Representations and Exploration for Deep Reinforcement Learning using Singular Value Decomposition

Yash Chandak [1]   Shantanu Thakoor [2]   Zhaohan Daniel Guo [2]   Yunhao Tang [2]
Remi Munos [2]   Will Dabney [2]   Diana Borsa [2]

Abstract

Representation learning and exploration are among the key challenges for any deep reinforcement learning agent. In this work, we provide a singular value decomposition-based method that can be used to obtain representations that preserve the underlying transition structure in the domain. Perhaps interestingly, we show that these representations also capture the relative frequency of state visitations, thereby providing an estimate for pseudo-counts for free. To scale this decomposition method to large-scale domains, we provide an algorithm that never requires building the transition matrix, can make use of deep networks, and also permits mini-batch training. Further, we draw inspiration from predictive state representations and extend our decomposition method to partially observable environments. With experiments on multi-task settings with partially observable domains, we show that the proposed method can not only learn useful representation on DM-Lab-30 environments (that have inputs involving language instructions, pixel images, and rewards, among others) but it can also be effective at hard exploration tasks in DM-Hard-8 environments.

1. Introduction

Developing reinforcement learning (RL) methods to tackle problems that have complex observations, partial observability, and sparse reward signals is an active research direction. Such problems provide interesting challenges because the observations can often contain multi-modal information consisting of visual features, text, and voice. Coupled with partial observability, an agent must look at a sequence of past observations and learn useful representations to solve

[1]University of Massachusetts [2]DeepMind. Correspondence to: Yash Chandak <ychandak@umass.edu>.

the RL task. Additionally, when the reward signal is sparse, exploring exhaustively becomes impractical and careful consideration is required to determine which parts of the environment should be explored. That is, efficient exploration depends on an agent's representation. Therefore, a unified framework to tackle both representation learning and exploration can be beneficial for tackling challenging problems.

Approaches for representation learning often include auxiliary tasks for predicting future events, either directly (Jaderberg et al., 2016) or in the latent space (Guo et al., 2020; 2022; Schwarzer et al., 2020). However, in the rich observation setting, fully reconstructing observations may not be practical, while reconstruction in the latent space is prone to representation collapse. Alternatively, auxiliary tasks using contrastive losses can prevent representation collapse (Chen et al., 2020; Srinivas et al., 2020), but can be quite sensitive to the choice of negative sampler (Zbontar et al., 2021).

Similarly, approaches targeting exploration often include reward bonuses based on counts (Bellemare et al., 2016), or reconstruction errors (Schmidhuber, 1991; Pathak et al., 2017). While these ideas are appealing, in rich-observation settings enumerating counts can be challenging and reconstruction can focus on irrelevant details, resulting in misleading bonuses. Methods like RND (Burda et al., 2018) partially address these concerns but are completely decoupled from any representation learning method. This substantially limits the opportunity for targeted exploration in the agent's model of the environment. A detailed overview of other related work is deferred to Section 6 and Appendix A.

In this work we provide De-Rex: <u>de</u> composition-based <u>r</u>epresentation and <u>ex</u>ploration, an approach that builds upon singular value decomposition (SVD) to train *both* a useful state representation and a pseudo-count estimate for exploration in rich-observation/high-dimensional settings.

Structure preserving representations: De-Rex uses a decomposition procedure to learn a state representation that preserves the transition structure without ever needing to build the transition matrix (Section 3). Further, our method requires only access to transition samples in a mini-batch form and is compatible with deep networks-based decom-

position (Section 4). This improves upon prior work which used decompositions but was restricted to tabular settings (Mahadevan, 2005; Mahadevan and Maggioni, 2007) or required symmetric matrices (e.g., Laplacian (Wu et al., 2018)) that could corrupt the underlying structure when the transition kernel is asymmetric.

Pseudo-counts for exploration: We show that the norms of these learned state representations capture relative state visitation frequency, thereby providing a very cheap procedure to obtain pseudo-counts (Sections 3 and 4). This significantly streamlines the exploration procedure in contrast to prior works that required learning exploratory options using spectral decomposition of the transition kernel (Machado et al., 2017a;b).

Performance on large-scale experiments: Building upon ideas from predictive state representations (Littman and Sutton, 2001; Singh et al., 2003), De-Rex also provides a procedure to tackle representation learning and exploration in partially observable environments (Section 5). We demonstrate the effectiveness of De-Rex on DM-Lab-30 (Beattie et al., 2016) and DM-Hard-8 exploration tasks (Paine et al., 2019), in a multi-task setting (*without* access to task label). These domains are procedurally generated, partially observable, and have multi-modal observations involving language instructions, pixel images, and rewards (Section 7).

2. Preliminaries

We will begin in a Markov decision process (MDP) setting and develop the key insights using singular value decomposition (SVD). Later in Section 5, we will extend the ideas to partially observable MDPs (POMDPs).

Any vector x is treated as a column vector. Any matrix $\mathbf{M}$ is denoted using bold capital letters, and $\mathbf{M}_{[i,j]}$ correspond to i^{th} row and j^{th} column of $\mathbf{M}$. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, p, r, d_0)$, where $\mathcal{S}$ is the set of finite states, $\mathcal{A}$ is a set of finite actions, r is the reward function, p is the transition function, and d_0 is the starting-state distribution. A policy π is a decision making rule, and let $\mathbf{P}_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ be the transition matrix induced by π such that $\mathbf{P}_{\pi[s,s']} := \sum_{a \in \mathcal{A}} \Pr(s'|s,a)\pi(a|s)$. Let the performance of a policy π be $J(\pi) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \right]$, where $\gamma \in [0,1)$ and R_t is the reward observed at time t on interacting with the environment with π. Let Π be a class of (parameterized) policies. The goal is to find $\arg\max_{\pi \in \Pi} J(\pi)$.

2.1. Why SVD?

Following the seminal work on proto-value functions (Mahadevan, 2005; Mahadevan and Maggioni, 2007), representations obtained from spectral decomposition have been shown to be effective not only towards minimizing the Bellman error (Behzadian et al., 2019) but also stabilizing off-
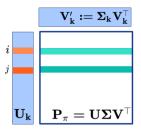

Figure 1: Consider top-k SVD decomposition of $\mathbf{P}_\pi \approx \mathbf{U_k}\mathbf{V'_k}$, where $\mathbf{V'_k} := \mathbf{\Sigma_k}\mathbf{V_k}^\top \in \mathbb{R}^{k \times |\mathcal{S}|}$. After decomposition, we consider the rows of $\mathbf{U_k} \in \mathbb{R}^{|\mathcal{S}| \times k}$ to be the $k \ll |\mathcal{S}|$ dimensional representation for the states. Intuitively, if two states have similar next state transition distributions (teal rows $\mathbf{P}_{\pi[i]}$ and $\mathbf{P}_{\pi[j]}$) then their representations (orange rows $\mathbf{U_{k[i]}}$ and $\mathbf{U_{k[j]}}$) would also be similar. Notice that rows in $\mathbf{U_k}$ can be thought of as coefficients for combining the common basis vectors $\mathbf{V'_k}$ to best estimate rows of $\mathbf{P}_\pi$ (i.e., $\mathbf{P}_{\pi[i]} \approx \mathbf{U_{k[i]}}\mathbf{V'_k}$, and $\mathbf{P}_{\pi[j]} \approx \mathbf{U_{k[j]}}\mathbf{V'_k}$). Therefore, if $\mathbf{P}_{\pi[i]} \approx \mathbf{P}_{\pi[j]}$, then $\mathbf{U_{k[i]}}\mathbf{V'_k} \approx \mathbf{U_{k[j]}}\mathbf{V'_k}$.

policy learning (Ghosh and Bellemare, 2020). Of particular interest are recent works by Lyle et al. (2021; 2022b) that show under idealized conditions that *state features* emerging from value function estimation, using just TD-learning, capture the top-k subspace of the eigenbasis of $\mathbf{P}_\pi$. As we show in the following, these top-k eigenvectors of $\mathbf{P}_\pi$ are *exactly* same as the top-k eigenvectors of $(\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}$. Therefore, these basis vectors can be particularly useful for obtaining state representations as the value function is linear in $(\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}$ (Sutton and Barto, 2018). That is, $v^\pi = (\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}R_\pi$, where $R_\pi \in \mathbb{R}^{|\mathcal{S}|}$ is a vector with one-step expected reward $\mathbf{E}_\pi[R|s]$ for all the states $s \in \mathcal{S}$.

Theorem 1. *If the eigenvalues of $\mathbf{P}_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ are real and distinct, [1] then for any k, the top-k eigenvectors of $\mathbf{P}_\pi$ and $(\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}$ are the same.*

Proofs for all the results are deferred to Appendix B. These lead to a natural question: *Can we explicitly decompose $\mathbf{P}_\pi$ to aid representation learning?* To answer this, Theorem 1 can be useful from a practical standpoint, as estimating and decomposing $\mathbf{P}_\pi$ can be much easier than doing so for $(\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}$. Unfortunately, however, eigen decomposition of $\mathbf{P}_\pi$ need not exist in general when $\mathbf{P}_\pi$ is not diagonalizable, for example, if $\mathbf{P}_\pi$ is not symmetric.

Therefore, we instead consider singular value decomposition (SVD), which generalizes eigen decomposition, of $\mathbf{P}_\pi$ and always exists (Van Loan, 1976). That is, there always exists orthogonal matrices $\mathbf{U} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, and a real-valued diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ such that $\mathbf{P}_\pi = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. In Figure 1, we provide some intuition

---

[1] we consider eigenvalues to be real and distinct to simplify the choice of 'top-k' eigenvalues.

for what the top-k SVD decomposition of $\mathbf{P}_\pi$ captures: $\mathbf{P}_\pi \approx \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^\top$, where $\mathbf{U_k} \in \mathbb{R}^{|\mathcal{S}|\times k}$ and $\mathbf{V_k} \in \mathbb{R}^{|\mathcal{S}|\times k}$ are the top-k left and right singular vectors corresponding to the top-k singular values $\mathbf{\Sigma_k} \in \mathbb{R}^{k\times k}$. We build upon this insight to develop an SVD-based scalable method that goes beyond just representation learning and can also be useful to obtain intrinsic bonuses to drive exploration.

Outside of RL, different matrix decomposition methods such as Kernel PCA (Schölkopf et al., 1997), deep decomposition (De Handschutter et al., 2021; Handschutter et al., 2021), and EigenGame (Gemp et al., 2020; Deng et al., 2022) have also been proposed. In what follows, we draw inspiration from these but focus on the challenges for the RL setup, i.e., asymmetric (transition) matrices, construction of exploration bonuses, and partial observability.

3. Tabular Setting

To convey the core insights and lay the foundations for our approach, we will first consider the tabular setting. In Section 4, we will extend these insights to develop scalable methods that use function approximations.

Decomposing $\mathbf{P}_\pi$ exactly would require transitions corresponding to all the states. This is clearly impractical, as an agent would rarely have access to all the possible states. Therefore, we consider decomposing a weighted version of $\mathbf{P}_\pi$. Let $\mathcal{D} := \{(s, a, r, s')_i\}_{i=1}^n$ be a dataset of n transition tuples collected using π. Let $d \in \Delta(\mathcal{S})$ be the distribution of the states s from which the observed transition tuples are sampled, and let $\mathbf{D} := \mathtt{diag}(d) \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$. In the following, we will consider decomposing the weighted matrix $\mathbf{DP}_\pi$ (when d is uniform, $\mathbf{DP}_\pi \propto \mathbf{P}_\pi$). As we will elaborate in the following sections, this choice of decomposing $\mathbf{DP}_\pi$ has two major advantages: **(a)** It will provide a practical representation learning procedure that only requires sampling observed transition tuples, and **(b)** it will also provide, almost automatically, estimates of pseudo-counts that will be useful for exploration.

