Feature Specialization and Clustering Improves Hierarchical Subtask Learning

Neale Van Stralen, Seung Hyun Kim, Huy T. Tran, Girish Chowdhary
University of Illinois at Urbana-Champaign
United States
{nealeav2,skim449,huytran1,girishc}@illinois.edu

ABSTRACT

Eigendecomposition methods have been shown to generate sets of useful options which improve learning speed when used in hierarchical reinforcement learning. However, these methods focus on navigation by learning reward-agnostic representations and struggle when presented with environments with dynamic reward structures, such as adversarial agents. Taking inspiration from mammals, which are known to maintain specialized groupings of cells to perform complex planning, we propose leveraging the linear feature space of the successor features framework to independently encode spatial and reward information of an environment. We show that subsequent decomposition of this representation results in options which separately relate to spatial or rewarding features, allowing for complex spatial planning around dynamic objects, such as adversaries. We also propose the use of principle component analysis to perform this decomposition, due to its use of a clustering basis, which we show better identifies options for spatial planning than common methods. We combine these ideas in our Specialized Neurons and Clustering Architecture (SNAC), which uses a split successor feature encoding and cluster-based decomposition, and empirically demonstrate that this architecture produces options that are sensitive to adversarial agents, thus improving learning speed and performance in challenging and dynamic spatial planning tasks.

KEYWORDS

Reinforcement Learning, Hierarchical Reinforcement Learning, Options Learning

1 INTRODUCTION

Mammals often solve complex problems by decomposing them into distinct subtasks. For example, in navigation, intermediate goal locations are often identified and used for efficient planning. There is now strong evidence that this type of hierarchical navigation leverages specialized neurons in the hippocampus, such as place and object cells which fire near specific spatial locations or important objects, respectively, to create a topological representation of the environment [5, 7, 31, 34]. Exactly how mammals utilize these representations to make decisions remains an active area of research, but the strength of mammalian pattern recognition suggests that clustering may play an important role in decision making [26]. Modern hierarchical reinforcement learning (HRL) agents mimic the overall structure of planning over subtasks (options) but currently lack feature specialization and effective cluster-based decomposition, which may be limiting performance in complex environments such as those with dynamic objects or adversaries.

Here, we ask the question, *"Can feature specialization and cluster-based options enable HRL agents to decompose environments into a useful task subspace, and will such decomposition result in significantly improved learning speed and performance?"* We address this question within the context of successor features (SFs) [16], the function approximation variant of the successor representation (SR) [9]. We focus on SFs because recent work has shown that eigendecomposition of SFs can be used to generate a useful set of options in a two-step approach that supports generalization to new tasks [22]. Furthermore, SR has been shown to learn topological maps similar to those in place cells of mice [23], suggesting that SFs inherently mimic certain aspects of spatial navigation seen in mammals. However, two key limitations prevent existing eigendecomposition methods for HRL, which we refer to as eigenoption methods, from being applied in complex and dynamic domains.

The first limitation is that existing eigenoption methods focus on learning reward-agnostic options which work well for navigation, but have limited sensitivity to dynamic elements of the environment, such as adversaries or mobile rewards. We address this limitation by improving the ability of SFs to model dynamic elements in the environment and subsequently generate options that specifically interact with those elements. Our key idea is to mimic the task-specialization of the hippocampus by splitting the latent SF representation into specialized components, each of which encodes a unique element of the environment, such as spatial location or adversary information. We then decompose each component to generate a set of specialized options that relate to those unique elements.

The second limitation is that existing eigenoption methods focus on identifying unique states, commonly bottlenecks from minimum cuts, to be used as option sub-goals [2, 20–22]. While options leading to bottleneck states can aid in navigation, they cannot sufficiently model complex tasks that require a more diverse set of options, such as navigating around an enemy in a room [15, 20]. We address this limitation by proposing use of a different graph decomposition method, principle component analysis (PCA), which produces a basis for k-means clustering of an environments' states that captures information about cluster centroids and intra-cluster structure. We use this basis to generate options that allow agents to navigate to key clusters in the environment and also perform detailed movement and control within those clusters, allowing for more diverse behavior.

Our main contribution is to combine these two ideas, task specialization through a split SF encoding and cluster-based option

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, da Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

discovery through PCA, into an HRL architecture for solving complex tasks in dynamic environments. We show that this architecture, SNAC: Specialized Neurons And Clustering Architecture, significantly improves learning speed and performance over existing eigenoption and general HRL methods. In particular, we show that our method significantly outperforms baselines in a grid-world environment with moving adversaries, due its ability to better handle dynamic objects. We also show that our method improves performance over baselines in Atari, although there is significant room for further improvement since current exploration methods for learning SFs struggle with large environments like this. A key takeaway is that even though existing eigenoption methods are reward-agnostic, we demonstrate that directly including reward modelling in the representation learning process improves performance and is a worthwhile avenue of research for decomposition-based HRL.

