fitting and stronger zero-shot performance in zero-shot goal-reaching tasks.

1 INTRODUCTION

Learning representations is key to reinforcement learning (RL). The successor representation (SR; Dayan, 1993), and its extension in continuous spaces, the successor measure (SM; Blier et al., 2021), represent a state by its future outcomes and summarize the transition dynamics of the environment. It has been shown to be effective in applications such as exploration (Machado et al., 2020; 2023) and reward shaping (Wu et al., 2019; Wang et al., 2021), and to be relevant in decision-making and neuroscience (Stachenfeld et al., 2014; 2017; Momennejad et al., 2017; Gomez et al., 2025). Recently, it has been the key mathematical object behind advances in zero-shot RL (Borsa et al., 2019; Touati & Ollivier, 2021; Touati et al., 2023; Bagatella et al., 2026).

The SM, and thus SM-based representations, are reward-agnostic. The intuition behind such methods is that they capture the underlying structure of the environment. Similarly, it is intuitive to expect agents to benefit from capturing and exploiting information about the reward function when it is also part of the overall environment structure; for example, when it remains unchanged across tasks. The benefits of reward-aware representations have been demonstrated when the agent is learning to perform a single task in the environment (Tse et al., 2025). Examples of inherent reward structures across tasks include the negative rewards associated with energy consumption or wear and tear in robotics, or the negative rewards associated with undesirable or dangerous regions in navigation.

We demonstrate that standard reward-agnostic zero-shot RL methods struggle in the setting where there exist shared reward structures across different task rewards. Standard zero-shot RL methods can be thought of as learning a set of basis functions in the pre-training phase, and then approximating downstream reward functions with the learned basis in the inference phase. We show that when the shared reward component has a large magnitude compared to the task-specific component, approximation errors for the shared component can dominate the task-specific component. When task-specific information is lost, the agent struggles to obtain good zero-shot performance.

The root cause of the above problem is that the burden of approximating downstream reward functions is purely placed on the inference phase. We address this problem based on the idea of offloading part of the burden of approximating downstream reward functions to the representation learned during the pre-training phase. In particular, since the shared reward structures remain

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

consistent across all downstream tasks, we propose to learn representations that capture these shared rewards. After learning such a representation, during the inference phase, it is then easier for the representation to focus on capturing information about the task-specific rewards.

To learn such a reward-aware representation, we propose the reward-aware Laplacian (ReAL) representation, a generalization of the Laplacian representation (Wu et al., 2019). The key idea of the ReAL representation is to use rewards to regulate the magnitude of entries in each learned basis function. Geometrically, it can be viewed as rewards modulating the shape of each learned basis function.

We perform experiments in OGBench (Park et al., 2025), a popular benchmark for zero-shot goal-reaching tasks, where we introduce low-reward regions that stay unchanged across all downstream tasks. The negative rewards for the low-reward regions form the shared reward component, while the varying goal reward forms the task-specific reward component. While the standard Laplacian and FB (Touati & Ollivier, 2021) representations fail to preserve goal information when the magnitude of the low rewards is large, the ReAL representation, by already incorporating the shared reward information, better captures task-specific information. Due to this, we observe that the ReAL representation consistently enables stronger zero-shot performance across different `antmaze` settings, achieving a success rate around 40% higher than baselines on average and a significantly higher return.

2 PRELIMINARIES

We use lowercase symbols (e.g., r, p) to denote functions, calligraphic fonts (e.g., $\mathcal{S}, \mathcal{A}$) to denote sets, bold lowercase symbols (e.g., $\mathbf{e}$) to denote vectors, and bold uppercase symbols (e.g., $\mathbf{I}, \mathbf{P}$) to denote matrices. We index the (i, j)-th entry of a matrix $\mathbf{M}$ by $\mathbf{M}(i, j)$.

The Laplacian representation. A Markov decision process (MDP) is defined as $(\mathcal{S}, \mathcal{A}, r, p)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \to \mathbb{R}$ is the reward function, and $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, where $\Delta(\mathcal{S})$ is the probability simplex over $\mathcal{S}$. Let $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ denote a policy, and $\mathbf{P}^\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ denote the transition probability matrix induced by π and p. As is common in this line of work (Wu et al., 2019; Wang et al., 2021; Gomez et al., 2024), we assume $\mathbf{P}^\pi$ to be symmetric to ensure a real eigensystem. The Laplacian is defined as $\mathbf{L}^\pi = \mathbf{I} - \mathbf{P}^\pi$. Note that π is often the behavior policy used to collect the dataset used to learn the Laplacian representation. We drop the superscript π hereafter for simplicity. Let $\mathbf{e}_i$ denote the eigenvector corresponding to the i-th smallest eigenvalue of the Laplacian. The Laplacian representation with dimension d is defined as $[\mathbf{e}_1(s), \ldots, \mathbf{e}_d(s)]^\top$ and can be learned using variants of the graph drawing objective (Wu et al., 2019; Wang et al., 2021), and, specifically, the augmented Lagrangian Laplacian objective (ALLO; Gomez et al., 2024):

$$\max_{\boldsymbol{\beta}} \min_{\mathbf{u} \in \mathbb{R}^{\mathbf{d} \times |\mathcal{S}|}} \text{ALLO}(\boldsymbol{\beta}, \mathbf{u}) = \max_{\boldsymbol{\beta}} \min_{\mathbf{u} \in \mathbb{R}^{\mathbf{d} \times |\mathcal{S}|}} \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle + \sum_{j=1}^{d} \sum_{k=1}^{j} \beta_{jk} (\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk}) + b \sum_{j=1}^{d} \sum_{k=1}^{j} (\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk})^2, \tag{1}$$

where $\boldsymbol{\beta} = [\beta_{1,1}, \beta_{2,1}, \beta_{2,2}, \ldots, \beta_{d,1}, \ldots, \beta_{d,d}] \in \mathbb{R}^{d(d+1)/2}$ is the vector of dual variables, $\mathbf{u} = [\mathbf{u}_1^\top, \ldots, \mathbf{u}_d^\top]^\top$ is the concatenation of the learned vectors, δ_{jk} is the Kronecker delta, $[\![\cdot]\!]$ is the stop-gradient operator, and $b \in (0, \infty)$ is a hyperparameter. The solution to the optimization problem, $\mathbf{u}_1, \ldots, \mathbf{u}_d$, is the eigenvectors of the Laplacian corresponding to the smallest d eigenvalues.

The Laplacian representation has been widely applied, such as for reward shaping (Wu et al., 2019; Wang et al., 2021), option discovery (Klissarov & Machado, 2023), planning (Shehmar et al., 2026), and zero-shot RL (Touati et al., 2023; Chandrasekar & Machado, 2026). The Laplacian representation is reward-agnostic. In this work, we extend it to incorporate reward information.

Zero-Shot RL. The goal in the standard zero-shot reinforcement learning setting is to obtain a near-optimal policy for downstream tasks without any additional learning or planning (Touati et al., 2023). Standard zero-shot RL methods usually consist of two phases.

In the pre-training phase, a dataset of reward-free transitions is used to learn a representation encoder $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$ and a policy $\pi(a|s, \mathbf{z})$, with $\mathbf{z} \in \mathbb{R}^d$, trained with the universal successor feature

approximator (USFA; Borsa et al., 2019). The representation encoder maps every state s to the corresponding representation $\boldsymbol{\phi}(s)$. Let $\boldsymbol{\Phi}$ be the matrix where each row is $\boldsymbol{\phi}(s)$ for a different s. The columns of $\boldsymbol{\Phi}$ can be viewed as a set of basis functions. Therefore, learning $\boldsymbol{\phi}$ can equivalently be seen as learning a set of basis functions. Each $\mathbf{z}$ can then be thought of as the d weights to combine these basis functions to form a reward function, i.e., $\mathbf{r}_\mathbf{z} = \boldsymbol{\Phi}\mathbf{z}$ or $r_\mathbf{z}(s) = \boldsymbol{\phi}(s)^\top \mathbf{z}$. The policy, π, is trained via off-policy learning to maximize the reward $\mathbf{r}_\mathbf{z}$ for a wide range of $\mathbf{z}$. If the training is perfect, π can solve any reward functions $\mathbf{r} \in \{\boldsymbol{\Phi}\mathbf{z} \mid \mathbf{z} \in \mathbb{R}^d\}$.

In the inference phase, for a downstream task, we are given a small dataset of reward samples $\mathcal{D}^i = \{s, r(s)\}$. We then compute $\mathbf{z}$ that combines the learned basis functions to approximate the given reward samples, i.e., $\boldsymbol{\phi}(s)^\top \mathbf{z} \approx r(s)$, usually by $\mathbf{z} = \mathbb{E}_{\mathcal{D}^i}[\boldsymbol{\phi}(s) r(s)]$. The policy $\pi(\cdot|\cdot, \mathbf{z})$ is then used to solve the task, where the optimality of this policy is governed by how well the set of learned basis functions can represent the downstream reward functions.