Representation Learning: In this section, we outline the key idea for representation learning. Before proceeding, we introduce some additional notation. Let $x_s \in \mathbb{R}^{|\mathcal{S}|}$ be the one-hot encoding corresponding to state s, and let $\mathbf{X} \in \mathbb{R}^{n\times|\mathcal{S}|}$ be the stack of x_s for the states s in the dataset $\mathcal{D}$. Similarly, let $y_s \in \mathbb{R}^{|\mathcal{S}|}$ be the one-hot encoding corresponding to s' and let $\mathbf{Y} \in \mathbb{R}^{n\times|\mathcal{S}|}$ be the matrix consisting the stack of s' in the dataset $\mathcal{D}$.

Using the dataset $\mathcal{D}$ we provide a sample estimate for $\mathbf{DP}_\pi$ that we would like to decompose. While we will *never* actually construct this, a possibly intractable, estimate of $\mathbf{DP}_\pi$, it will be useful for deriving the proposed procedure.

Thus, let

$$\mathbf{A_n} := \frac{1}{n}\mathbf{X}^\top\mathbf{Y} = \frac{1}{n}\sum_{i=1}^n x_i y_i^\top. \tag{1}$$

Theorem 2. $\mathbf{A_n}$ *is an unbiased and a consistent estimator of* $\mathbf{DP}_\pi$, *i.e.,*

$$\forall i, j, \quad \mathbb{E}\left[\mathbf{A_{n[i,j]}}\right] = \mathbf{DP}_{\pi[i,j]}, \quad \mathbf{A_{n[i,j]}} \xrightarrow{a.s.} \mathbf{DP}_{\pi[i,j]}.$$

With a slight overload of notation, let $\mathbf{U} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ and $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ be the left and the right singular vectors for $\mathbf{A_n}$, and let $\mathbf{\Sigma} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ be the singular values such that

$$\mathbf{A_n} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top. \tag{2}$$

Now pre-multiplying both sides in (2) with $\mathbf{U}^\top$, post-multiplying with $\mathbf{V}$ and observing that $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$,

$$\mathbf{U}^\top\mathbf{A_n}\mathbf{V} = \mathbf{\Sigma}. \tag{3}$$

Expanding (3) using (1), we obtain

$$\frac{1}{n}\sum_{i=1}^n \mathbf{U}^\top x_i y_i^\top \mathbf{V} = \mathbf{\Sigma}. \tag{4}$$

Now, let $f_\mathbf{U} : \mathcal{S} \to \mathbb{R}^{|S|}$ and $g_\mathbf{V} : \mathcal{S} \to \mathbb{R}^{|S|}$ be linear functions parameterized by 'weights' $\mathbf{U}$ and $\mathbf{V}$ respectively. Then (4) can be expressed as,

$$\frac{1}{n}\sum_{i=1}^n f_\mathbf{U}(x_i)g_\mathbf{V}(y_i)^\top = \mathbf{\Sigma}. \tag{5}$$

The form in (5) is useful because it naturally leads to a loss function to search for 'parameters' $\mathbf{U}$ and $\mathbf{V}$ that provide the decomposition in (2). Specifically, if we want the top-k decomposition, let $\widehat{\mathbf{U}} \in \mathbb{R}^{|\mathcal{S}|\times k}$ and $\widehat{\mathbf{V}} \in \mathbb{R}^{|\mathcal{S}|\times k}$ be the estimates for top-k left and right singular vectors, and let $\mathbf{\Sigma}(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}) \in \mathbb{R}^{k\times k}$ be defined such that,

$$\mathbf{\Sigma}\left(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}\right) := \frac{1}{n}\sum_{i=1}^n f_{\widehat{\mathbf{U}}}(x_i)g_{\widehat{\mathbf{V}}}(y_i)^\top,$$

where $f_{\widehat{\mathbf{U}}} : \mathcal{S} \to \mathbb{R}^k$ and $g_{\widehat{\mathbf{V}}} : \mathcal{S} \to \mathbb{R}^k$. To perform the top-k decomposition, one could now search for $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ that makes $\mathbf{\Sigma}(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})$ resemble the properties of $\mathbf{\Sigma}$ i.e., $\mathbf{\Sigma}(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})$ is a diagonal matrix with large values on the diagonal. Specifically, we consider the following optimization problem where given $\widehat{\mathbf{U}}^\top\widehat{\mathbf{U}} = \widehat{\mathbf{V}}^\top\widehat{\mathbf{V}} = \mathbf{I}$ the diagonal elements of $\mathbf{\Sigma}(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})$ are maximized while the off-diagonal elements are 0,

$$\mathbf{U}^*, \mathbf{V}^* := \underset{\widehat{\mathbf{U}}, \widehat{\mathbf{V}}}{\arg\min} -\sum_{i=1}^k \mathbf{\Sigma}(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})_{[i,i]}$$

$$\text{s.t. } \mathbf{\Sigma}(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})_{[i,j]} = 0 \quad \forall i \neq j \tag{6}$$
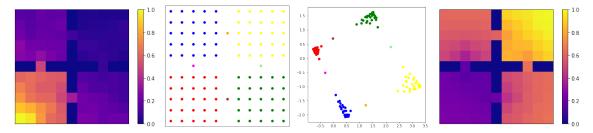
Figure 2: **(A)** State visitation frequency in the data collected using a random policy. **(B)** Reference colors for the states in the 4rooms. **(C)** 2D Representations learned from 121-dimension tabular representation. **(D)** Bonus constructed using the learned representations. Legend: values are normalized between 0 and 1 and darker color represents lower value.

Resulting function $f_{\mathbf{U}^*}$ provides the representation for the states. For instance, similar to the example in Figure 1, $f_{\mathbf{U}^*}(x_s) = \mathbf{U}^{*\top} x_s$, which is the row of $\mathbf{U}^* \in \mathbb{R}^{|S| \times k}$ corresponding to the state s because x_s is one-hot. The function $g_{\mathbf{V}^*}$ is only used in searching for the function $f_{\mathbf{U}^*}$. In Figure 2(c), we provide an illustrative example on the 4-rooms domains, where it can be observed that the learned representation clusters the states that have similar transition distributions (even when $\mathbf{D}$ is far from uniform).

Exploration: We now discuss how representations obtained from the SVD decomposition of $\mathbf{DP}_\pi$ can also be beneficial for exploration. In the following, we show that a pseudo-count estimate of a state's visitation frequency, relative to other states, can be obtained instantly by just computing the norms of the learned state representation.

Let the decomposition of $\mathbf{DP}_\pi = \mathbf{U\Sigma V}^\top$ and as earlier let the representation function $f_{\mathbf{U}}$ be parameterized linearly with left singular values $\mathbf{U}$. Let $\mathbf{\Lambda} := \mathbf{\Sigma}^2$, then the weighted norm of the representations $\|f_{\mathbf{U}}(x_s)\|_{\mathbf{\Lambda}^{-1}}$ is inversely proportional to $d(s)$, i.e., *how often s has been observed in the dataset relative to other states*.

Theorem 3. *If $\mathbf{P}_\pi \mathbf{P}_\pi^\top$ and $\mathbf{D}$ are invertible, then for $\alpha_s := (\mathbf{P}_\pi \mathbf{P}_\pi^\top)_{[s,s]}^{-1}$,*

$$\left\| f_{\mathbf{U}}(x_s) \right\|_{\mathbf{\Lambda}^{-1}}^2 = \frac{\alpha_s}{d(s)^2}. \quad (7)$$

Here α_s is a state-dependent constant that is independent of $d(s)$. Theorem 3 is particularly appealing because not only can f_U provide representations that respect the similarity in terms of next-state transitions, but it also orients the representations such that norms of the representations can be used to indicate pseudo-counts. Further estimating the bonus only requires computing $\left\| f_{\mathbf{U}}(x_s) \right\|_{\mathbf{\Lambda}^{-1}}^2$, where $\mathbf{\Lambda}$ is a diagonal matrix which can be inverted in $O(k)$ time, where k is the dimension of the representation. Therefore computing the norm is also an $O(k)$ time operation. Figure 2(D), provides an illustrative example of what $\|f_{\mathbf{U}}(x_s)\|_{\mathbf{\Lambda}^{-1}}$ captures in the 4-rooms domain.

Remark 1. *Intuitively, there are extra degrees of freedom in how representations can be rotated/re-oriented such that their relative distances from each other are the same, but their norm changes. This freedom is leveraged in (7) to re-orient the representations such that states that have higher visitation tend to get representations that have a lower norm value. This can be visualized in Fig 2(C) and 3(C).*

Remark 2. *Alternatively, observe that $\mathbf{DP}_\pi$ is related to the (normalized) count matrix (Bellemare et al., 2016). Therefore, decomposing it results in representations that have properties of both the visitation frequency $\mathbf{D}$ and the transition structure $\mathbf{P}_\pi$.*

4. Function Approximations

While (6) provides the problem *formulation*, it does not provide a scalable way to *perform* the optimization. Further, we would want to extend the procedure to work with deep neural networks. In the following, we elaborate on a way to achieve that.

Representation Learning Leveraging the form in (5), we extend the idea using non-linear functions $f_{\theta_1} : \mathbb{R}^d \to \mathbb{R}^k$ and $g_{\theta_2} : \mathbb{R}^d \to \mathbb{R}^k$ that are parameterized by weights $\theta := [\theta_1, \theta_2]$ and take states (need not be tabular) as inputs,

$$\mathbf{\Sigma}(\theta) := \frac{1}{n} \sum_{i=1}^n f_{\theta_1}(s_i) g_{\theta_2}(s_i')^\top \in \mathbb{R}^{k \times k}. \quad (8)$$

Specifically, let

$$\mathcal{L}_{\text{diag}}(\mathbf{\Sigma}(\theta)) := \frac{1}{k} \sum_{i=1}^k \mathbf{\Sigma}(\theta)_{[i,i]}, \quad (9)$$

$$\mathcal{L}_{\text{off}}(\mathbf{\Sigma}(\theta)) := \frac{1}{k^2 - k} \sum_{\substack{i,j=1 \\ i \neq j}}^k \mathbf{\Sigma}(\theta)_{[i,j]}^2. \quad (10)$$

Now similar to (6) we define a loss with soft constraints by combining (9) and (10),

$$\mathcal{L}(\mathbf{\Sigma}(\theta)) := -\mathcal{L}_{\text{diag}}(\mathbf{\Sigma}(\theta)) + \lambda_r \mathcal{L}_{\text{off}}(\mathbf{\Sigma}(\theta)), \quad (11)$$
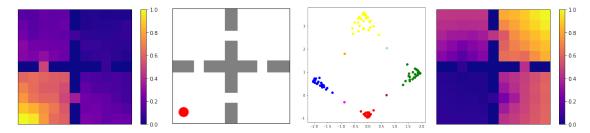
Figure 3: **(A)** State visitation frequency in the data collected using a random policy. **(B)** Example of the domain image used as the state. **(C)** 2D representations learned from images (30x30x3) corresponding to the states. Reference colors for the states are the same as that used in Figure 2 (B). **(D)** Bonus constructed using the learned representations. Legend: values are normalized between 0 and 1 and a darker color represents a lower value.

where λ_r is a hyper-parameter. For the tabular setting in (6), the condition for $\widehat{\mathbf{U}}^\top\widehat{\mathbf{U}} = \widehat{\mathbf{V}}^\top\widehat{\mathbf{V}} = \mathbf{I}$ prevents unbounded maximization of the diagonal by ensuring that the columns of $\mathbf{U}$ (i.e., each feature dimension), and columns of $\mathbf{V}$ have a norm of one. In the function approximation setting it is not immediate what this condition should be. Therefore, to prevent unbounded maximization of $\mathbf{\Sigma}(\theta)_{[i,i]}$ in (9), we use batchnorm (Ioffe and Szegedy, 2015) on the output of both f and g to ensure that each feature dimension remains normalized. In Figure 3(C) we illustrate the representations learned by minimizing the loss in (11) with convolutional neural networks, batch-normalization, and $\lambda_r = 1$.

Mini-batch optimization: In practice, computing $\mathbf{\Sigma}(\theta)$ in (8) using all n samples is impractical. Instead, it would be preferable to have a mini-batch estimate of $\mathbf{\Sigma}(\theta)$,

$$\widehat{\mathbf{\Sigma}}(\theta) \coloneqq \frac{1}{b}\sum_{i=1}^{b} f_{\theta_1}(s_i)g_{\theta_2}(s_i')^\top, \qquad (12)$$

where $b \ll n$ is the mini-batch size. Unfortunately, it can be shown that while the gradients for $\mathcal{L}_{\mathrm{diag}}(\widehat{\mathbf{\Sigma}}(\theta))$ are unbiased, gradients for $\mathcal{L}_{\mathrm{off}}(\widehat{\mathbf{\Sigma}}(\theta))$ will be biased in general.

Theorem 4. *Gradient of $\mathcal{L}_{\mathrm{diag}}(\widehat{\mathbf{\Sigma}}(\theta))$ is an unbiased estimator of $\mathcal{L}_{\mathrm{diag}}(\mathbf{\Sigma}(\theta))$, and gradient of $\mathcal{L}_{\mathrm{off}}(\widehat{\mathbf{\Sigma}}(\theta))$ is in general a biased estimate of $\mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta))$, i.e.,*

$$\mathbb{E}\left[\partial_\theta \mathcal{L}_{\mathrm{diag}}\left(\widehat{\mathbf{\Sigma}}(\theta)\right)\right] = \partial_\theta \mathcal{L}_{\mathrm{diag}}(\mathbf{\Sigma}(\theta)),$$
$$\mathbb{E}\left[\partial_\theta \mathcal{L}_{\mathrm{off}}\left(\widehat{\mathbf{\Sigma}}(\theta)\right)\right] \neq \partial_\theta \mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta)),$$

where the expectation is over the randomness of the mini-batch.

This problem can be attributed to the fact that in (10),

$$\mathbf{\Sigma}(\theta)^2_{[i,j]} = \left(\frac{1}{n}\sum_{k=1}^{n} f_\theta(s_k)\, g_\theta(s_k')^\top_{[i,j]}\right)^2,$$

the square is *outside* the expectation. This is reminiscent of the issue in residual gradient optimization of the mean squared Bellman error (Baird, 1995). Therefore, to address this issue, we also use the double-sampling trick to obtain unbiased gradient estimates. Specifically, let

$$\widehat{\widehat{\mathbf{\Sigma}}}(\theta) \coloneqq \left(\frac{1}{b_2}\sum_{i=1}^{b_2} f_{\theta_1}(s_i)g_{\theta_2}(s_i')^\top\right), \qquad (13)$$

be estimated using b_2 samples of a separate batch of data than the one used in (12). Then using (12) and (13), we use $\widehat{\mathcal{L}}_{\mathrm{off}}(\theta)$ instead of $\mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta))$ in (10), where

$$\widehat{\mathcal{L}}_{\mathrm{off}}(\theta) \coloneqq \frac{1}{(k^2-k)}\sum_{\substack{i,j=1 \\ i\neq j}}^{k} \widehat{\mathbf{\Sigma}}(\theta)_{[i,j]}\ \mathrm{sg}\left(\widehat{\widehat{\mathbf{\Sigma}}}(\theta)_{[i,j]}\right)$$
$$+ \frac{1}{(k^2-k)}\sum_{\substack{i,j=1 \\ i\neq j}}^{k} \widehat{\widehat{\mathbf{\Sigma}}}(\theta)_{[i,j]}\ \mathrm{sg}\left(\widehat{\mathbf{\Sigma}}(\theta)_{[i,j]}\right) \quad (14)$$

where sg corresponds to the stop-gradient operator.

Theorem 5. *Gradient $\partial_\theta \widehat{\mathcal{L}}_{\mathrm{off}}(\theta)$ is an unbiased estimator of $\partial_\theta \mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta))$, i.e.,*

$$\mathbb{E}\left[\partial_\theta \widehat{\mathcal{L}}_{\mathrm{off}}(\theta)\right] = \partial_\theta \mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta)).$$

Exploration: Having developed a mini-batch optimization procedure to obtain the representations, we now extend (7) to the function approximation setting. Specifically, using $\mathbf{\Lambda}(\theta) \coloneqq \mathbf{\Sigma}^2(\theta)$, we construct the estimate for pseudo-count using $\|f_{\theta_1}(s)\|_{\mathbf{\Lambda}(\theta)^{-1}}$.

However, as discussed earlier, obtaining $\mathbf{\Sigma}(\theta)$ exactly can be hard in practice. Therefore, in practice **(a)** we keep a running mean estimate $\widetilde{\mathbf{\Sigma}}(\theta)$ for different $\widehat{\mathbf{\Sigma}}(\theta)$ observed across mini-batches. Further, **(b)** as the loss in (11) encourages $\widetilde{\mathbf{\Sigma}}(\theta)$ to be diagonal, we only maintain a running estimate $\widetilde{\mathbf{\Sigma}}(\theta)$ for the diagonal elements and keep the off-diagonal elements as zero. Now using $\widetilde{\mathbf{\Lambda}}(\theta) \coloneqq \widetilde{\mathbf{\Sigma}}^2(\theta)$, we construct

the estimate for pseudo-counts of s using $\|f_{\theta_1}(s)\|_{\widetilde{\mathbf{\Lambda}}(\theta)^{-1}}$. In Figure 3 (D) we illustrate the effectivity of estimating pseudo-counts using this procedure.

5. POMDP & Predictive State Representation

An important advantage of the proposed approach is that we can consider other matrices (e.g., $\mathbf{P} \in \mathbb{R}^{|S||A| \times |S|}$ instead of $\mathbf{P}_\pi \in \mathbb{R}^{|S| \times |S|}$) as well. In this section, we take a step further and consider the *system-dynamics matrix* that are well-suited for *partially observable* settings.

Brief Review: For a partially observable system, let $\mathcal{O}$ be the set of possible observations and let $O_t \in \mathcal{O}$ be a random variable for the observations at time t. Let $\mathcal{H}$ be a set of observation-action sequences, such that $\mathcal{H} = \bigcup_{t \in \{0,1,...,T\}} \mathcal{H}_t$, where T is the horizon length, $\mathcal{H}_0 = \mathcal{O}$, and $\forall t > 0, \mathcal{H}_t = \mathcal{H}_{t-1} \times \mathcal{A} \times \mathcal{O}$. Let a *test* $\tau \in \mathcal{H}$ be any sequence of action and observations $(a_1, o_1, ..., a_j, o_j)$. Given a history $h = (o_0, a_0, ..., a_{i-1}, o_i) \in \mathcal{H}$, let $p(\tau|h) := \Pr(O_{i+1} = o_1, ..., O_{i+j} = o_j | h, A_i = a_1, ..., A_{i+j-1} = a_j)$ be the history-conditioned prediction of a test τ. Intuitively, $p(\tau|h)$ encodes the probability of future outcomes given the observed history. A system-dynamics matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{H}| \times |\mathcal{H}|}$ is defined to have its rows correspond to histories and columns correspond to tests, such that $\mathbf{W}_{[i,j]} := p(\tau_j|h_i)$. Figure 4 provides an illustration of a low-rank system-dynamics matrix $\mathbf{W} = \mathbf{U}\mathbf{V}'$, where $\mathbf{U} \in \mathbb{R}^{|\mathcal{H}| \times k}$, $\mathbf{V}' \in \mathbb{R}^{k \times |\mathcal{H}|}$. Further, let $(z_i)_{i=1}^k$ correspond to the latent variables of the system.
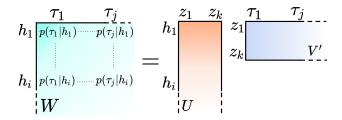

Figure 4: A low-rank system-dynamics matrix (adapted from the work by Singh et al. (2012)). Intuitively, row $\mathbf{U}_{[\mathbf{i}]}$ provides a good representation for history h_i because it contains *all the information that is sufficient to infer the outcome probability of all the futures (tests), given h_i*.

Viewing z_i's from different perspectives provides different insights. If a POMDP has k unobserved states then z can correspond to the hidden states of the system. Correspondingly, the rows $\mathbf{U}_{[i]}$ would be the belief distribution over those k-hidden states, given the history h_i so far, and columns of $\mathbf{V}'$ contain the probability of observing the futures/tests if the agent is in a given hidden state. In contrast, PSRs consider z to be *core tests* (Littman and Sutton, 2001; Singh et al., 2003; 2012; James and Singh, 2004; Rudary and Singh,

2003), such that the probability of all the remaining tests can be obtained as linear combinations of the probabilities of these core tests occurring. Therefore, $\mathbf{U}_{[i]}$ encodes the probability of these k core tests occurring given the history h_i, and the columns of $\mathbf{V}'$ contain the coefficients for the *linear combination*. Mathematically, this subtle difference from the standard POMDP perspective allows $\mathbf{V}'$ to have values that are not constrained to be probability distribution nor be positive, and thus allows PSR to model a much larger class of dynamical systems (Singh et al., 2012). Transformed PSR (TPSR) extends this perspective further and lets $\mathbf{U}_{[i]}$ correspond to be a small number of *linear combinations* of the probabilities of a larger number of tests. Mathematically, the key difference is that this permits $\mathbf{U}_{[i]}$ to not be probability distributions and have negative values as well. This perspective is particularly useful as the left singular vectors from the SVD on $\mathbf{W}$ now directly provide sufficient information from histories needed to infer future outcomes (Rosencrantz et al., 2004).

5.1. De-Rex: Decomposition based Representation and Exploration

We build upon the idea of TPSR that was often restricted to small-scale setups (Rosencrantz et al., 2004; Boots et al., 2011; Boots and Gordon, 2011), and extend them to a large-scale setting with function approximators using ideas developed in Section 4. That is, in (8) we use $f_{\theta_1} : \mathcal{H} \to \mathbb{R}^k$ and $g_{\theta_2} : \mathcal{H} \to \mathbb{R}^k$ which take *histories* and *tests* as inputs, instead of states and next states, respectively.

Specifically, let dataset $\mathcal{D}$ consist of n trajectories. For the ith trajectory $(o_0, a_0, ...a_{T-1}, o_T)$ in $\mathcal{D}$, we define $h_{ij} := (o_0, a_0, ..., o_j)$ and $\tau_{ij} := (a_j, o_{j+1}, ..., a_{T-1}, o_T)$. We extend (8), such that

$$\mathbf{\Sigma}(\theta) := \frac{1}{nT} \sum_{i=1}^{n} \sum_{j=0}^{T-1} f_{\theta_1}(h_{ij}) g_{\theta_2}(\tau_{ij})^\top \in \mathbb{R}^{k \times k}. \quad (15)$$

Now, using the double sampling trick from (14) and using a loss similar to (11), we can diagonalize $\mathbf{\Sigma}(\theta)$ in (15). We provide more discussion on these in Appendix C.

4 rooms (POMDP): Similar to Section 3 and 4, we also visualize the learned *representations for the histories* and the estimated *pseudo-counts for histories*. For this purpose, we construct a domain where the observation O_t at time t corresponds to an image of size $(30, 30, 3)$ that has the top-down view of the domain (similar to Figure 3 (B)), albeit that with probability p the agent's marker (red dot) is completely hidden. Therefore, when the marker is hidden, the observation O_t does not provide any information about the location of the agent, and the location needs to be inferred using the history. More details about this POMDP domain can be found in Appendix D, where we also provide an abla-
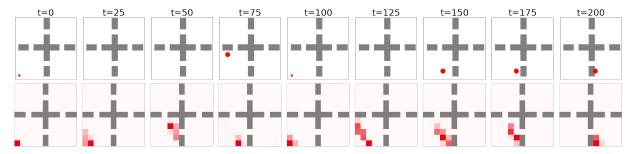
Figure 5: Similar to Figure 2 and 3, we provide plots in Figure 9 (in Appendix D) to assess the quality of the learned distributions. However, it is not particularly informative for representations in higher dimensions. Therefore, as an alternative, we learned a classification model (using a single-layer neural network) that aims to classify the true underlying state at the end of the observation sequence of a given (partial) history using the representation of that (partial) history. This provides us with a proxy for how well can the learned representations be used as to decode beliefs over the underlying true state. Top-row illustrates the snapshots of the observations that were provided to the agent at timesteps $t = \{0, 25, 50, 75, 100, 125, 150, 175, 200\}$ (notice the agent missing in some frames). The bottom rows denote the belief decoded from the representation of the history till timestep t. See text for more details. **Legend:** For the bottom row, the deeper red color indicates a higher degree of belief. For the top row, the red color indicates the location of the agent.

tion study across different values of $p = \{0.0, 0.3, 0.6, 0.9\}$. Here, we present the results for $p = 0.3$.