3 Zero-Shot RL with Shared Reward Component

In zero-shot RL, it is intuitive to expect agents to benefit from capturing and exploiting information about the reward function when it is also part of the overall environment structure. In this paper, we focus on the setting in which, across different tasks, there are shared reward structures that reflect inherent environmental properties, e.g., the negative rewards of wear and tear in robotics.

3.1 Zero-Shot RL with Shared Reward Component

While there can be different formalizations of the setting where there exist shared reward structures across tasks, we focus on the zero-shot RL setting where we assume that any downstream reward can be viewed as the sum of a permanent component that is shared for all downstream rewards, and a task-specific transient component. The simplicity of this setting enables us to take a first step at studying the benefits of reward-aware representations under shared reward structures.

We use ZESP to refer to this *ZEro-shot RL setting with a Shared Permanent component*. For ZESP, during the pre-training phase, instead of observing a dataset of reward-less transitions, the agent observes a dataset of transitions with only the permanent rewards. For the inference phase, similar to the standard zero-shot RL setting, the agent only observes the full downstream rewards, and cannot observe any of the permanent or transient components in isolation.

3.2 Standard Zero-shot RL Methods Fail in ZESP

For pedagogical purposes, here we specifically focus on zero-shot goal-reaching tasks in a maze with a low-reward region. The low-reward region is fixed in the maze and does not vary with the goal, and thus the negative rewards associated with it form the permanent component. The varying goal reward forms the transient component. We create such a problem by introducing a low-reward region to the large maze in OGBench (Park et al., 2025), visualized in Fig. 5 (right) in Appendix A.1. We use the provided `antmaze-large-navigate` dataset, and relabel the dataset with the permanent reward (-9 for states in the low-reward region, and 0 otherwise). Any downstream reward function gives 0 for the goal, -10 for states in the low-reward region, and -1 otherwise.

In the ZESP setting, even though the dataset offers information about the permanent reward component, standard zero-shot RL methods cannot make use of it, and place the burden of fitting downstream reward functions purely on the inference phase. These methods can be thought of as learning a set of reward-agnostic basis functions from the dataset. The zero-shot performance of these methods depends on how well the set of basis vectors linearly fits downstream reward functions.

Reward-agnostic bases struggle to fit downstream reward functions in this maze setting. Fig. 1 shows one ground-truth downstream reward function and the reward function reconstructed using two standard zero-shot RL methods, the Laplacian and the forward-backward (FB; Touati & Ollivier, 2021) representations. We can observe that the reconstructed reward functions for these reward-agnostic representations fail to assign the maximum value to the goal, and instead assign the maximum value to another position. This happens because the weights for the linear fit, $\mathbf{z}$, are computed using $\mathbf{z} = \mathbb{E}_{\mathcal{D}^i}[r(S)\boldsymbol{\phi}(S)]$, and under the reward decomposition, can be written as $\mathbf{z} = \mathbb{E}_{\mathcal{D}^i}[r_p(S)\boldsymbol{\phi}(S)] + \mathbb{E}_{\mathcal{D}^i}[r_t(S)\boldsymbol{\phi}(S)]$, where r_p, r_t are the permanent and transient components,

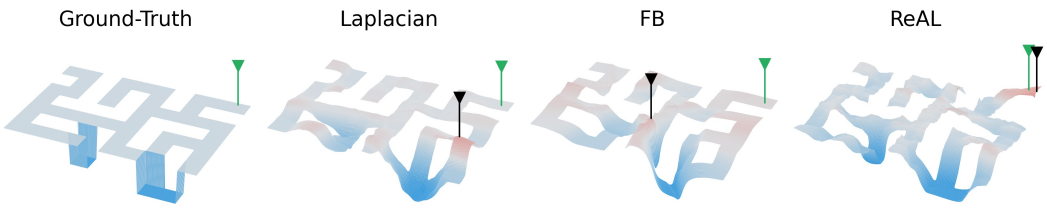


Figure 1: The ground-truth downstream reward function for `antmaze-large` task 1 with a low-reward region, and the reward functions reconstructed using the Laplacian, FB, and ReAL representations with 50 dimensions respectively. The green marker shows the ground-truth goal location, while the black marker shows the position with the maximum reconstructed reward.

respectively. Therefore, the full fitted reward is simply the sum of the fitted permanent part and the fitted transient part, the former capturing the low-reward region in this case, and the latter capturing the goal location. When capturing the permanent component (a step-like function that gives -9 at the low-reward region and 0 everywhere else), the finite set of basis vectors cannot represent it perfectly and creates local approximation errors. Since the permanent component has a larger magnitude compared to the transient component, these approximation errors become large fluctuations that dominate the goal signal in the full fitted reward.

As Laplacian eigenvectors (and, similarly, the backward representation of the FB representation) can be viewed as the Fourier basis of the graph induced by the MDP (Mahadevan, 2005), the above phenomenon can be viewed as the manifestation of the Gibbs phenomenon (Gottlieb & Shu, 1997) in RL. A property of the Gibbs phenomenon is that as the number of basis functions increases, the region of large approximation errors narrows and concentrates closer to the discontinuity of the function we are approximating (in this case, the boundary of the low-reward region), but the magnitude of the approximation error near the discontinuity does not shrink to 0. Therefore, naively increasing the number of basis functions cannot remove the local overshoot that obscures the goal signal.

We demonstrate this issue by computing the goal localization rate for the Laplacian and FB representations, which measures how well the fitted reward function preserves goal information. Fig. 2 (left) shows four distinct downstream goal locations for the large maze. We consider the localization of the goal to be successful if the global maxima of the fitted reward functions using the representations are close enough to the ground-truth goal position (within the dashed circles). Since the magnitude of the permanent reward component also affects the localization rate, we report the localization rate averaged over the cases when the low-reward region leads to rewards of $-1, -2, \ldots, -10$.

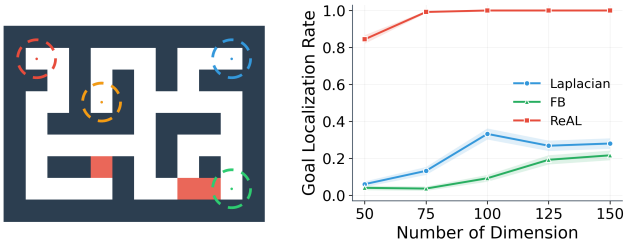


Figure 2: *Left*: Distinct downstream goal locations for the large maze in OGBench (Park et al., 2025). We consider goal information to be preserved if the fitted reward function peaks within the dashed circle around the ground-truth goal. *Right*: The goal localization rate for the Laplacian, FB, and ReAL representations at different dimensions.

Fig. 2 (right) shows the average goal localization rate for the Laplacian and FB representations as the representation dimension increases from 50 to 150. It can be seen that while increasing the number of dimensions, and thus the number of basis functions, initially leads to better preservation of goal information, such benefits quickly plateau.

4 Reward-Aware Representations Under Shared Rewards

The issue with reward-agnostic representations is that all burden for fitting downstream reward functions is placed solely on the inference phase, and these representations are expected to fit both

the shared reward structures and task-specific rewards well, which, as we have seen in the previous section, can be hard. We propose to address this problem by offloading some of the burden of representing downstream rewards to the learned representation. Specifically, we propose to learn a reward-aware representation to capture information about the shared reward structures. With the representation already being aware of the shared reward structures, it should be able to capture task-specific information more easily than reward-agnostic representations.

4.1 The Reward-Aware Laplacian Representation

We now introduce the *Reward-Aware Laplacian (ReAL) representation*, a generalization of the Laplacian representation that uses reward information to modulate its magnitude.

Definition 4.1 (Reward-Aware Laplacian)**.** The reward-aware Laplacian is $\mathbf{L}^r = \mathbf{I} - \mathbf{P}^\pi + c\mathbf{R}$, where r is the reward function whose information to incorporate, $\mathbf{R}$ is a diagonal matrix with $\mathbf{R}(i,i) = f(r(s_i)/\lambda)$ for a function $f : \mathbb{R} \to \mathbb{R}$, and $c \geq 0, \lambda > 0$ are hyperparameters.

Definition 4.2 (ReAL Representation)**.** Let $\mathbf{e}_i^r$ denote the eigenvector corresponding to the i-th smallest eigenvalue of $\mathbf{L}^r$. The ReAL representation is $[\mathbf{e}_1^r(s), \ldots, \mathbf{e}_d^r(s)]$.

The ReAL representation is a generalization of the standard Laplacian representation:

Proposition 4.3. *When $c = 0$, or when the reward function is constant, the ReAL representation reduces to the Laplacian representation.*

Proof. The proposition trivially holds when $c = 0$, as the reward-aware Laplacian simply reduces to the standard Laplacian. When the reward function is constant, $c\mathbf{R} = a\mathbf{I}$ for some constant a, and so the reward-aware Laplacian is simply $a\mathbf{I}$ added to the standard Laplacian. Since adding a multiple of the identity matrix only shifts the eigenvalues and does not change the eigenvectors, the ReAL representation reduces to the Laplacian representation. □

We can learn the eigenvectors of the reward-aware Laplacian with the following objective:

Definition 4.4 (RALLO)**.** The reward-aware variant of the ALLO objective (Eq. 1), *RALLO*, is

$$\max_{\boldsymbol{\beta}} \min_{\mathbf{u} \in \mathbb{R}^{\mathbf{d} \times |\mathcal{S}|}} \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{L}^r \mathbf{u}_i \rangle + \sum_{j=1}^{d} \sum_{k=1}^{j} \beta_{jk} (\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk}) + b \sum_{j=1}^{d} \sum_{k=1}^{j} (\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk})^2$$

$$= \max_{\boldsymbol{\beta}} \min_{\mathbf{u} \in \mathbb{R}^{\mathbf{d} \times |\mathcal{S}|}} \text{ALLO}(\boldsymbol{\beta}, \mathbf{u}) + c \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{R}\mathbf{u}_i \rangle. \tag{2}$$

As $\mathbf{u}_1, \ldots, \mathbf{u}_d$ optimized with RALLO approximates the ground-truth ReAL eigenvectors, $\mathbf{e}_1^r, \ldots, \mathbf{e}_d^r$, the learned ReAL representation is simply $[\mathbf{u}_1(s), \ldots, \mathbf{u}_d(s)]$.

The only difference between the RALLO and ALLO objectives is the extra inner product terms, $\langle \mathbf{u}_i, \mathbf{R}\mathbf{u}_i \rangle = \mathbb{E}_{S \sim \rho}[f(r(S)/\lambda)\mathbf{u}_i(S)^2]$, where ρ denotes the state distribution induced by the behavior policy that generated the dataset used for RALLO. These terms use the reward of a state, $r(s)$, to modulate the magnitudes of the corresponding eigenvector entries, $\mathbf{u}_i(s)$, and are the source of the reward-awareness of the ReAL representation. The hyperparameter c controls the strength of reward-awareness, while λ determines how sensitive the magnitude modulation is to the rewards.

Choice of f. The function f determines how the state rewards affect the magnitude modulation. While one is free to choose any f, in this paper, we make the following design choices: (1) We have chosen f to be non-negative because, since the Laplacian is positive semi-definite (PSD), a non-negative f will also ensure the reward-aware Laplacian is also PSD. (2) We have chosen f to be decreasing. As a result, the magnitudes of the eigenvector entries for low-reward states will be suppressed. We have chosen to suppress $\mathbf{u}_i(s)$ for low-reward s based on the intuition that each normalized eigenvector can be thought of as distributing a fixed amount of mass among different states. Suppressing the magnitudes of low-reward states corresponds to distributing less mass for low-reward states, meaning that more mass is allocated to distinguishing between high-reward states, which are likely more important for the agent.

Given these design choices, we have chosen $f(x) = \sigma(-x)$, where σ is the sigmoid function, which is also bounded and robust to different input values. To provide a visualization of the magnitude

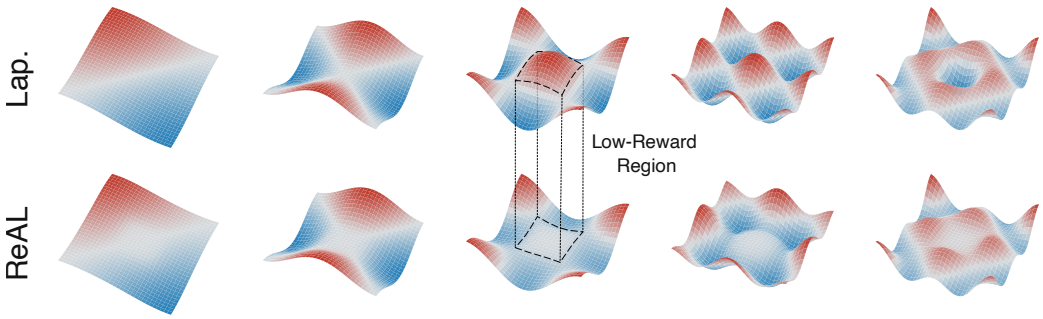


Figure 3: The eigenvectors of the standard Laplacian (top) and reward-aware Laplacian (bottom) for a square open room with a low-reward region (-1 reward) in the middle.

modulation effect, we show in Fig. 3 the ReAL eigenvectors using $f(x) = \sigma(-x)$ for an open room where the reward is -1 for the low-reward region in the middle of the room and 0 otherwise.

Relationships with other representations. It is known that the SR and the Laplacian share the same eigenvectors (Stachenfeld et al., 2014; Machado et al., 2018). The default representation (DR) is a reward-aware generalization of the SR (Piray & Daw, 2021; Tse et al., 2025), while the reward-aware Laplacian is a reward-aware generalization of the Laplacian. It therefore does not come as a surprise that the objective for learning the ReAL representation is similar to that for learning the principal eigenvector of the DR (Tse & Machado, 2026). However, the DR is defined in the restricted class of linearly solvable MDPs (Todorov, 2009). It assumes the existence of terminal states and, by definition, corresponds to choosing $f(x) = \exp(-x)$. These two properties of the DR jointly induce principal eigenvectors in the DR's eigensystem that have poor numerical properties, requiring significant effort to learn in practice, and limiting the learning of subsequent eigenvectors (Tse & Machado, 2026). Unlike the DR, the reward-aware Laplacian is defined in any standard MDP without the assumption of terminal states, allowing it to be applied much more broadly. It is also not restrictively defined to select the exponential as f, and is free to choose a better-behaved f, such as the sigmoid. These key differences remove the principal eigenvectors with poor numerical properties from the ReAL's eigensystem, enabling multiple eigenvectors to be learned easily. Note, finally, that the reward-aware Laplacian can be viewed as an instantiation of the source Laplacian (Theophilou & Şimşek, 2026) where the standard Laplacian is augmented with a diagonal matrix capturing reward information. Theophilou & Şimşek (2026) evaluated the source Laplacian with a diagonal matrix capturing state visitation counts, and their results were limited to the tabular case.

4.2 The ReAL Representation Under Shared Reward Structures

We now demonstrate that the ReAL representation is indeed able to capture the reward structure that is shared across tasks. Specifically, we apply the ReAL representation in the ZESP setting. Geometrically, by capturing permanent reward information, the shapes of the ReAL basis functions are already closer to that of the permanent reward component. Since all downstream rewards share the same permanent component, it is then easier to combine these reward functions to reconstruct the shape of the downstream rewards.

Fig. 1 (rightmost) shows the fitted reward function using the ReAL representation. We can see that, unlike reward-agnostic representations, the ReAL representation is much more effective at recovering information about the goal location. We can also observe that, since the ReAL basis functions capture permanent reward information, they reconstruct the low-reward region much better. We observe, however, that apart from the goal and low-reward regions, the reward surface constructed using ReAL basis functions is overall less smooth than those constructed using the Laplacian and FB representations. We speculate that it is because the Laplacian eigenvectors are, by construction, the smoothest functions on the graph induced by the MDP. When we introduce reward-awareness, we are inevitably sacrificing some of the smoothness. We therefore view the ReAL representation as trading off between the smoothness of the basis functions and the fidelity

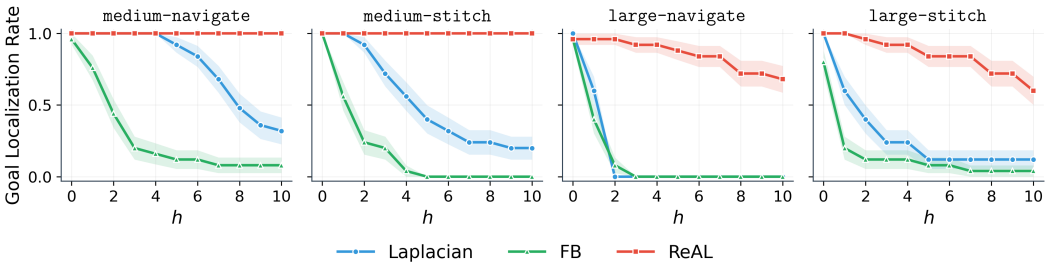


Figure 4: The goal localization rate for different representations as h increases, averaged over 5 runs. The shaded region indicates the std. error.

of task-specific information. Finally, Fig. 2 shows the goal localization rate for the ReAL representation at different dimensions. We can see that the ReAL representation achieves a significantly higher goal localization rate compared to reward-agnostic representations.

5 Experiments

In this section, we show, quantitatively, that (1) the ReAL representation enables better downstream reward fitting than reward-agnostic representations in a broader set of settings, and that (2) better downstream reward fitting using the ReAL representation leads to better zero-shot performance.

We use `antmaze` environments from OGBench (Park et al., 2025), a popular benchmark for zero-shot goal-reaching tasks. We augment the available mazes with a reward function that is negative in parts of the state space and zero otherwise. This function forms the permanent reward component of the problem (see Appendix A for details). We vary the magnitude of the permanent component to examine its effect on downstream reward fitting and zero-shot performance. That is, the permanent reward component is $-h\mathbb{I}_{\{\text{agent in low-reward region}\}}$, where $h \geq 0$ is a constant and $\mathbb{I}$ is the indicator function. The transient component is simply 0 at the goal and -1 otherwise, with the goal location varying across tasks. For the pre-training phase, we relabel transitions in the `antmaze` datasets with the permanent component. To ensure that our findings are not restricted to a single `antmaze` setting, we consider the `antmaze` datasets $\{\texttt{medium}, \texttt{large}\} \times \{\texttt{navigate}, \texttt{stitch}\}$. For simplicity, we follow the naming convention of a prior work (Bagatella et al., 2026). For example, we use `antmaze-mn` to refer to `antmaze-medium-navigate`.