A sufficient statistic for any history is the ground truth state at the end of that history. It can be observed in Figures 9 and 10 (in Appendix D) that for the histories that are ending at a similar ground truth state (unknown to the agent), the proposed De-Rex method is able to provide similar representations for those histories. Additionally, the proposed method is also able to provide pseudo-counts for the underlying true state even when the true state is unknown.

In Figure 5, we provide an alternative way of assessing the quality of the learned representation for the histories. It can be observed that De-Rex provides representations that can be effective at decoding the true underlying states.

6. Related Work

Recent work by Tang et al. (2022) illustrates, under idealized assumptions, how self-predictive representation (Guo et al., 2020; 2022; Schwarzer et al., 2020) are related to spectral decomposition. Alternatively, Ren et al. (2022) consider explicitly decomposing the transition kernel, however, their method makes use of an estimated model, and their loss function requires access to the state distribution, which is typically not available directly. Further, Touati and Ollivier (2021); Lan et al. (2022) illustrate how the decomposition of long-term occupancy matrix could be useful for representation learning but they leave the exploration question open. Some of these methods use a contrastive learning loss, which might be complimentary to the proposed non-contrastive loss (Garrido et al., 2022). Spectral properties of the transition matrix have also been used to

learn options that aid exploration (Machado et al., 2017a;b; Liu et al., 2017). Further, norms of a related quantity, known as successor representations, have been shown to be useful pseudo-counts (Machado et al., 2020).

We discuss other related works in detail in Appendix A. Perhaps the work most relevant to ours is by Wu et al. (2018) which discusses how to decompose the Laplacian of $\mathbf{P}_\pi$ to obtain the associated eigenfunctions. Similar to ours, their method can also make use of deep networks and only requires access to transition samples. However, their method is based on using a *symmetric* Laplacian matrix, which may not be ideal depending on the underlying structure in the domain (for e.g., when $\mathbf{P}_\pi$ is not symmetric) being captured by the representations. Further, their method does not directly lends itself to estimating pseudo-counts for exploration either. In Section 7 we provide an empirical comparison against this method.

7. Empirical Results

In this section, we aim to empirically investigate the properties of the proposed De-Rex algorithm for large-scale domains. Specifically, we aim to answer the two main questions pertaining to representation learning and exploration.

Q1. Can De-Rex learn useful representations from rich observations?

To investigate this question, we use the DMLab 30 (Beattie et al., 2016) environment (Beattie et al., 2016). DMLab is particularly useful here as it is a procedurally-generated, partially observable 3-D world where properties such as object shapes, colors, and positions can vary every episode. The environments consist of 30 diverse navigation and puzzle-

solving tasks, where the state observations range from language instruction to pixel images from a first-person view, and rewards. The action set consists of a mix of discrete and continuous actions corresponding to motions for moving the agent's location, adjusting the visual angles, and controls for using various tools available in the environment.

To make it more challenging, we consider the multi-task setting where there is a single agent that interacts with all the tasks at once and must learn a common representation across all of these tasks. In our setup, there is *no explicit task identification* that is given to the agent, and so the agent must be able to deduce the task from its environment. We consider the following algorithms for comparison. Detailed discussions regarding implementation and hyperparameters for these are available in Appendix C.

RL Baseline: As our base RL algorithm, we use V-MPO (Song et al., 2019), a policy optimization method that had achieved state-of-the-art performance across several RL benchmarks involving both discrete and continuous tasks. To avoid confounding factors, this baseline agent does not have any explicit representation learning loss.

Laplacian decomposition: We compare against the work by Wu et al. (2018) that uses spectral graph drawing to learn state representations by decomposing the Laplacian associated with the transition kernel. As a baseline decomposition-based method, we use their proposed loss as an auxiliary task with the VMPO agent.

De-Rex Representation: Similar to the baseline decomposition-based method, we use the proposed auxiliary representation learning loss based on (11) and (15) alongside the VMPO agent. For these experiments, we focus only on understanding the usefulness of the representations being learned, without any additional exploration bonuses.

An empirical comparison of the algorithms for each task is presented in Figure 6. In Appendix E we also provide aggregate and separate learning curves for each task. It can be observed that the baseline VMPO that does not explicitly employ any representation learning losses, struggles to perform in the large-scale, rich observation setting. In comparison, the baseline method that uses decomposition of the Laplacian to learn state representations does slightly improve the performance. However, the proposed De-Rex representation learning method that is designed to explicitly handle partial observability provides the most improvement. We found that adding the De-Rex bonus for exploration did *not* improve results for these tasks, as in the dense reward settings bonuses can distract more than be helpful.

Q2. Can De-Rex explore complex environments with sparse rewards?

Solving large-scale domains require both: learning represen-
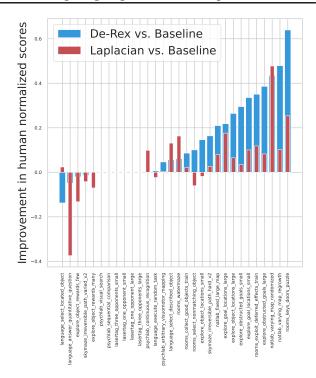

Figure 6: Per-game comparison of (a) De-Rex representation vs. VMPO baseline and (b) Laplacian representation vs. VMPO baseline. We compare the final human normalized scores (i.e., average of the last 2% of the total $2 \cdot 10^{10}$ training steps). Each score is averaged over three seeds.

tations and efficient exploration. To investigate this ability, we use the DMLab Hard-8 Suite (Gulcehre et al., 2019) that consists of 8 hard exploration tasks in the DMLab environment (Beattie et al., 2016) with sparse rewards, which require the agent to solve a sequence of non-trivial sub-tasks in the correct order to obtain the final reward. Similar to DMLab-30, we make the setting more challenging by considering the multi-task setup. For all the agents, we modify the reward at timestep t by adding an algorithm-dependent bonus to encourage exploration, $R'_t := R_t + \lambda_b \text{ Bonus}_t$, where λ_b is a hyper-parameter. Besides comparing against the methods introduced before (that do not employ any exploration strategy and their $\text{Bonus}_t = 0$ everywhere), we also compare the following methods.

Random Network Distillation: RND (Burda et al., 2018) has emerged as a strong baseline exploration method that consists of training an observation encoder to predict a separate, fixed, randomly initialized encoder of the same observation. The prediction loss is then used to compute Bonus_t for exploration.

De-Rex Exploration: An important aspect of the proposed work is that the estimates for the pseudo-count of state visitations can be obtained easily. Besides using representation
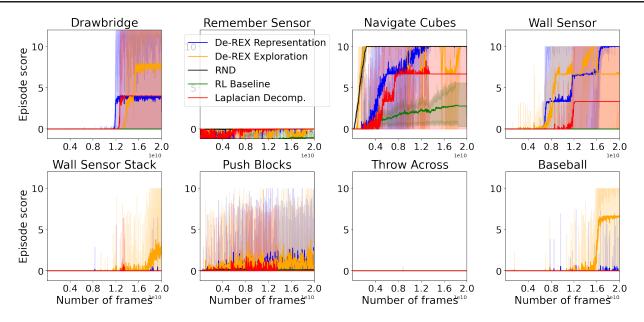
Figure 7: Learning curves for each task on the DM-Hard-8 suite (Gulcehre et al., 2019) in the multi-task setup. The shaded area corresponds to the minimum and maximum across three seeds. High variance in these environments is due to *extremely sparse rewards* (e.g., only at the end of completing a sequence of complex tasks the agent is able to get any reward). Because of this, if the agent is able to explore well enough it solves the task, or else it fails completely.

loss, we set $\texttt{Bonus}_t$ using the norm of the state/history representation at timestep t.

An empirical comparison of these methods is presented in Figure 7. It can be observed that the base VMPO method fails to solve any task as it lacks any explicit way to encourage exploration. Similarly, RND and the Laplacian decomposition (Burda et al., 2018) methods only manage to solve *one* DMHard-8 task. This can be attributed to the fact that the exploration bonus in RND does not include any representation learning component which is essential in the rich-observation setting. Whereas, the Laplacian decomposition method includes representation learning loss but has no explicit procedure to encourage exploration.

De-Rex provides a single, unified, procedure for both representation learning and estimating bonuses to encourage exploration. When only representation loss is used, *De-Rex Representation* is still able to solve nearly *two* tasks. When both the representation loss and exploration bonuses are used, *De-Rex Exploration* is able to solve nearly *four* tasks. Appendix E contains additional ablation studies on the DM-Hard-8 tasks using different hyperparameters combinations.

8. Discussion and Conclusion

We proposed an SVD based representation and exploration procedure. Our approach has three key advantages. (a) Simplicity: The proposed representation learning loss is

completely model-free and the exploration bonuses can be constructed readily by computing the norms of the representations. (b) Theory and empirical results: the proposed approach is grounded in the fundamental framework of SVD, and shows how SVD can be used not only for representation learning but also for exploration in challenging environments. (c) Extensibility: Our method can not only be used with other matrices of interest but also the proposed procedure can be used as an auxiliary loss with any future baseline (deep) RL algorithm.

Limitations: While we took the initial steps to lay out the analysis in the tabular setting, we believe an important future direction is to further our understanding by theoretically analyzing the function approximation setting as well. Further, investigating how to *explicitly* account for rewards during representation learning could be fruitful (using the proposed method as an auxiliary loss induces *implicit* dependency on rewards as the losses from policy/value functions also shape the representation).

9. Acknowledgement

References

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

A. Agarwal, S. Kakade, A. Krishnamurthy, and W. Sun. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020.

C. Allen, N. Parikh, O. Gottesman, and G. Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8229–8241, 2021.

M. G. Azar, B. Piot, B. A. Pires, J.-B. Grill, F. Altché, and R. Munos. World discovery models. *arXiv preprint arXiv:1902.07685*, 2019.

K. Azizzadenesheli, A. Lazaric, and A. Anandkumar. Reinforcement learning of pomdps using spectral methods. In *Conference on Learning Theory*, pages 193–256. PMLR, 2016.

A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

B. Behzadian, S. Gharatappeh, and M. Petrik. Fast feature selection for linear value function approximation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 601–609, 2019.

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

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

B. Boots and G. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 293–300, 2011.

B. Boots, S. M. Siddiqi, and G. J. Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, 2011.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

W. Dabney, A. Barreto, M. Rowland, R. Dadashi, J. Quan, M. G. Bellemare, and D. Silver. The value-improvement path: Towards better representations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7160–7168, 2021.

P. De Handschutter, N. Gillis, and X. Siebert. A survey on deep matrix factorizations. *Computer Science Review*, 42: 100423, 2021.

Z. Deng, J. Shi, and J. Zhu. Neuralef: Deconstructing kernels by deep neural networks. *arXiv preprint arXiv:2205.00165*, 2022.

A. Erraqabi, M. C. Machado, M. Zhao, S. Sukhbaatar, A. Lazaric, L. Denoyer, and Y. Bengio. Temporal abstractions-augmented temporally contrastive learning: An alternative to the laplacian in rl. *arXiv preprint arXiv:2203.11369*, 2022.

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

Q. Garrido, Y. Chen, A. Bardes, L. Najman, and Y. Lecun. On the duality between contrastive and non-contrastive self-supervised learning. *arXiv preprint arXiv:2206.02574*, 2022.

I. Gemp, B. McWilliams, C. Vernade, and T. Graepel. Eigengame: Pca as a nash equilibrium. *arXiv preprint arXiv:2010.00554*, 2020.

D. Ghosh and M. G. Bellemare. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning*, pages 3556–3565. PMLR, 2020.

K. Gregor, D. Jimenez Rezende, F. Besse, Y. Wu, H. Merzic, and A. van den Oord. Shaping belief states with generative environment models for rl. *Advances in Neural Information Processing Systems*, 32, 2019.

Y. Grinberg, H. Aboutalebi, M. Lyman-Abramovitch, B. Balle, and D. Precup. Learning predictive state representations from non-uniform sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

C. Gulcehre, T. Le Paine, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. In *International conference on learning representations*, 2019.

Z. D. Guo, S. Doroudi, and E. Brunskill. A pac rl algorithm for episodic pomdps. In *Artificial Intelligence and Statistics*, pages 510–518. PMLR, 2016.

Z. D. Guo, B. A. Pires, B. Piot, J.-B. Grill, F. Altché, R. Munos, and M. G. Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pages 3875–3886. PMLR, 2020.

Z. D. Guo, S. Thakoor, M. Pîslar, B. A. Pires, F. Altché, C. Tallec, A. Saade, D. Calandriello, J.-B. Grill, Y. Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *arXiv preprint arXiv:2206.08332*, 2022.

P. D. Handschutter, N. Gillis, and X. Siebert. A survey on deep matrix factorizations. *Computer Science Review*, 42:100423, nov 2021. doi: 10.1016/j.cosrev.2021. 100423. URL https://doi.org/10.1016%2Fj. cosrev.2021.100423.

M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

In *International conference on machine learning*, pages 448–456. PMLR, 2015.

M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

M. R. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the twenty-first international conference on Machine learning*, page 53, 2004.

J. Kwon, Y. Efroni, C. Caramanis, and S. Mannor. Rl for latent mdps: Regret guarantees and a lower bound. *Advances in Neural Information Processing Systems*, 34: 24523–24534, 2021.

C. L. Lan, J. Greaves, J. Farebrother, M. Rowland, F. Pedregosa, R. Agarwal, and M. G. Bellemare. A novel stochastic gradient descent algorithm for learning principal subspaces. *arXiv preprint arXiv:2212.04025*, 2022.

M. Littman and R. S. Sutton. Predictive representations of state. *Advances in neural information processing systems*, 14, 2001.

M. Liu, M. C. Machado, G. Tesauro, and M. Campbell. The eigenoption-critic framework. *arXiv preprint arXiv:1712.04065*, 2017.

Q. Liu, A. Chung, C. Szepesvári, and C. Jin. When is partially observable reinforcement learning not scary? *arXiv preprint arXiv:2204.08967*, 2022.

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

C. Lyle, M. Rowland, and W. Dabney. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022a.

C. Lyle, M. Rowland, W. Dabney, M. Kwiatkowska, and Y. Gal. Learning dynamics and generalization in reinforcement learning. *arXiv preprint arXiv:2206.02126*, 2022b.

M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pages 2295–2304. PMLR, 2017a.

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

M. C. Machado, M. G. Bellemare, and M. Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.

S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 553–560, 2005.

S. Mahadevan and M. Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10), 2007.

P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

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

O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

T. L. Paine, C. Gulcehre, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

R. Parr, C. Painter-Wakefield, L. Li, and M. Littman. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th international conference on Machine learning*, pages 737–744, 2007.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 752–759, 2008.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

M. Petrik. An analysis of laplacian methods for value function approximation in mdps. In *IJCAI*, pages 2574–2579, 2007.

T. Ren, T. Zhang, L. Lee, J. E. Gonzalez, D. Schuurmans, and B. Dai. Spectral decomposition representation for reinforcement learning. *arXiv preprint arXiv:2208.09515*, 2022.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *Proceedings of the twenty-first international conference on Machine learning*, page 88, 2004.

N. Roy, G. Gordon, and S. Thrun. Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40, 2005.

M. Rudary and S. Singh. A nonlinear predictive state representation. *Advances in neural information processing systems*, 16, 2003.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

S. Singh, M. James, and M. Rudary. Predictive state representations: A new theory for modeling dynamical systems. *arXiv preprint arXiv:1207.4167*, 2012.

S. P. Singh, M. L. Littman, N. K. Jong, D. Pardoe, and P. Stone. Learning predictive state representations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 712–719, 2003.

H. F. Song, A. Abdolmaleki, J. T. Springenberg, A. Clark, H. Soyer, J. W. Rae, S. Noury, A. Ahuja, S. Liu, D. Tirumala, et al. V-mpo: On-policy maximum a posteriori

policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019.

Z. Song, R. E. Parr, X. Liao, and L. Carin. Linear feature encoding for reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Y. Tang, Z. D. Guo, P. H. Richemond, B. Á. Pires, Y. Chandak, R. Munos, M. Rowland, M. G. Azar, C. L. Lan, C. Lyle, et al. Understanding self-predictive learning for reinforcement learning. *arXiv preprint arXiv:2212.03319*, 2022.

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

C. F. Van Loan. Generalizing the singular value decomposition. *SIAM Journal on numerical Analysis*, 13(1):76–83, 1976.

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

L. Wang, Q. Cai, Z. Yang, and Z. Wang. Embed to control partially observed systems: Representation learning with provable sample efficiency. *arXiv preprint arXiv:2205.13476*, 2022.

L. Weng. Exploration strategies in deep reinforcement learning. *Online: https://lilianweng. github. io/lil-log/2020/06/07/exploration-strategies-in-deep-reinforcement-learning. html*, 2020.

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

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

W. Zhan, M. Uehara, W. Sun, and J. D. Lee. Pac reinforcement learning for predictive state representations. *arXiv preprint arXiv:2207.05738*, 2022.

Representations and Exploration for Deep Reinforcement Learning using Singular Value Decomposition
(Supplementary Material)

A. Extended Related Work

Auxiliary Losses and Exploration Bonuses: Several prior works have analyzed properties of states features for TD learning (Parr et al., 2007; 2008; Song et al., 2016), studied different notions of optimality (Nachum et al., 2018; Bellemare et al., 2019; Allen et al., 2021), and the effect of auxiliary tasks on representation learning (Dabney et al., 2021; Lyle et al., 2021; 2022a;b). Specifically, similar to D-REX, auxiliary losses have been shown to help obtain higher expected returns quicker by learning better representations (Jaderberg et al., 2016; Mirowski et al., 2016; Hessel et al., 2019; Gregor et al., 2019; Guo et al., 2020; Erraqabi et al., 2022). However, unlike these methods, D-REX neither requires negative sampling, nor any target networks for representation learning nor does it require reconstruction at the raw observation or representation level.

Auxiliary losses can also provide intrinsic motivation signals to drive exploration (Houthooft et al., 2016; Pathak et al., 2017; Burda et al., 2018; Azar et al., 2019; Badia et al., 2020; Guo et al., 2022). Such intrinsic bonuses are also related to pseudo-counts of state visitations (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017). However, unlike these methods, D-REX does not require any sophisticated density estimation procedure and can directly use the norms of the learned representations to estimate pseudo-counts. We refer the readers to the work by Weng (2020) for an exhaustive review on various exploration strategies.

Decomposition Based Methods Following the seminal work by Mahadevan (2005); Mahadevan and Maggioni (2007) several researchers have explored the utility of spectral representation and decompositions in decision making in MDPs (Petrik, 2007; Wang et al., 2021; Behzadian et al., 2019), POMDPs (Hsu et al., 2012; Roy et al., 2005; Azizzadenesheli et al., 2016; Boots et al., 2011; Grinberg et al., 2018). PAC learnability and regret guarantees have also been developed for MDPs (Misra et al., 2020; Agarwal et al., 2020) and POMDPs (Wang et al., 2022; Kwon et al., 2021; Guo et al., 2016; Liu et al., 2022; Zhan et al., 2022) using various rank conditions on the dynamics. Complimentary to these results, our method is aimed at solving large-scale environments that present challenges both for representation learning and exploration.

B. Proofs

Theorem 1. *If the eigenvalues of $\mathbf{P}_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ are real and distinct,* [2] *then for any k, the top-k eigenvectors of $\mathbf{P}_\pi$ and $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ are the same.*

Proof. We first show that any eigenvector of $\mathbf{P}_\pi$ is also an eigenvector of $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$. Subsequently, we will show that the ordering of these eigenvectors, based on their associated eigenvalues, are also the same.

Let $v \in \mathbb{R}^{|S| \times 1}$ be an eigenvector of $\mathbf{P}_\pi$ and λ be its associated eigenvalue, i.e.,

$$\mathbf{P}_\pi v = \lambda v.$$

Now it can be observed that the eigenvectors for $\mathbf{P}_\pi$ and $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ are the same,

$$(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} v = \left(\sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi{}^t \right) v = \sum_{t=0}^{\infty} (\gamma \lambda)^t v.$$

Further, as $0 \leq \gamma < 1$ and $|\lambda| \leq 1$,

$$(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} v = \frac{1}{1 - \gamma \lambda} v.$$

Let $\lambda' := 1/(1 - \gamma \lambda)$ be the eigenvalue associated with v for the matrix $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$. As $|\lambda| \leq 1$, λ and λ' are monotonously related, i.e., larger values of λ imply larger values of λ' and the ordering of the top-k eigenvectors for both $\mathbf{P}_\pi$ and $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ are the same.

---

[2] we consider eigenvalues to be real and distinct to simplify the choice of 'top-k' eigenvalues.

$\square$

Theorem 2. $\mathbf{A_n}$ *is an unbiased and a consistent estimator of* $\mathbf{DP}_\pi$, *i.e.*,

$$\forall i,j, \quad \mathbb{E}\left[\mathbf{A_{n}}_{[i,j]}\right] = \mathbf{DP}_{\pi[i,j]}, \qquad\qquad \mathbf{A_{n}}_{[i,j]} \xrightarrow{a.s.} \mathbf{DP}_{\pi[i,j]}.$$

Proof. **Unbiased:**

$$\mathbb{E}\left[\mathbf{A_n}\right] = \mathbb{E}\left[\frac{1}{n}\mathbf{X}^\top\mathbf{Y}\right] = \mathbb{E}\left[\frac{1}{n}\mathbf{X}^\top\mathbb{E}[\mathbf{Y}|\mathbf{X}]\right].$$

In the tabular setting, as $\mathbb{E}[\mathbf{Y}|\mathbf{X}]$ is equal to the distribution of the next state Y given interaction using π from the state X, therefore $\mathbb{E}[\mathbf{Y}|\mathbf{X}] = \mathbf{P}_\pi$,

$$\mathbb{E}\left[\mathbf{A_n}\right] \stackrel{(a)}{=} \mathbb{E}\left[\frac{1}{n}\mathbf{X}^\top\mathbf{X}\mathbf{P}_\pi\right] = \mathbb{E}\left[\frac{1}{n}\mathbf{X}^\top\mathbf{X}\right]\mathbf{P}_\pi. \tag{16}$$

Now observe that because of one-hot encoding, $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$ is a diagonal matrix, where i^{th} diagonal entry consists of the normalized visitation counts for the state i. Therefore,

$$
\begin{aligned}
\mathbb{E}\left[\frac{1}{n}\mathbf{X}^\top\mathbf{X}\right]_{[i,i]} &= \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n} x_j x_j^\top\right]_{[i,i]} \\
&= \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n} 1_{\{s_j=s_i \wedge s_j=s_i\}}\right] \\
&= \mathbb{E}\left[1_{\{S=s_i\}}\right] \\
&= d(s_i).
\end{aligned}
$$