We compare the ReAL representation with two standard zero-shot RL methods: the Laplacian representation (Wu et al., 2019; Gomez et al., 2024; Chandrasekar & Machado, 2026) and the forward-backward representation (FB; Touati & Ollivier, 2021) at 50 dimensions. Details on the architecture, hyperparameters, and algorithmic details are in Appendix A.2. As supporting evidence that our implementations of the Laplacian and FB representations are correct, we present their success rates in standard OGBench `antmaze` (without low-reward regions) in Appendix B.1. Note that our implementations significantly outperform those in a recent work (Bagatella et al., 2026).

5.1 Downstream Reward Reconstruction

To measure how well different representations preserve goal information, similar to Section 3, we measure the goal localization rate. Specifically, we use each representation to fit each downstream reward function, and consider goal information to be preserved if the fitted downstream reward function peaks at a position whose distance to the ground-truth goal location is below 3.5 units (see Fig. 2 left). We report the goal localization rate averaged over downstream tasks for each representation.

Fig. 4 shows the average goal localization rate as h increases for different representations and `antmaze` settings. We can observe that while the goal localization rate decreases as h increases in general, the ReAL representation degrades much more gracefully compared to reward-agnostic representations, consistently having higher localization rates. This is because it is easier for the ReAL representation to capture task-specific information, as it already captures part of the information of the downstream reward functions. We further visualize, for each representation, the reconstructed reward function for different h and different tasks in `antmaze-ln` in Appendix B.2.

Table 1: The difference in (top) success rate (%) and (bottom) undiscounted return between the ReAL representation and (left) the Laplacian representation and (right) the FB representation for the four `antmaze` settings, averaged over 5 runs. The actual mean success rate (and std. error) for each method is available in Appendix B.3.

	ReAL − Laplacian					ReAL − FB				
h	0	1	2	3	4	0	1	2	3	4
				Success Rate (%)						
antmaze-mn	8.0	45.2	73.6	80.0	82.4	24.0	9.6	42.4	61.6	68.8
antmaze-ms	−1.2	72.8	59.2	54.0	33.6	31.6	63.2	54.8	48.0	30.8
antmaze-ln	8.8	72.8	58.8	46.8	38.0	40.4	66.4	62.4	47.6	38.4
antmaze-ls	9.6	27.6	22.4	12.0	6.4	20.8	28.0	22.0	13.2	7.6
average	6.3	54.6	53.5	48.2	40.1	29.2	41.8	45.4	42.6	36.4
				Undiscounted Return						
antmaze-mn	94	424	621	639	607	217	145	390	526	523
antmaze-ms	38	572	437	376	244	360	520	400	338	234
antmaze-ln	93	413	284	168	172	292	380	276	153	134
antmaze-ls	58	178	127	85	44	124	158	102	50	12
average	71	397	367	317	267	248	301	292	267	226

5.2 Zero-Shot Performance

As we have shown, using the ReAL representation enables better downstream reward fitting, in particular, better preservation of goal information. We now show that this leads to actual performance gains. As it is common in zero-shot goal-reaching evaluations, we report the success rate for reaching downstream goals. As the downstream reward functions now do not simply contain the goal reward, we also report the undiscounted returns for the evaluation episodes.

Table 1 (top) shows the difference in the success rates between the ReAL representation and (left) the Laplacian representation and (right) the FB representation. When $h = 0$ (when there is no low-reward region in the downstream reward function), the ReAL representation has comparable or slightly higher success rates than the best performing reward-agnostic representation. The only case where the ReAL representation has a lower success rate than baselines is for `antmaze-ms` when $h = 0$, but the difference is small and could be attributed to noise. Note that a possible reason for the ReAL representation to have slightly higher success rates than the best performing reward-agnostic representation, even when $h = 0$, is that for each representation and for each dataset, we train a single USFA, and thus use the same trained actor to generate behavior for all h (see details in Appendix A.2.3). The behavior generated for different h still differs since different h induces different $\mathbf{z}$ that the actor conditions on. However, since reward-agnostic representations cannot distinguish the goal well for $h > 0$, the behavior learned for $h > 0$ could degrade the behavior learned for $h = 0$ through shared neural network parameters. On the other hand, the ReAL representation has significantly higher success rates when $h > 0$, often exceeding the success rates for reward-agnostic baselines by at least 40% on average. By observing the success rates for the individual methods, we can see that the Laplacian and FB representation's success rates (see Table 7 and 8 in Appendix B.3) drop significantly when h increases, while that for the ReAL representation (see Table 9 in Appendix B.3) degrades much more gracefully, matching the trend for the goal localization rate.

One phenomenon we observe is that the success rates we report can be slightly higher than the goal localization rates reported in Fig. 4, e.g., for the FB representation for `antmaze-mn`. We confirmed that there are two reasons for this: (1) The distance threshold by which we determine whether the goal is localized is a hard threshold that returns a binary value, and the computed distance can lie just slightly above the threshold. (2) The goal localization rate only takes into account the position where the fitted reward function takes its global maximum value, while there could be other local maxima in the fitted reward function. When the ground-truth goal location corresponds to a local maximum, it is still possible for the agent to reach the goal, especially if the agent's start location is close to the local maxima. Apart from these reasons, we hypothesize that behavior cloning when

training the USFA could also be a cause. It is possible for the agent to mimic behavior present in the dataset that enables it to occasionally reach the goal, even though goal information is lost.

Table 1 (bottom) shows the difference in the undiscounted return between the ReAL representation and (left) the Laplacian representation and (right) the FB representation. Similar to our observations with the success rates, we observe that the ReAL representation achieves significantly higher returns compared to the reward-agnostic representations.

We additionally report the success rates and undiscounted returns for the Laplacian and FB representations at 100 dimensions in Appendix B.4. While there are minor improvements compared to 50 dimensions, they still significantly underperform the ReAL representation, confirming our earlier discussion that simply increasing the number of dimensions does not address the ZESP setting.

6 Related Work

Zero-shot RL. This line of work builds on the successor representations (SR; Dayan, 1993), successor features (SF; Barreto et al., 2017), and successor measures (SM; Blier et al., 2021), learning a representation and USFAs (Borsa et al., 2019) to obtain general agents capable of solving a wide range of tasks. The main difference among most methods is what representations these agents use. The Laplacian representation (Wu et al., 2019) can be viewed as learning a Fourier basis for the graph induced by the MDP (Mahadevan, 2005). Forward-Backward representations (Touati et al., 2023; Tirinzoni et al., 2025) learn features to represent the SM of a class of policies. HILP (Park et al., 2024) learns features to linearly represent goal-reaching value functions. TD-JEPA (Bagatella et al., 2026) and RLDP (Jajoo et al., 2025) learn predictive latent regularized representations to model future states using a contrastive loss. We refer the readers to (Touati et al., 2023; Di Ventura et al., 2025) for an overview of zero-shot RL.

Reward-aware representations. Most representation learning methods for RL exploit the reward information from the single task that they are trained on. Bisimulation-based methods learn representations that preserve both the transition dynamics and rewards (Givan et al., 2003; Ferns et al., 2004; Ravindran, 2004; Zhang et al., 2021; Castro et al., 2021). Other approaches, such as DeepMDP (Gelada et al., 2019) and value-equivalent models (Grimm et al., 2020), explicitly optimize latent representations to preserve reward prediction and value functions. Among related works, our method is closely related to default representation (DR; Piray & Daw, 2021; Tse et al., 2025) and its extension to function approximation (Tse & Machado, 2026). The ReAL representation, unlike the DR, is not restricted to linearly-solvable MDPs (Todorov, 2009), and can be more broadly applied in any standard MDPs for any reward functions. Finally, as discussed in Section 4.1, the reward-aware Laplacian can be viewed as an instantiation of the source Laplacian (Theophilou & Şimşek, 2026; Vijayan et al., 2026) that augments the standard Laplacian with reward information.

7 Conclusion

In this paper, we have shown that when there are shared reward structures across different tasks, standard zero-shot RL methods can fail. Specifically, in the zero-shot RL setting with a shared permanent reward component (ZESP), these methods fail to capture task-specific information, leading to poor zero-shot performance. The problem with these methods is that they place all the burden for fitting downstream reward functions in the inference phase. Leveraging the idea of offloading part of such a burden to the learned representation, we propose to capture shared reward structures by learning a reward-aware representation. To do so, we proposed the reward-aware Laplacian (ReAL) representation, a generalization of the Laplacian representation that uses reward information to modulate its magnitude. Using the ReAL representation to capture shared reward structures enables preserving task-specific information in the fitted reward functions much more reliably, leading to significant improvements in zero-shot performance.