Therefore, $\mathbb{E}\left[\frac{1}{n}\mathbf{X}^\top\mathbf{X}\right] = \mathbf{D}$, and using (16),

$$\mathbb{E}\left[\mathbf{A_n}\right] = \mathbf{DP}_\pi.$$

Consistency:

$$
\begin{aligned}
\mathbf{DP}_{\pi[i,j]} &= \mathbf{D}_{[i,i]}\mathbf{P}_{\pi[i,j]} \\
&= d(s_i)\Pr(S'=s_j|S=s_i;\pi) \\
&= \Pr(S'=s_j, S=s_i;\pi)
\end{aligned}
$$

$$\mathbf{A_{n}}_{[i,j]} = \frac{1}{n}\mathbf{X}^\top\mathbf{Y}_{[i,j]} = \frac{1}{n}\sum_{k=1}^{n}(x_k y_k^\top)_{[i,j]}$$

As x_k and y_k are one-hot encoded vectors for the states S_k and S'_k,

$$\mathbf{A_{n}}_{[i,j]} = \frac{1}{n}\sum_{k=1}^{n} 1_{\{S_k=s_i \wedge S'_k=s_j\}}$$

Let $X_k = 1_{\{S_k=s_i \wedge S'_k=s_j\}}$, then we know from the strong law of large numbers,

$$\frac{1}{n}\sum_{k=0}^{\infty} X_k \xrightarrow{a.s.} \mathbb{E}[1_{\{S_k=s_i \wedge S'_k=s_j\}}] = \Pr(S'=s_j, S=s_i;\pi)$$

Therefore,

$$
\begin{aligned}
\forall i,j, \quad \mathbf{A_{n}}_{[i,j]} &\xrightarrow{a.s.} \Pr(S'=s_j, S=s_i;\pi) \\
&= \mathbf{DP}_{\pi[i,j]}.
\end{aligned}
$$

$\square$

Theorem 3. *If* $\mathbf{P}_\pi \mathbf{P}_\pi^\top$ *and* $\mathbf{D}$ *are invertible, then for* $\alpha_s := (\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1}_{[s,s]}$,

$$\|f_\mathbf{U}(x_s)\|^2_{\mathbf{\Lambda}^{-1}} = \frac{\alpha_s}{d(s)^2}. \tag{7}$$

Proof. First notice that,

$$\mathbf{DP}_\pi = \mathbf{U\Sigma V}^\top$$
$$\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top = \mathbf{U\Sigma V}^\top \mathbf{V\Sigma}^\top \mathbf{U}^\top$$
$$= \mathbf{U\Sigma^2 U}^\top, \qquad\qquad \because \mathbf{V}^\top \mathbf{V} = \mathbf{I}$$
$$= \mathbf{U\Lambda U}^\top \qquad\qquad \text{where, } \mathbf{\Lambda} := \mathbf{\Sigma^2}.$$

Recall that for any matrix M with full column-rank, it's pseudo-inverse is given by $\mathbf{M}^\dagger := (\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top$ such that $\mathbf{M}^\dagger \mathbf{M} = \mathbf{I}$. As $(\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top)^\top = \mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top$,

$$(\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top)^\dagger = (\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top \mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top)^{-1} \mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top$$
$$= (\mathbf{U\Lambda U}^\top \mathbf{U\Lambda U}^\top)^{-1} \mathbf{U\Lambda U}^\top$$
$$\overset{(a)}{=} (\mathbf{U\Lambda^2 U}^\top)^{-1} \mathbf{U\Lambda U}^\top$$
$$\overset{(b)}{=} \mathbf{U\Lambda^{-2} U}^\top \mathbf{U\Lambda U}^\top$$
$$= \mathbf{U\Lambda^{-1} U}^\top, \tag{17}$$

where (a) follows from the fact that $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$ and (b) follows from the fact that $\mathbf{U}^{-1} = \mathbf{U}^\top$.

Alternatively, under the assumption that $\mathbf{P}_\pi \mathbf{P}_\pi^\top$ and $\mathbf{D}$ are invertible,

$$(\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top)^\dagger = ((\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top)^\top (\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top))^{-1} (\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}^\top)^\top$$
$$= (\mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D^2 P}_\pi \mathbf{P}_\pi^\top \mathbf{D})^{-1} \mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}$$
$$= \mathbf{D^{-1}}(\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1} \mathbf{D^{-2}}(\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1} \mathbf{D^{-1}} \mathbf{DP}_\pi \mathbf{P}_\pi^\top \mathbf{D}$$
$$= \mathbf{D^{-1}}(\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1} \mathbf{D^{-1}}. \tag{18}$$

Combining (17) and (18),

$$\mathbf{U\Lambda^{-1} U}^\top = \mathbf{D^{-1}}(\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1} \mathbf{D^{-1}}.$$

In the tabular setting, $x_s \in \mathbb{R}^{|S| \times 1}$ correspond to the one-hot feature encoding for the state s, and the linear function $f_U(x_s) = \mathbf{U}^\top x_s$

$$\langle f_U(x_s), f_U(x_s) \rangle_{\mathbf{\Lambda^{-1}}} = x_s^\top \mathbf{U\Lambda^{-1} U}^\top x_s$$
$$= x_s^\top \mathbf{D^{-1}}(\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1} \mathbf{D^{-1}} x_s$$
$$\overset{(a)}{=} \frac{\alpha_s}{d(s)^2},$$

where, $\alpha_s = (\mathbf{P}_\pi \mathbf{P}_\pi^\top)^{-1}_{[s,s]}$, and the simplification in step (a) follows because x_s is one-hot and $\mathbf{D}$ is a diagonal matrix. $\square$

Theorem 4. *Gradient of* $\mathcal{L}_{\mathrm{diag}}(\widehat{\mathbf{\Sigma}}(\theta))$ *is an unbiased estimator of* $\mathcal{L}_{\mathrm{diag}}(\mathbf{\Sigma}(\theta))$, *and gradient of* $\mathcal{L}_{\mathrm{off}}(\widehat{\mathbf{\Sigma}}(\theta))$ *is in general a biased estimate of* $\mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta))$, *i.e.,*

$$\mathbb{E}\left[\partial_\theta \mathcal{L}_{\mathrm{diag}}\left(\widehat{\mathbf{\Sigma}}(\theta)\right)\right] = \partial_\theta \mathcal{L}_{\mathrm{diag}}(\mathbf{\Sigma}(\theta)),$$
$$\mathbb{E}\left[\partial_\theta \mathcal{L}_{\mathrm{off}}\left(\widehat{\mathbf{\Sigma}}(\theta)\right)\right] \neq \partial_\theta \mathcal{L}_{\mathrm{off}}(\mathbf{\Sigma}(\theta)),$$

where the expectation is over the randomness of the mini-batch.

Proof. We re-write (8) as

$$\boldsymbol{\Sigma}(\theta) := \mathbb{E}\left[f_{\theta_1}(S)g_{\theta_2}(S')^\top\right],$$

where the expectation is over the distribution specified by the n sampled transition tuples in $\mathcal{D}$, i.e., $\boldsymbol{\Sigma}(\theta)$ is the sample average of $f_{\theta_1}(s_i)g_{\theta_2}(s_i')^\top$ across the n samples in $\mathcal{D}$.

From (10) and (12),

$$
\begin{aligned}
\mathbb{E}\left[\partial_\theta \mathcal{L}_{\mathrm{diag}}\left(\widehat{\boldsymbol{\Sigma}}(\theta)\right)\right] &= \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^k \partial_\theta \widehat{\boldsymbol{\Sigma}}(\theta)_{[i,i]}\right] \\
&\overset{(a)}{=} \frac{1}{k}\sum_{i=1}^k \partial_\theta \mathbb{E}\left[\widehat{\boldsymbol{\Sigma}}(\theta)_{[i,i]}\right] \\
&= \frac{1}{k}\sum_{i=1}^k \partial_\theta \boldsymbol{\Sigma}(\theta)_{[i,i]} \\
&= \partial_\theta \mathcal{L}_{\mathrm{diag}}(\boldsymbol{\Sigma}(\theta)),
\end{aligned}
$$

where (a) follows because we are looking at the case where the sampling distribution is fixed.

In the following we use a counter-example to show that the gradient for the mini-batch version of off-diagonal based loss does not provide an unbiased estimator in general. For this counter-example, let the mini-batch size $b = 1$. Therefore,

$$
\begin{aligned}
\mathbb{E}\left[\partial_\theta \mathcal{L}_{\mathrm{off}}\left(\widehat{\boldsymbol{\Sigma}}(\theta)\right)\right] &= \mathbb{E}\left[\frac{1}{k^2-k}\sum_{\substack{i,j=1 \\ i\neq j}}^k \partial_\theta \widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]}^2\right], \\
&= \frac{1}{k^2-k}\sum_{\substack{i,j=1 \\ i\neq j}}^k \partial_\theta \mathbb{E}\left[\widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]}^2\right], \\
&\overset{(b)}{=} \frac{1}{k^2-k}\sum_{\substack{i,j=1 \\ i\neq j}}^k \partial_\theta \mathbb{E}\left[f_{\theta_1}(S)g_{\theta_2}(S')_{[i,j]}^{\top\,2}\right], \\
&\textcolor{red}{\neq} \frac{1}{k^2-k}\sum_{\substack{i,j=1 \\ i\neq j}}^k \partial_\theta \mathbb{E}\left[f_{\theta_1}(S)g_{\theta_2}(S')_{[i,j]}^{\top}\right]^{\textcolor{red}{2}}, \\
&= \partial_\theta \mathcal{L}_{\mathrm{off}}\left(\boldsymbol{\Sigma}(\theta)\right),
\end{aligned}
$$

where the step (b) follows because the mini-batch size being considered is 1, and the step marked in red *fails* because for a random variable X, in general $\mathbb{E}[X^2] \neq \mathbb{E}[X]^2$.

$\square$

Theorem 5. *Gradient $\partial_\theta \widehat{\mathcal{L}}_{\mathrm{off}}(\theta)$ is an unbiased estimator of $\partial_\theta \mathcal{L}_{\mathrm{off}}(\boldsymbol{\Sigma}(\theta))$, i.e.,*

$$\mathbb{E}\left[\partial_\theta \widehat{\mathcal{L}}_{\mathrm{off}}(\theta)\right] = \partial_\theta \mathcal{L}_{\mathrm{off}}(\boldsymbol{\Sigma}(\theta)).$$

Proof.

$$\widehat{\mathcal{L}}_{\mathrm{off}}(\theta) := \underbrace{\frac{1}{(k^2-k)}\sum_{\substack{i,j=1 \\ i\neq j}}^k \widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]}\ \mathrm{sg}\left(\widehat{\widehat{\boldsymbol{\Sigma}}}(\theta)_{[i,j]}\right)}_{(I)} + \underbrace{\frac{1}{(k^2-k)}\sum_{\substack{i,j=1 \\ i\neq j}}^k \widehat{\widehat{\boldsymbol{\Sigma}}}(\theta)_{[i,j]}\ \mathrm{sg}\left(\widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]}\right)}_{(II)},$$

$$\mathbb{E}\left[\partial_\theta(I)\right] := \frac{1}{(k^2-k)} \sum_{\substack{i,j=1\\i\neq j}}^{k} \mathbb{E}\left[\partial_\theta\widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]} \;\; \mathrm{sg}\left(\widehat{\widehat{\boldsymbol{\Sigma}}}(\theta)_{[i,j]}\right)\right],$$

$$\stackrel{(a)}{=} \frac{1}{(k^2-k)} \sum_{\substack{i,j=1\\i\neq j}}^{k} \mathbb{E}\left[\partial_\theta\widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]}\right] \mathbb{E}\left[\widehat{\widehat{\boldsymbol{\Sigma}}}(\theta)_{[i,j]}\right],$$

$$= \frac{1}{(k^2-k)} \sum_{\substack{i,j=1\\i\neq j}}^{k} \partial_\theta \mathbb{E}\left[\widehat{\boldsymbol{\Sigma}}(\theta)_{[i,j]}\right] \mathbb{E}\left[\widehat{\widehat{\boldsymbol{\Sigma}}}(\theta)_{[i,j]}\right],$$

$$= \frac{1}{(k^2-k)} \sum_{\substack{i,j=1\\i\neq j}}^{k} \partial_\theta\boldsymbol{\Sigma}(\theta)_{[i,j]}\boldsymbol{\Sigma}(\theta)_{[i,j]},$$

$$= \frac{1}{2}\partial_\theta\left(\frac{1}{(k^2-k)} \sum_{\substack{i,j=1\\i\neq j}}^{k} \boldsymbol{\Sigma}(\theta)_{[i,j]}^2\right),$$

$$= \frac{1}{2}\partial_\theta\mathcal{L}_{\mathrm{off}}\left(\boldsymbol{\Sigma}(\theta)\right),$$

where step (a) follows because $\widehat{\boldsymbol{\Sigma}}(\theta)$ and $\widehat{\widehat{\boldsymbol{\Sigma}}}(\theta)$ are independently computed using different batches of data. Similarly, it can be observed that $\mathbb{E}\left[\partial_\theta(I)\right] = \mathbb{E}\left[\partial_\theta(II)\right]$. Therefore,

$$\mathbb{E}\left[\partial_\theta\widehat{\mathcal{L}}_{\mathrm{off}}(\theta)\right] = \mathbb{E}\left[\partial_\theta(I)\right] + \mathbb{E}\left[\partial_\theta(II)\right]$$
$$= \partial_\theta\mathcal{L}_{\mathrm{off}}(\boldsymbol{\Sigma}(\theta)).$$

$\square$

C. Experimental details

We provide further details on the algorithmic and implementation details of deep RL experiments in the paper.

DM-Lab-30 environments. DM-Lab-30 is a set of visually complex maze navigation tasks (Beattie et al., 2016). The only input the agent receives is an image of the first-person view of the environment, a natural language instruction (in some tasks), and the reward. Agents can provide multiple simultaneous actions to control movement (forward/back, strafe left/right, crouch, jump), looking (up/down, left/right) and tagging (in laser tag levels with opponent bots). DM-Lab-30 has proved challenging for model-free RL algorithms (Beattie et al., 2016; Guo et al., 2020).

DM-Hard-8 environments: DM-Hard-8 is a set of similar visually complex maze navigation tasks with sparse reward, and is commonly used as a test bed for exploration (Guo et al., 2022). This benchmark comprises of 8 hard exploration tasks aimed at emphasizing the challenges encountered when learning from sparse rewards in a procedurally-generated 3-D world with partial observability, continuous control, and highly variable initial conditions. Each task requires reaching an apple in the environment through interaction with specific objects in its environment. Only on reaching the apple provides a reward to the agent. Being procedurally generated, properties such as object shapes, colors, and positions are different in every episode. The agent sees only the first-person view from its position.

Agent architecture. Since the environment is partially observable, we adopt a recurrent network architecture following prior work such as (Beattie et al., 2016; Song et al., 2019; Guo et al., 2020). In particular, the agent uses a convolutional neural network inside $\mathcal{F}_1 : \mathcal{O} \to \mathbb{R}^k$ to process the image in the input $o_t \in \mathcal{O}$ into latent embedding. The observation contains (i) the force the agent's hand is currently applying as a fraction of total grip strength, a single scalar between 0 and
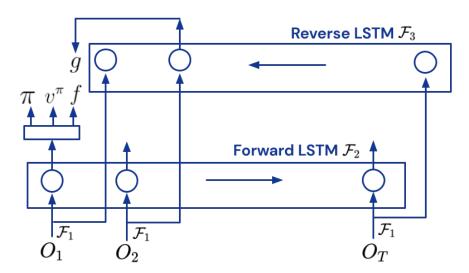
Figure 8: Recall that observation O_t contains the previous action A_{t-1}. Let $\mathcal{F}_1(O_i)$ be the embedding for the observation at timestep i. Let the belief state $b_t = \mathcal{F}_2(\mathcal{F}_1(O_0), \mathcal{F}_1(O_1), ..., \mathcal{F}_1(O_t))$ obtained using a forward LSTM. Let $\mathcal{F}_3$ be a reverse LSTM, such that $b'_t = \mathcal{F}_3(\mathcal{F}_1(O_T), \mathcal{F}_1(O_{T-1}), ..., \mathcal{F}_1(O_t))$. Then $f_{\theta_1}(h_t) \in \mathbb{R}^k$ in (15) is defined to be $f_{\theta_1}(b_t)$, and $g_{\theta_2}(\tau_t)$ is defined to be $g_{\theta_2}(b'_{t+1})$.

1; (ii) a boolean indicating whether the hand is currently holding an object; (iii) the distance of the agent's hand from the main body; (iv) the last action taken by the agent; each of these are embedded using a linear projection to 20 dimensions followed by a ReLU. (v) the previous reward obtained by the agent, passed by a signed hyperbolic transformation; (vi) a text instruction specific to the task currently being solved, with each word embedded into 20 dimensions and processed sequentially by an LSTM (Hochreiter and Schmidhuber, 1997) to an embedding of size 64. All of these quantities, along with the output of $\mathcal{F}_1$, are concatenated and passed through a linear layer to an embedding of size 512. An LSTM $\mathcal{F}_2$ with the embedding size 512 processes each of these observation representations sequentially to form the recurrent history representation. The core output of the LSTM is the belief $b_t = \mathcal{F}_2(h_t) \in \mathbb{R}^k$. Using the notation from the main paper, we can understand the belief as a function of the history $h_t = (o_0, a_0...o_t)$, thanks to the recurrent nature of the agent architecture.

Finally, we calculate value functions $v_\theta(b_t)$ and policy $\pi_\theta(a_t|b_t)$ based on the belief using MLP heads. Further, all the algorithms are built upon VMPO with a similar reward normalization scheme as in RND (Burda et al., 2018) and we normalize the raw rewards by an EMA estimate of its standard deviation. See Appendix A.3 in the work by Guo et al. (2022) for the exact procedure. For DM-Hard8 environments, rewards and bonuses are normalized separately before combining them using λ_b. That is, let R_t and Bonus_t be the normalized reward and bonus, then the effective reward for timestep t is, $R_t + \lambda_b \, \text{Bonus}_t$, where λ_b is a hyper-parameter.

When an algorithm makes use of additional representation loss Rep_{loss} then it is combined with the RL_{loss} as following, $\text{RL}_{\text{loss}} + w_{\text{loss}}\text{Rep}_{\text{loss}}$, where w_{loss} is a hyper-parameter, and RL_{loss} is the loss provided by the baseline VMPO for learning the policy and critic.

Baseline VMPO. VMPO is a model-free RL algorithm (Song et al., 2019) that achieves state-of-the-art performance in a number of challenging RL benchmarks. At its core, VMPO is a policy optimization algorithm. Importantly, the construction of VMPO updates are based on a number of prior inventions in the policy optimization literature, such as trust region optimization (Schulman et al., 2015; 2017), regression-based updates and adjusting trust regions based on primal-dual optimizations (Abdolmaleki et al., 2018) and multi-step estimation of advantage functions (Espeholt et al., 2018). We refer readers to (Song et al., 2019) for a complete description of the algorithm and its hyper-parameters. The original VMPO algorithm adopts pixel-control as an auxiliary objective (Beattie et al., 2016) in the DM-Lab-30 experiments; we do not apply such an auxiliary loss in our experiment to avoid confounding effects. Further, there is no exploration bonus. That is, $\lambda_b = w_{\text{loss}} = 0$.

De-Rex. This is the proposed method that provides a unified way to learn representation and also obtain pseudo counts for exploration. *De-Rex representation* only employs the representation learning loss as an auxiliary task, and there is no exploratory bonus, i.e., $\lambda_b = 0$. The representation loss $\text{Rep}_{\text{loss}} = \mathcal{L}(\mathbf{\Sigma}(\theta))$ is based on (11) and is combined with the VMPO baseline. For implementing (15), we base the functions f_{θ_1} and g_{θ_2} on forward and backward LSTMs, as illustrated in Figure 8. We observed that defining the representation loss as an auxiliary loss on a separate projection/transformation of belief b_t is more beneficial than directly applying the loss on b_t, which is being used directly by the policy and the value functions.

In comparison, *De-Rex Exploration* employs both representation learning loss and the exploratory bonus. Here the Bonus_t corresponds to the norm of the representation $f_{\theta_1}(h_t) \in \mathbb{R}^k$ as discussed in Section 5. In Tables 1 and 2 we provide more details about the hyper-parameters.

Hyper-parameter	Value
λ_r	10.0 (Tuned in smaller experiments)
λ_b	(N/A)
w_{loss}	0.01 (Hypersweep $\{1, 0.1, 0.01, 0.001\}$)
k	512 (Hypersweep $\{1024, 512, 64\}$)
Batch Size	256

Table 1: Hyper-parameters for De-Rex Representation for DMLab30

Hyper-parameter	Value
λ_r	10.0
λ_b	0.001 (Hypersweep $\{1, 0.1, 0.01, 0.001\}$)
w_{loss}	0.01 (Hypersweep $\{1, 0.1, 0.01, 0.001\}$)
k	64 (Hypersweep $\{512, 256, 64, 8\}$)
Batch Size	256

Table 2: Hyper-parameters for De-Rex Exploration for DMHard-8

Laplacian graph drawing (Wu et al., 2018) : This is the baseline decomposition method that decomposes the Laplacian associated with the transition matrix. Let u and v be state and next-state pairs, then their proposed loss is given by

$$\mathcal{L}_{\text{diag}} := \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} (f_{\theta_1}(u_i)_j - f_{\theta_1}(v_i)_j)^2$$

$$\mathcal{L}_{\text{off}} := \frac{1}{nk^2} \sum_{i=1}^{n} \sum_{j=1}^{k} \sum_{l=1}^{k} (f_{\theta_1}(u_i)_j f_{\theta_1}(u_i)_l - \delta_{j,l})(f_{\theta_1}(v_i)_j f_{\theta_1}(v_i)_l - \delta_{j,l})$$

$$\text{Rep}_{\text{loss}} := \mathcal{L}_{\text{diag}} + \lambda_r \mathcal{L}_{\text{off}}$$

While the original formulation was defined for MDP, where u and v correspond to the state and the next state. However, as we are dealing with POMDPs, we let the state be the history so far, therefore we set $u = h$ and $v = h'$. The architecture for the function f_{θ_1} is the same as the one used for De-Rex. This Laplacian baseline does not provide any means for exploration and hence $\lambda_b = 0$ throughout. For λ_r, we searched between $[1, 10]$ and found 10 to work the best. Other hyper-parameters were similar to De-Rex.

Random Network Distillation (RND). RND (Burda et al., 2018) provides an exploration strategy that generates an exploration bonus based on the prediction between the current network $\xi_\theta(o_t) \in \mathbb{R}^k$ and a randomly initialized network $\xi_{\theta_0}(o_t) \in \mathbb{R}^k$. Note that θ_0 is randomly initialized and is not trained over time. The current network θ is trained to minimize the squared error $l_\theta(o_t) = \|\xi_\theta(o_t) - \xi_{\theta_0}(o_t)\|_2^2$. Meanwhile, $l_\theta(o_t)$ is also used as an exploration bonus.
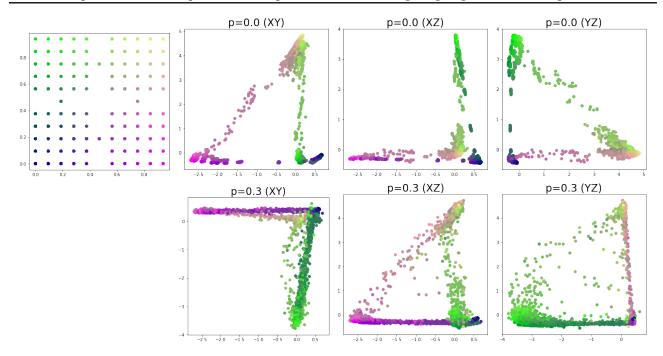
Figure 9: This plot is analogous to Figure 2(B) and Figure 2(C) and extends the visualization technique to the POMDP setting. We plot the representations for all possible (partial) histories in the data collected **Top-Left** Ground truth color to represent the underlying true state s. **Top-row** Illustration of the learned three-dimensional representations for the histories in the 4-rooms (POMDP) setting where $p = 0.0$. To view 3D representations, we plot slices (XY, XZ, YZ) of the axes. **Bottom-row** Illustration of the learned three-dimensional representations for the histories in the 4-rooms (POMDP) setting where $p = 0.3$. To view 3D representations, we plot slices (XY, XZ, YZ) of the axes. **Legend:** (Partial) histories that have s as the true underlying state at the end of their sequence of observations have their representations share the same color as the reference color for s. Therefore, as there are multiple histories that end at a given state, for a single color of the true state there are multiple learned representations with the same color.