Future work can consider applying the reward-aware Laplacian to other applications where the standard Laplacian has proven useful, such as reward shaping (Wu et al., 2019; Wang et al., 2021), option discovery (Klissarov & Machado, 2023), and planning (Shehmar et al., 2026). Moreover, we introduced reward-awareness only into the Laplacian representation, and future work can extend reward-awareness to other zero-shot RL methods.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on experiments and supporting data analysis, mainly through implementing code to visualize algorithm training outcomes and aggregate training results across different random seeds. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, proposing or refining hypotheses, assisting with translation, cleaning and reformatting datasets, or interpreting results. Additionally, we used generative AI tools for creating or modifying scientific figures and editing software code. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide experiment details in Appendix A, including the environment and dataset specification, neural network architectures, hyperparameter tuning procedure, final hyperparameters used, and important algorithmic details. Code will be released upon paper acceptance to facilitate reproducibility.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *CoRR*, abs/2101.07123, 2021.

Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Rémi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for Markov decision processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Siddarth Chandrasekar and Marlos C Machado. The Laplacian keyboard: Beyond the linear span. *CoRR*, abs/2602.07730, 2026.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Jacopo Di Ventura, Jan Felix Kleuker, Aske Plaat, and Thomas Moerland. A unified framework for zero-shot reinforcement learning. In *Reinforcement Learning Beyond Rewards Workshop at the Reinforcement Learning Conference (RLBrew)*, 2025.

Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2004.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning (ICML)*, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003.

Diego Gomez, Michael Bowling, and Marlos C. Machado. Proper Laplacian representation learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Diego Gomez, Michael Bowling, J Quinn Lee, and Marlos C Machado. Representations in the hippocampal-entorhinal system emerge from learning sensory predictions. *bioRxiv*, 2025.

David Gottlieb and Chi-Wang Shu. On the Gibbs phenomenon and its resolution. *SIAM review*, 39 (4):644–668, 1997.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Pranaya Jajoo, Harshit Sikchi, Siddhant Agarwal, Amy Zhang, Scott Niekum, and Martha White. Regularized Latent Dynamics Prediction is a Strong Baseline for Behavioral Foundation Models. In *Reinforcement Learning Beyond Rewards Workshop at the Reinforcement Learning Conference (RLBrew)*, 2025.

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

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations (ICLR)*, 2018.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Marlos C. Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80):1–69, 2023.

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *International Conference on Machine learning (ICML)*, 2005.

Ida Momennejad, Evan M. Russek, Jin H. Cheong, Matthew M. Botvinick, Nathaniel D. Daw, and Samuel J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680–692, 2017.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation Policies with Hilbert Representations. In *International Conference on Machine Learning (ICML)*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

Payam Piray and Nathaniel D. Daw. Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature Communications*, 12(1):4942, 2021.

Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

Dikshant Shehmar, Matthew Kyle Schlegel, Matthew E. Taylor, and Marlos C. Machado. Laplacian representations for decision-time planning. In *International Conference on Machine Learning (ICML)*, 2026.

Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. Design principles of the hippocampal cognitive map. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017.

Andreas Theophilou and Özgür Şimşek. Novel exploration via orthogonality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Learning Representations (ICLR)*, 2025.

Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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

Hon Tik Tse and Marlos C Machado. DROGO: Default representation objective via graph optimization in reinforcement learning. *CoRR*, abs/2602.00403, 2026.

Hon Tik Tse, Siddarth Chandrasekar, and Marlos C. Machado. Reward-aware proto-representations in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Priyesh Vijayan, Élodie Côté-Gauthier, Mathieu Reymond, Sarath Chandar, Doina Precup, and Isabeau Prémont-Schwarz. Addressing sparse-rewards in RL with scalable hierarchical novel eigen options. In *Reinforcement Learning Beyond Rewards Workshop at the Reinforcement Learning Conference (RLBrew)*, 2026.

Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better Laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning (ICML)*, 2021.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations (ICLR)*, 2019.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations (ICLR)*, 2021.

A EXPERIMENT DETAILS

We report experimental details not covered in the main text, including the environments considered and all implementation details.

A.1 ENVIRONMENTS

We consider the `ant` locomotion task from OGBench (Park et al., 2025), which requires controlling an 8-DoF ant. The agent has access to the full 27-dimensional state along with the x-y position in the maze. We used two dataset variants, `navigate` and `stitch`, collected through a directional policy trained via soft actor critic (SAC; Haarnoja et al., 2018). In the `navigate` dataset, the agent repeatedly moves towards randomly sampled goals across the maze. In contrast, the `stitch` dataset consists of shorter trajectories, testing the agent's trajectory-stitching capability.

We consider the `medium` and `large` maze layouts from OGBench, and added low-reward regions (colored *red* in Fig. 5) to form the permanent reward component. These low-reward regions are selected so that there is a feasible path for the agent to solve all downstream tasks, while the agent must take some detours to reach the goal. These low-reward regions are fixed in the maze and do not vary with the goal; thus, the negative rewards associated with them form the permanent reward component, while the varying goal reward forms the transient reward component. For pre-training, we use the standard OGBench datasets and relabel them with the permanent reward (-9 for states in the low-reward region, and 0 otherwise). Note that before training the ReAL encoder, we normalize the permanent reward component over the dataset to have zero mean and unit variance. During inference, the transient, task-specific component for any downstream reward function assigns 0 to the goal and -1 to all other states. We vary the magnitude of the permanent component to examine its effect on downstream reward fitting and zero-shot performance. That is, the permanent reward component is $-h\mathbb{I}_{\{\text{agent in low-reward region}\}}$, where $h \geq 0$ is a constant and $\mathbb{I}$ is the indicator function.



Figure 5: OGBench `medium` (left) and `large` (right) maze with low-reward regions colored in *red*.

An episode ends when the agent comes within a distance of 1.0 of the goal position, deemed as successfully reaching the goal, or when the episode time limit (1000 steps) is reached. The agent is evaluated on 5 state-goal pairs pre-defined in OGBench, with a slightly randomized start position upon each episode reset. We report the average success rate across 5 tasks, using 5 random seeds and 10 evaluation episodes per task.

A.2 IMPLEMENTATION DETAILS

In this section, we present implementation details for the Laplacian, FB, and ReAL representations. We train these representations at dimension $d = 50$. The Laplacian, FB, and ReAL representations share the same components: (1) a representation encoder, ϕ, and (2) a universal successor feature approximator (USFA; Borsa et al., 2019). While the Laplacian and ReAL representations train the encoder and USFA separately, the FB representation trains them jointly.

A.2.1 REPRESENTATION ENCODER

Architecture. We use the same architecture for the representation encoder, ϕ, across all representations. The representation encoder takes the observation as input, and outputs the corresponding representation. The input observations are normalized using the mean and standard deviation over the dataset. The neural network consists of 4 hidden layers with 512 hidden units: a linear layer followed by layer normalization (Ba et al., 2016) and the tanh activation, followed by three linear layers followed by the ReLU activation (Nair & Hinton, 2010).

Hyperparameters. For the encoder, we sweep over the step size $\in \{5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}\}$, and the batch size $\in \{256, 512, 1024\}$. We pick the best hyperparameters based on the goal localization rate. We report common hyperparameters shared for all representations in Table 2. We report ALLO- or RALLO-specific hyperparameters, used for the Laplacian and ReAL representations, in Table 3. We did not tune these hyperparameters, and simply used values suggested by the authors of the ALLO paper (Gomez et al., 2024). Finally, for ReAL specifically, we fixed $\lambda = 1$, and picked $c = 0.5$ from $\{0.1, 0.5, 1.0\}$ by picking the c value that leads to good goal localization rates for both `antmaze-medium-navigate` and `antmaze-large-navigate`.

Table 2: Common encoder hyperparameters shared by all representations.

Hyperparameter	Value
Step size	1×10^{-4}
Batch size	512
Optimizer	Adam
Gradient Steps	1M

Table 3: (R)ALLO-specific encoder hyperparameters for the Laplacian and ReAL representations.

Hyperparameter	Value
Positive-pair Sampling Discount	0.5
Initial dual value	-1.0
Dual Clip Range	$[-100, 100]$
Dual Step Size	1.0
Initial Barrier Coefficient	0.5
Barrier Coefficient Range	[0, 0.5]

A.2.2 Universal Successor Feature Approximator

We train a $\mathbf{z}$-conditioned actor and a USFA-style critic (Borsa et al., 2019) using TD3+BC (Fujimoto & Gu, 2021). Similar to the representation encoder, the input observations are normalized using the mean and standard deviation over the dataset. Additionally, the $\mathbf{z}$ that these networks condition on is normalized to have norm $\sqrt{d}$.

Architecture. The critic takes as input the observation, action, and $\mathbf{z}$, and outputs the successor features for the observation-action pair under the policy optimizing $r_{\mathbf{z}}$. Computing the dot product between the successor features and $\mathbf{z}$ retrieves the Q-values. We first use two small embedding networks to learn the observation-action embedding and the observation-$\mathbf{z}$ embedding. Both embedding networks are two-layer MLPs: a linear layer with 512 hidden units followed by layer norm and tanh, followed by another linear layer with 256 hidden units followed by ReLU. The two 256-dimensional embeddings are then concatenated to form a 512-dimensional vector, which is passed to the rest of the critic network: four linear layers with 512 hidden units followed by ReLU activation, followed by an output layer.

The actor takes as input the observation, the representation of the observation computed using $\boldsymbol{\phi}$, and $\mathbf{z}$, and outputs the action. Note that we apply the stop-gradient operator to $\boldsymbol{\phi}(\text{obs})$ so that the gradient does not flow from the actor to the representation encoder. Similar to the critic, we first use two small embedding networks for the observation-representation pair and the observation-representation-$\mathbf{z}$ triple, before concatenating the two embeddings. The architecture of the embedding networks mirrors that for the critic. The concatenated vector is similarly passed through four hidden layers with 512 units with ReLU, and finally an output layer with a tanh activation. The network output is rescaled to the action range.

Hyperparameters. We sweep over the actor step size $\in \{1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}\}$, the critic step size $\in \{1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}\}$, the actor update frequency $\in \{1, 2\}$, and the

target network $\tau \in \{5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}\}$, and the batch size $\in \{256, 512, 1024\}$. We choose the best hyperparameters that maximize the success rate of the Laplacian representation in standard OGBench, and use those selected hyperparameters for both the Laplacian and ReAL representations for OGBench in the ZESP setting, where low-reward regions are introduced. We report common hyperparameters shared by the Laplacian and ReAL representations in Table 4. For the FB representation, we follow the hyperparameters used in a prior work (Bagatella et al., 2026). Additionally, we observe that the FB representation is sensitive to the orthonormal regularization parameter, so we sweep over $\{1, 100, 500\}$ for each dataset.

Table 4: Common USFA hyperparameters shared by the Laplacian and ReAL representations.

Hyperparameter	**Value**
Discount factor	0.99
Gradient Steps	1M
Batch size	1024
Optimizer	Adam
$\mathbf{z}$-sampling Mixing Ratio	0.5
Critic Step size	1×10^{-4}
Policy smoothing standard deviation	0.2
Policy smoothing truncation level	0.3
Gradient Clipping	1.0
Target network τ	5×10^{-4}
Actor Step Size	5×10^{-5}
Actor Update Frequency	1

A.2.3 Algorithmic Details

Behavior Cloning. For all representations, we use the standard TD3+BC (Fujimoto & Gu, 2021) for policy extraction. For a given $\mathbf{z}$,

$$\pi = \arg\max_{\pi} \mathbb{E}_{s,a\sim\mathcal{D}} \left[Q(s, \pi(s|\mathbf{z})|\mathbf{z}) - \alpha(\pi(s|\mathbf{z}) - a)^2 \right]. \tag{3}$$

The standard zero-shot evaluation pipelines typically use TD3 (Fujimoto et al., 2018) for policy extraction, but are evaluated on a high-coverage dataset. OGBench, on the other hand, provides expert policies, but its action coverage is restricted. TD3 bootstraps on its own learned policy, which can cause actions to drift out of distribution relative to the dataset. Thus, the behavior cloning term regularizes the policy parameters towards the data distribution, with α controlling the strength of the regularization. We conduct a sweep over $\alpha \in \{0.3, 0.6, 1.0\}$ for each dataset and report the best results obtained.

$\mathbf{z}$-sampling. In the ZESP setting, every downstream $\mathbf{z}$ during inference can be expressed as $\mathbf{z}_p + \mathbf{z}_t$, where $\mathbf{z}_p$ and $\mathbf{z}_t$ denote the permanent and transient components respectively. To ensure that the downstream $\mathbf{z}$'s are not out of the training $\mathbf{z}$ distribution, in the pre-training phase, we sample $\mathbf{z}$ that are close to $\mathbf{z}_p$. Specifically, we first use the learned representation and samples from the pre-training dataset to fit $\mathbf{z}_p$. Note that for the FB representation, since the representation encoder and USFA are trained jointly, we periodically re-fit $\mathbf{z}_p$. Then, for any $\mathbf{z}$ sampled in the standard manner (Touati et al., 2023) (either the $\mathbf{z}$ fitting the reward function for a random goal, or a random $\mathbf{z}$ sampled from the standard normal distribution), we use $\mathbf{z} + \mathbf{z}_p$, normalized to have norm $\sqrt{d}$, for training.

Simultaneous training for different h. As mentioned previously, we vary the magnitude of the permanent reward component, h. While we could separately train a USFA for each of the permanent reward magnitude, to save computation, we train a single USFA for all permanent reward magnitudes in the range $[0, 10]$. We can achieve this by slightly modifying $\mathbf{z}$ sampling. Let $\bar{\mathbf{z}}_p$ be the $\mathbf{z}$ fitted for the permanent reward component when its magnitude, h, is 1. We can simply obtain the $\mathbf{z}$'s for other h by $h\bar{\mathbf{z}}_p$. Using this fact, we simply use $\mathbf{z} + b\bar{\mathbf{z}}_p$, normalized to have norm $\sqrt{d}$, to train a single USFA, where $\mathbf{z}$ is sampled in the standard manner (Touati et al., 2023), and b is sampled uniformly random from $[0, 10]$.

B Additional Experiment Results

In this section, we present additional experiment results.

B.1 Standard OGBench Results

As supporting evidence that our implementations of the Laplacian (Gomez et al., 2024) and FB (Touati & Ollivier, 2021) representations are correct, we present their performance for `antmaze` in the standard OGBench in Tables 5 and 6, respectively. Note that our implementations significantly outperform those in a recent work (Bagatella et al., 2026).

Table 5: The mean success rate (%) and the standard error for the Laplacian representation for different `antmaze` settings in standard OGBench, averaged over 5 runs.

Task	1	2	3	4	5	**avg**
`antmaze-mn`	98.0 ± 2.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	98.0 ± 2.0	99.2 ± 0.5
`antmaze-ms`	100.0 ± 0.0	100.0 ± 0.0	98.0 ± 2.0	96.0 ± 2.4	100.0 ± 0.0	98.8 ± 0.8
`antmaze-me`	50.0 ± 9.5	88.0 ± 3.7	54.0 ± 14.0	44.0 ± 11.2	68.0 ± 18.5	60.8 ± 1.9
`antmaze-ln`	100.0 ± 0.0	94.0 ± 2.4	96.0 ± 2.4	90.0 ± 3.2	94.0 ± 6.0	94.8 ± 1.4
`antmaze-ls`	84.0 ± 6.8	14.0 ± 7.5	96.0 ± 2.4	74.0 ± 10.3	50.0 ± 8.4	63.6 ± 3.6

Table 6: The mean success rate (%) and the standard error for the FB representation for different `antmaze` settings in standard OGBench, averaged over 5 runs.

Task	1	2	3	4	5	**avg**
`antmaze-mn`	96.0 ± 2.4	100.0 ± 0.0	100.0 ± 0.0	98.0 ± 2.0	100.0 ± 0.0	98.8 ± 0.5
`antmaze-ms`	94.0 ± 4.0	42.0 ± 19.1	98.0 ± 2.0	92.0 ± 3.7	100.0 ± 0.0	85.2 ± 4.0
`antmaze-me`	16.0 ± 5.1	38.0 ± 11.6	0.0 ± 0.0	0.0 ± 0.0	44.0 ± 10.3	19.6 ± 3.5
`antmaze-ln`	94.0 ± 6.0	78.0 ± 10.7	98.0 ± 2.0	86.0 ± 6.8	86.0 ± 6.0	88.4 ± 2.1
`antmaze-ls`	80.0 ± 7.1	26.0 ± 17.8	92.0 ± 5.8	72.0 ± 15.6	62.0 ± 16.6	66.4 ± 9.0

B.2 Reward Fitting Visualization

We visualize the reward function fitted using the Laplacian, FB, and ReAL representations for `antmaze-large-navigate` in Figures 6, 7, and 8 respectively. We can observe that the ReAL representation preserves the goal location much more reliably.