D. Ablation Study for 4-Rooms POMDP Domain

In Figures 2 and 3 the representations learned and the bonuses obtained were visualized for the fully-observable setting. In this section, we aim to further extend this qualitative assessment by performing ablations on the partially observable setting. To restrict the focus on just the representation learning component and the bonuses constructed, we consider a partially observable extension of the 4 rooms domains. This provides a controlled environment to test the effectiveness and limitations of the proposed De-Rex method.

4 rooms (POMDP): In this domain, observation O_t at time t corresponds to an image of size $(30, 30, 3)$ that has the top-down view of the domain (similar to Figure 3 (B)), albeit that with probability p the agent's marker (red dot) is completely hidden. Therefore, when the marker is hidden, the observation O_t does not provide any information about the location of the agent, and the location needs to be inferred using the history. Naturally, a higher value of p induces a higher degree of partial observability, and $p = 0$ indicates full observability (the proposed De-Rex method does not assume full observability and still uses history-dependent representations). The action set corresponds to the four directions of movement, and the horizon length is 250.

Therefore, this domain provides a controlled setup to understand the method across different degrees of partial observability. In the following, we provide qualitative analysis of the representations and bonuses learned by our method, across different values of p. Data was collected using a uniform random policy.

Representations: In the fully observable case, the proposed method used a representation dimension of 2 (Figure 3). For the partially observable setting, we found representation dimensions of 3 and 4 to be useful for $p < 0.5$ and $p > 0.5$ respectively. In Figure 9 we illustrate the effectiveness of learning representations using the De-Rex approach. Architecture
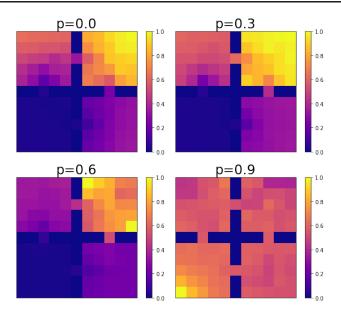
Figure 10: This plot is analogous to Figure 2(D) and Figure 3(D) and extends the visualization technique to the POMDP setting. The ground truth state visitation is the same as in Figure 2(A). **Top-left** Bonuses constructed using learned representations of the history when $p = 0.0$. **Top-right** Bonuses constructed using learned representations of the history when $p = 0.3$. **Bottom-left** Bonuses constructed using learned representations of the history when $p = 0.6$. **Bottom-right** Bonuses constructed using learned representations of the history when $p = 0.9$. **Legend:** Brighter color indicates higher value bonus. Colors for each of the true state s is computed based on the average of bonuses from all (partial) histories that have s as the true underlying state at the end of their sequence of observations.

for our f and g function follows the one illustrated in Figure 8, where the observation are individually processed via a conv-net, before passing it to a forward and a reverse LSTM to encode histories and futures, respectively. Outputs of these LSTMs are then subjected to a one-layer MLP to form f and g.

A sufficient statistic for any history is the ground truth state at the end of the history. It can be observed that for the histories that are ending at similar ground truth state (unknown to the agent), the proposed De-Rex method is still able to provide similar representations for those histories.

Bonuses: In Figure 10 we plot the bonuses obtained using the De-Rex procedure. It can be observed that when partial observability is not extreme, the proposed method is able to provide pseudo-counts for the underlying true state even when the true state is unknown. This can be attributed to the fact that De-Rex provides a joint mechanism for both learning a representation to capture the underlying sufficient statistic, and for estimating pseudo-counts using those represetnations.

Belief Distribution: While Figure 9 provides one way to assess the quality of the learned distributions, it can be used to visualize representations in higher dimensions. Therefore, in Figures 11, 12, 13, and 14 we provide an alternate way to assess the quality of the learned representations.

For these figures, we learned a classification model (using a single-layer neural network) that aims to classify the true underlying state at the end of the observation sequence of a given (partial) history using the representation of that (partial) history. This provides us with a proxy for how well can the learned representations be used as to decode beliefs over the underlying true state.

As expected, when $p = 0.0$ there is no partial observability and agent learns to use (ignore) history to obtain representations which can be used to precisely decode the belief over the ground truth location of the agent. As the degree of partial observability increases, it can be observed that the belief distribution get more diffused as there is less certainty about the agent's location. Nonetheless, De-Rex provides representations that can be used to decode true underlying states within close proximity.
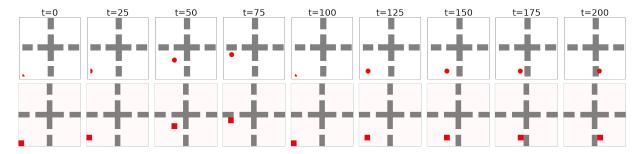
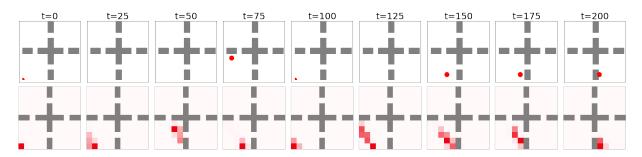Figure 11: Beliefs for a given trial under the POMDP setting with $p = 0.0$.

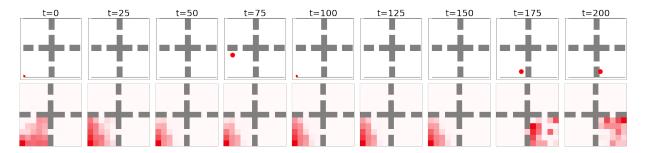Figure 12: Beliefs for a given trial under the POMDP setting with $p = 0.3$.

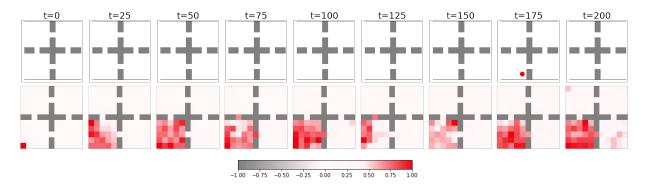Figure 13: Beliefs for a given trial under the POMDP setting with $p = 0.6$.

Figure 14: Beliefs for a given trial under the POMDP setting with $p = 0.9$. Top-rows illustrate the snapshots of the observations that were given to the agent at timesteps $t = \{0, 25, 50, 75, 100, 125, 150, 175, 200\}$. The bottom rows denote the belief decoded from the representation of the history till timestep t. See text for more details. **Legend:** All plots on this page share the same legend. For the bottom rows, the deeper red color indicates a higher degree of belief. For the top row, the red color indicates the location of the agent. Higher p indicates a higher degree of partial-observability and thus more frames have the agent missing.
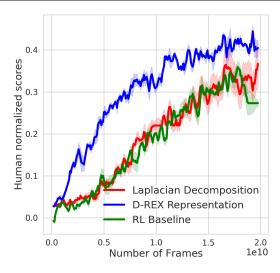
Figure 15: Human normalized scores aggregated across all the 30 tasks on DMLab30 using 3 seeds

E. More Large-Scale Results

E.1. DMLab30

We measure the performance of each task in DMLab30 environment with the human-normalized performance $(z_i - u_i)/(h_i - u_i)$ with $1 \leq i \leq 30$, where u_i and h_i are the raw score performance of random policy and humans, and z_i is the score obtained by the agent. A normalized score of 1 indicates that the agent performs as well as humans on the task.

In Figure 15 we plot the human normalized score aggregated across all the domains and in Figure 18 we plot the learning curves for each of the tasks individually. It can be observed in both that when dealing with complex environments, the baseline decomposition-based method fails to provide significant improvement. In comparison, the proposed De-Rex method has the potential to perform much better.

E.2. DM-Hard-8

To better understand how much can just learning representations help with hard exploration tasks, we perform ablation studies on the DM-Hard-8 tasks for the SVD representation and Laplacian decomposition method. For both the methods we plot the results for different values of (a) w_{loss} and (b) dimensions of the output of f_{θ_1} on which the decomposition is done.

As it can be observed from the plots in Figure 16 and 17, just by using representation learning alone, D-Rex is able to solve at most 2 tasks, and the Laplacian decomposition method can solve only one task. This illustrates the importance of constructing good pseudo-counts to aid exploration, as evident in Figure 7 by the De-Rex Exploration method.
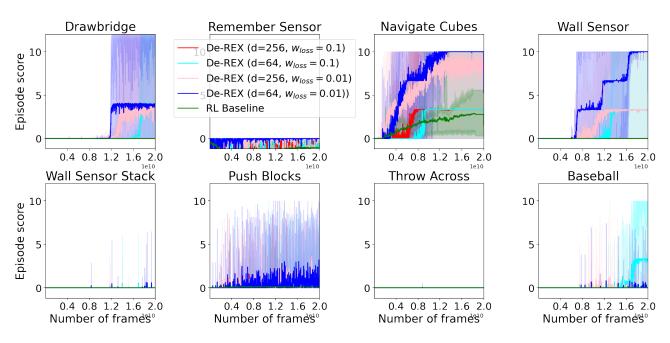
Figure 16: Parameter Study: De-Rex Representation. Sensitivity on weighting the SVD loss in the representation (w_{loss}) and the dimension of the SVD decomposition (d)
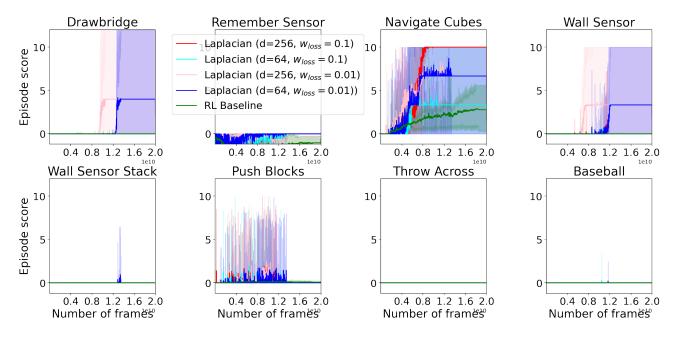

Figure 17: Lapacian Decomposition Representation. Sensitivity on weighting the decomposition loss in the representation (w_{loss}) and the dimension of the SVD decomposition (d)
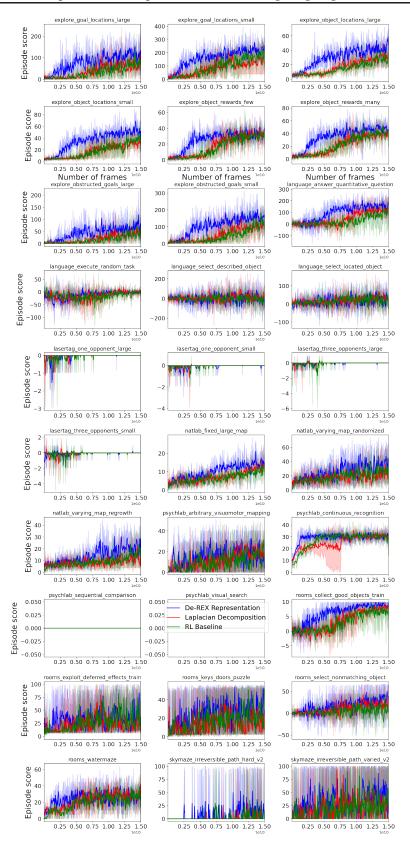
Figure 18: Individual learning curves for De-Rex Representation, Laplacian decomposition, and the VMPO baseline methods on the DMLab30 tasks.