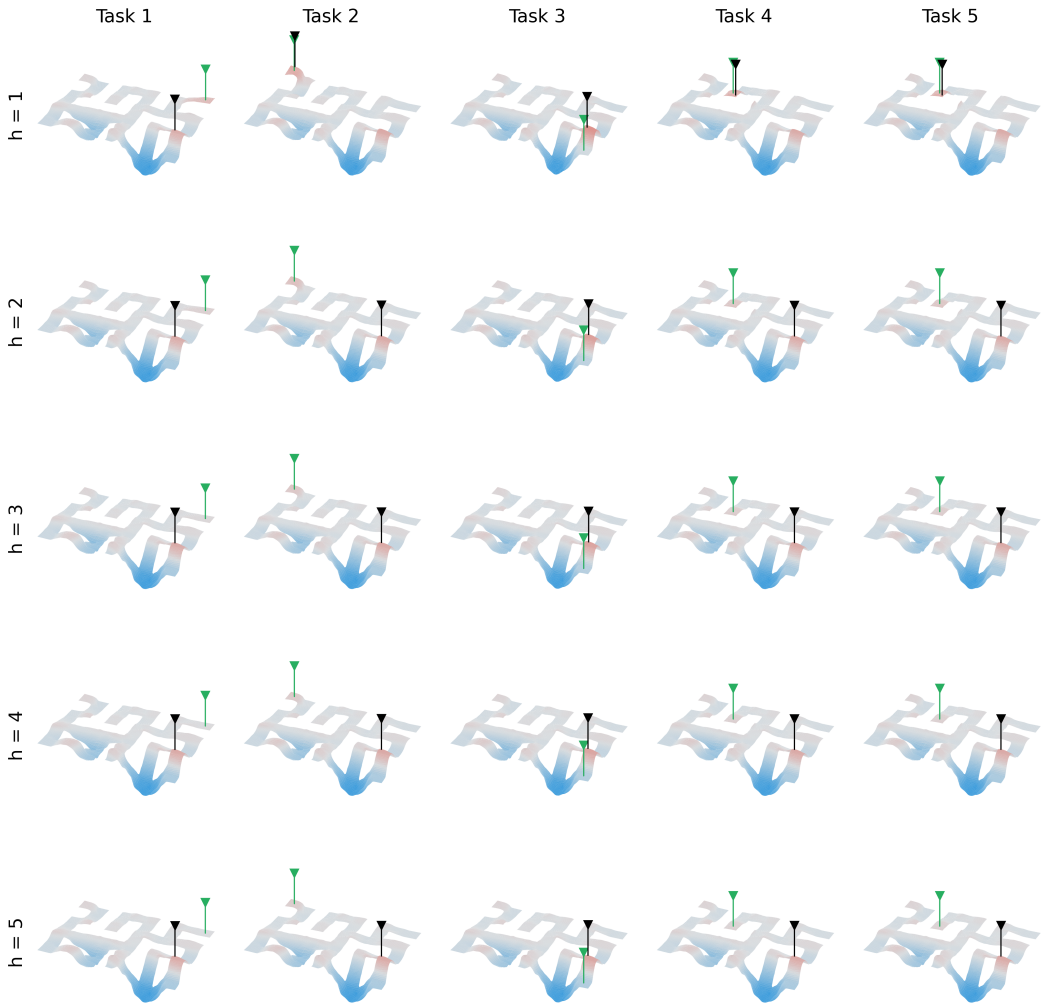


Figure 6: **The fitted reward function using the Laplacian representation** for different h and different tasks in `antmaze-ln`. The green marker shows the ground-truth goal location, while the black marker shows the position with the maximum reconstructed reward.

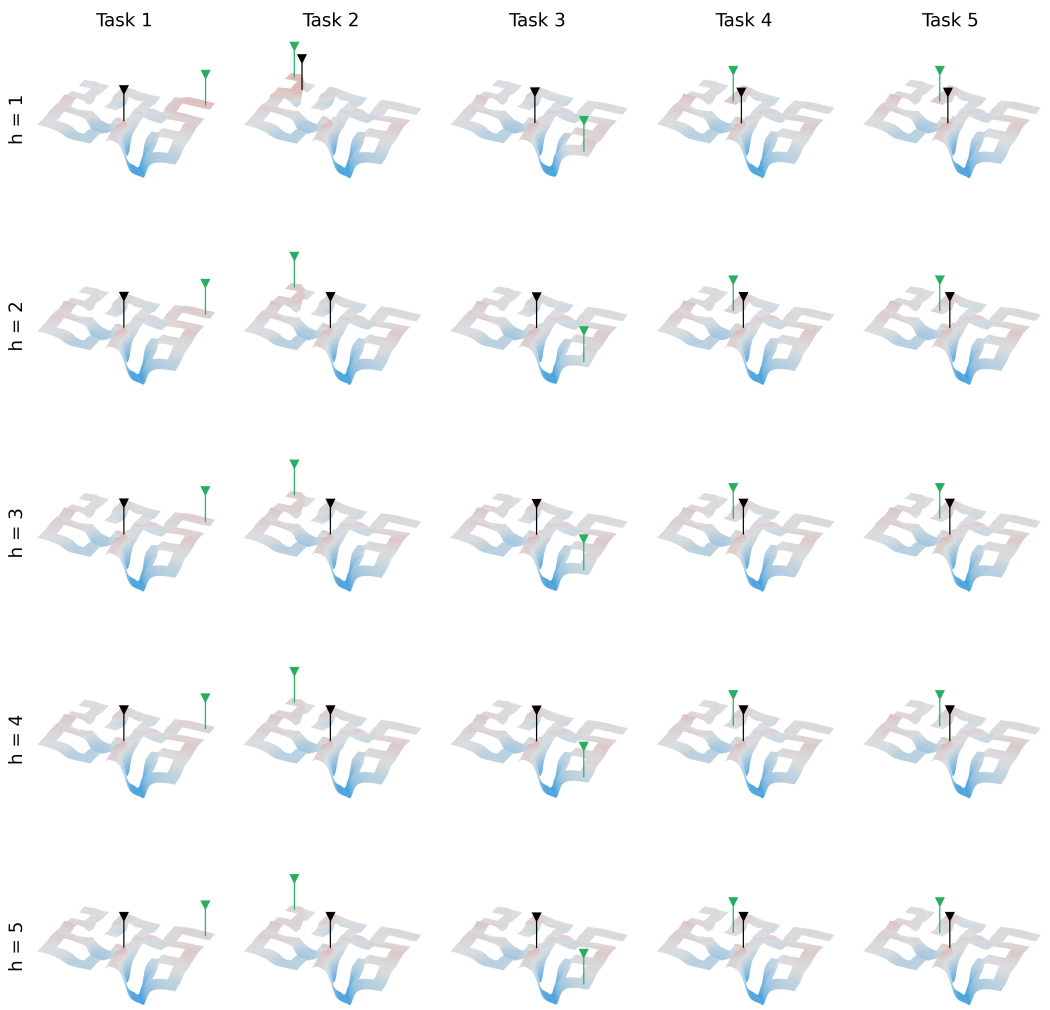


Figure 7: **The fitted reward function using the FB representation** for different h and different tasks in `antmaze-ln`. The green marker shows the ground-truth goal location, while the black marker shows the position with the maximum reconstructed reward.

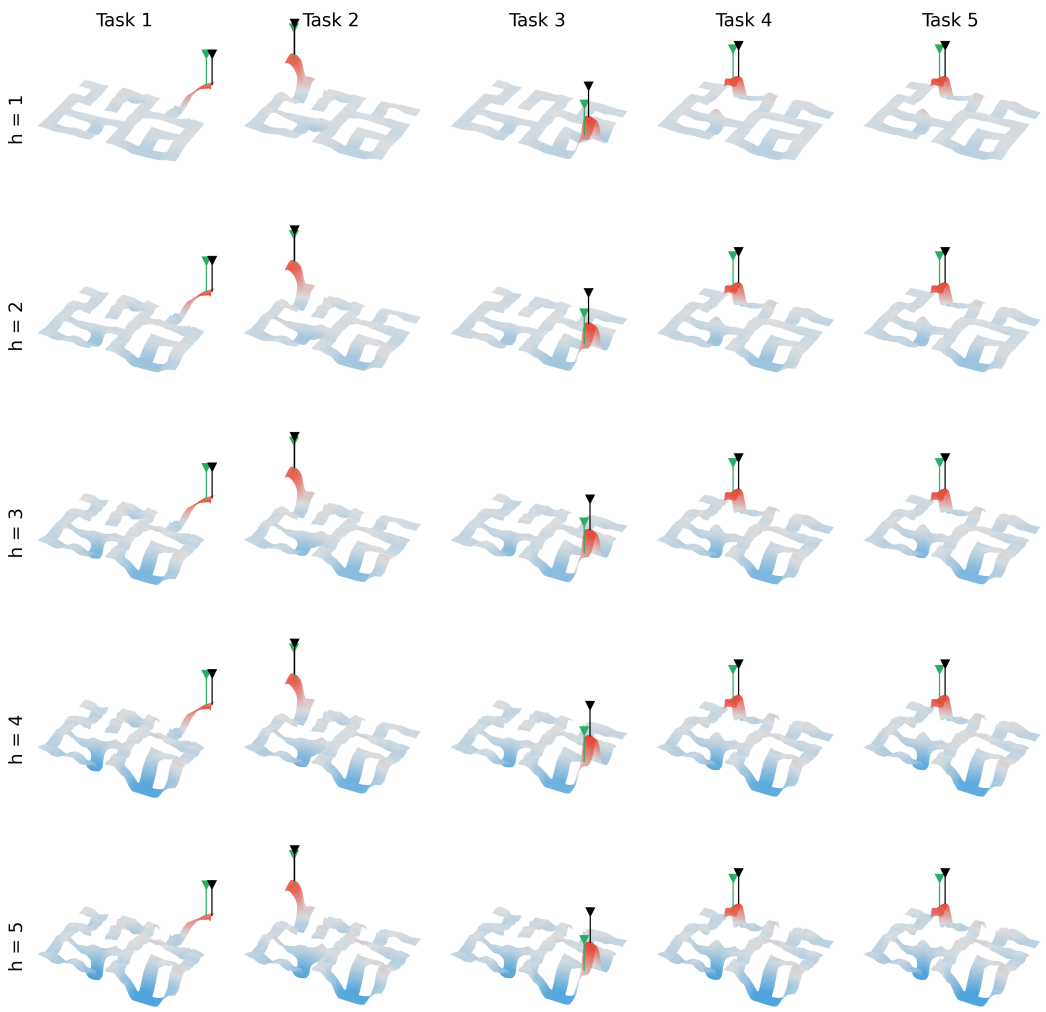


Figure 8: **The fitted reward function using the ReAL representation** for different h and different tasks in `antmaze-ln`. The green marker shows the ground-truth goal location, while the black marker shows the position with the maximum reconstructed reward.

B.3 ZERO-SHOT PERFORMANCE

We show the success rates for the Laplacian representation, the FB representation, and the ReAL representation in Tables 7, 8, and 9 respectively, and, similarly, the undiscounted returns for the three representations in Tables 10, 11, and 12 respectively.

Table 7: The mean success rate (%) and the standard error for the Laplacian representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
`antmaze-mn`	90.0 ± 8.0	52.4 ± 14.5	24.8 ± 13.0	17.2 ± 9.0	14.8 ± 5.8
`antmaze-ms`	98.0 ± 1.1	24.8 ± 1.7	16.0 ± 1.1	11.2 ± 1.5	12.4 ± 1.0
`antmaze-ln`	78.8 ± 1.5	4.0 ± 1.3	5.6 ± 2.5	2.8 ± 1.2	1.2 ± 0.8
`antmaze-ls`	24.8 ± 2.7	3.6 ± 1.8	2.0 ± 0.9	3.2 ± 1.6	3.6 ± 1.2

Table 8: The mean success rate (%) and the standard error for the FB representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
`antmaze-mn`	74.0 ± 2.6	88.0 ± 2.0	56.0 ± 7.5	35.6 ± 2.8	28.4 ± 2.9
`antmaze-ms`	65.2 ± 5.1	34.4 ± 3.7	20.4 ± 1.9	17.2 ± 1.9	15.2 ± 2.1
`antmaze-ln`	47.2 ± 6.3	10.4 ± 3.3	2.0 ± 0.6	2.0 ± 1.5	0.8 ± 0.5
`antmaze-ls`	13.6 ± 2.5	3.2 ± 3.2	2.4 ± 1.9	2.0 ± 1.3	2.4 ± 2.4

Table 9: The mean success rate (%) and the standard error for the ReAL representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
`antmaze-mn`	98.0 ± 0.9	97.6 ± 0.8	98.4 ± 1.2	97.2 ± 1.5	97.2 ± 0.8
`antmaze-ms`	96.8 ± 0.5	97.6 ± 0.8	75.2 ± 4.2	65.2 ± 0.8	46.0 ± 2.9
`antmaze-ln`	87.6 ± 5.0	76.8 ± 10.5	64.4 ± 15.9	49.6 ± 13.1	39.2 ± 11.9
`antmaze-ls`	34.4 ± 2.5	31.2 ± 2.6	24.4 ± 1.6	15.2 ± 2.1	10.0 ± 1.4

Table 10: The mean undiscounted return and the standard error for the Laplacian representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
`antmaze-mn`	-329.2 ± 92.1	-670.2 ± 97.4	-878.0 ± 70.7	-920.3 ± 44.1	-935.7 ± 30.2
`antmaze-ms`	-274.7 ± 18.2	-845.1 ± 13.2	-918.8 ± 14.0	-961.3 ± 22.9	-956.3 ± 13.3
`antmaze-ln`	-594.3 ± 12.6	-1000.5 ± 7.1	-1001.8 ± 4.3	-1007.9 ± 5.4	-1039.2 ± 13.3
`antmaze-ls`	-861.6 ± 10.4	-999.3 ± 7.9	-1016.0 ± 7.4	-1029.0 ± 15.7	-1025.6 ± 15.4

Table 11: The mean undiscounted return and the standard error for the FB representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
`antmaze-mn`	-452.1 ± 23.4	-390.7 ± 19.8	-647.8 ± 39.9	-807.6 ± 21.1	-851.7 ± 11.8
`antmaze-ms`	-596.5 ± 40.4	-793.7 ± 26.3	-881.6 ± 9.5	-923.8 ± 19.2	-946.1 ± 20.5
`antmaze-ln`	-792.7 ± 25.5	-967.1 ± 11.1	-994.5 ± 3.1	-992.9 ± 6.0	-1001.1 ± 3.5
`antmaze-ls`	-927.5 ± 14.2	-980.1 ± 20.1	-990.8 ± 7.8	-994.4 ± 5.7	-993.3 ± 7.7

Table 12: The mean undiscounted return and the standard error for the ReAL representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
`antmaze-mn`	-235.5 ± 6.7	-246.0 ± 9.5	-257.4 ± 13.5	-281.2 ± 12.3	-328.3 ± 19.1
`antmaze-ms`	-236.9 ± 3.7	-273.5 ± 18.3	-481.4 ± 23.6	-585.4 ± 17.3	-711.9 ± 17.9
`antmaze-ln`	-501.2 ± 28.9	-587.5 ± 62.6	-718.1 ± 108.2	-839.5 ± 99.8	-867.6 ± 69.1
`antmaze-ls`	-804.0 ± 15.7	-821.8 ± 1.5	-888.6 ± 15.3	-944.4 ± 9.2	-981.4 ± 4.8

B.4 INCREASING THE NUMBER OF DIMENSIONS

In this section, we study the effect of increasing the representation dimension for the Laplacian and FB representations. We show the success rates for the Laplacian and FB representation at 100 dimensions in Tables 13 and 14 respectively, and, similarly, the undiscounted returns in Tables 15 and 16 respectively. We can observe that while increasing the number of dimensions slightly improves zero-shot performance for $h > 0$, both reward-agnostic representations at 100 dimensions still significantly underperform the ReAL representation at 50 dimensions.

Table 13: The mean success rate (%) and the standard error for the 100-dimensional Laplacian representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
antmaze-mn	96.0 ± 0.9	71.6 ± 7.3	42.4 ± 9.7	30.0 ± 7.7	24.4 ± 6.7
antmaze-ms	78.8 ± 3.5	46.4 ± 2.5	22.4 ± 0.8	17.2 ± 1.5	14.0 ± 1.6
antmaze-ln	65.2 ± 6.5	14.4 ± 6.8	9.6 ± 6.9	10.4 ± 7.3	7.6 ± 5.8
antmaze-ls	36.0 ± 2.5	8.8 ± 2.2	3.2 ± 1.0	5.6 ± 1.3	7.2 ± 2.1

Table 14: The mean success rate (%) and the standard error for the 100-dimensional FB representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
antmaze-mn	76.8 ± 3.9	82.0 ± 6.2	68.4 ± 6.6	52.8 ± 7.1	44.4 ± 7.0
antmaze-ms	63.6 ± 8.1	35.6 ± 4.3	23.2 ± 2.7	17.2 ± 2.3	9.2 ± 1.6
antmaze-ln	51.2 ± 5.9	14.0 ± 4.0	3.6 ± 1.2	0.8 ± 0.5	0.8 ± 0.5
antmaze-ls	22.4 ± 2.0	17.2 ± 1.7	11.6 ± 2.3	8.4 ± 2.6	7.6 ± 2.6

Table 15: The mean undiscounted return and the standard error for the 100-dimensional Laplacian representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
antmaze-mn	-258.7 ± 9.8	-559.1 ± 64.1	-770.7 ± 72.7	-846.6 ± 57.5	-910.5 ± 45.2
antmaze-ms	-411.6 ± 37.7	-681.8 ± 10.6	-860.3 ± 6.6	-901.0 ± 13.6	-918.4 ± 15.6
antmaze-ln	-683.6 ± 37.7	-942.1 ± 26.5	-962.6 ± 38.0	-962.7 ± 37.6	-993.0 ± 20.3
antmaze-ls	-816.1 ± 6.7	-963.7 ± 12.5	-994.1 ± 4.0	-993.7 ± 11.8	-992.7 ± 8.1

Table 16: The mean undiscounted return and the standard error for the 100-dimensional FB representation for different `antmaze` settings, averaged over 5 runs.

h	0	1	2	3	4
antmaze-mn	-483.5 ± 36.0	-476.2 ± 41.6	-617.5 ± 40.6	-723.9 ± 39.3	-783.7 ± 31.8
antmaze-ms	-644.7 ± 53.3	-785.9 ± 29.2	-892.0 ± 18.1	-926.5 ± 17.0	-990.6 ± 28.0
antmaze-ln	-808.0 ± 32.0	-946.3 ± 23.5	-999.0 ± 6.0	-1000.8 ± 0.8	-1014.1 ± 7.7
antmaze-ls	-871.3 ± 9.4	-906.2 ± 11.4	-941.3 ± 15.6	-953.7 ± 15.8	-961.1 ± 15.9