2 BACKGROUND AND RELATED WORK

We formulate our problem as a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R)$. Here, $\mathcal{S}$ is the state space of the environment, $\mathcal{A}$ is the action space of the agent, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. RL algorithms seek to find a policy $\pi(a|s) : \mathcal{S} \rightarrow \mathcal{A}$ that selects actions over the state space that maximize the expected discounted return $\mathbb{E}_\pi[\sum_{i=0}^{\inf} \gamma^i r_{t+i} | s_t = s]$.

HRL seeks to improve learning speed and performance in problems with large state-action spaces by finding and exploiting hierarchies within the environment [2, 16, 25]. The semi-MDP formulation is a leading framework for HRL which decomposes a policy into Markovian options [32]. A Markovian option $\omega \in \Omega$ is a tuple $(\mathcal{I}_\omega, \pi_\omega, \beta_\omega)$, where $\mathcal{I}_\omega$ is an initiation condition, π_ω is a policy for the option, β_ω is a termination condition, and Ω is the set of all options. In the semi-MDP framework, options are called in sequence by a hierarchical controller (HC) using a policy $\pi : S \rightarrow \Omega$ to solve complex environments through smaller, more manageable subtasks.

2.1 Successor Features

SR provides an alternative structure for RL that focuses on separating environment dynamics from rewards [9]. This separation allows one to recompute value functions under dynamically changing rewards, which can be used to enable fast policy adaptation to new reward structures [9, 16, 38]. SR accomplishes this separation by correlating proximal spatial locations to one another, similar to how mammals encode temporally adjacent states with similar brain firing in place cells [23, 31]. More formally, the SR is the expected discounted future state occupancy, defined as,

$$M^\pi(s, a, s') = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \mathbb{1}[s_{t+i+1} = s'] | s_t = s, a_t = a \right], \quad (1)$$

where $\mathbb{1}[\cdot]$ is one when the input argument is true. The resulting SR defines an adjacency matrix for a fully connected graph that models the MDP, where the $(i, j)^{th}$ element of M is the expected number of times an agent will visit state s_j in the future, when starting in state s_i and taking action a. A reward vector $R(s')$, representing the reward generated at a state s', can then be used to directly evaluate the action-value function Q^π as follows,

$$Q^\pi(s, a) = \sum_{s'} M(s, a, s') R(s'). \quad (2)$$

SFs generalize SR with function approximation to allow modelling of environments with large state spaces [17]. The basis of SFs is that the MDP state space can be represented by a feature space, $\boldsymbol{\phi}(s) : \mathcal{S} \rightarrow \mathbb{R}^N$, and the spatial correlation from SR can be applied to those features. SFs thus capture the expected discounted future feature occupancy, and are defined as,

$$\boldsymbol{\psi}^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{\phi}(s_{t+i+1}) | s_t = s, a_t = a \right], \quad (3)$$

where the ith element of $\boldsymbol{\psi}^\pi$ represents the expected discounting of the ith feature of $\boldsymbol{\phi}$. Similar to SR, the action-value function can be approximated with a reward vector $\mathbf{w}$, specifying the reward generated by each feature, as follows,

$$Q^\pi(s, a) = \boldsymbol{\psi}^\pi(s, a)^\top \mathbf{w}. \quad (4)$$

We omit the π notation in $\boldsymbol{\psi}^\pi$ for the remainder of this paper.

The key power of SFs is that they implicitly encode a graph of the environment, as adjacent states in the MDP maintain similar encodings in $\boldsymbol{\psi}$, but the effectiveness of the graph is influenced by how one defines $\boldsymbol{\phi}$. Modern SF architectures assume, and empirically show, that state and reward prediction can generate sufficiently dense encoding for $\boldsymbol{\phi}$ to accurately capture the dynamics of the environment [3, 17, 18]. Using reward prediction does not effect SFs ability to separate dynamics and reward structure, it simply identifies important features of the environment correlated to reward (e.g., enemies or goal flags), similar to how humans and mice identify important locations or objects as a basis for navigation [31, 34].

2.2 Option Discovery

There are two primary approaches for learning a set of options, $\{\pi_1, \ldots, \pi_N\}$, that can be used in the semi-MDP framework [32]. Exploration-based methods learn options which lead to diverse states in the environment with the goal of improving exploration of extremely large state spaces during training [1, 8, 11, 14]. Two such methods are DIYAN [11], which learns diverse policies by maximizing option discernibility using a discriminator, and Deep Covering Options (DCO), which learns options that lead to poorly explored states with a Laplacian representation. However, exploration-based methods struggle when used with complex tasks in HRL because the generated options focus on high-level explorative navigation and often forgo learning aspects of low-level control.

In contrast, decomposition-based methods decompose a learned representation into a set of options which cover a diverse range of subtasks in the environment [21, 22]. We focus on decomposition-based methods because their options identify a broader range of strategic positions for option goals, while exploration methods only define options which navigate to distant states in the environment.

Decomposition-based methods are generally implemented in a two-step process. The first step learns a graph-like representation of the environment using, for example, deep graph Laplacians [21, 37], proto-value functions [19], or SFs [22]. Such representations have been shown to uncover equivalent, graph-like representations in

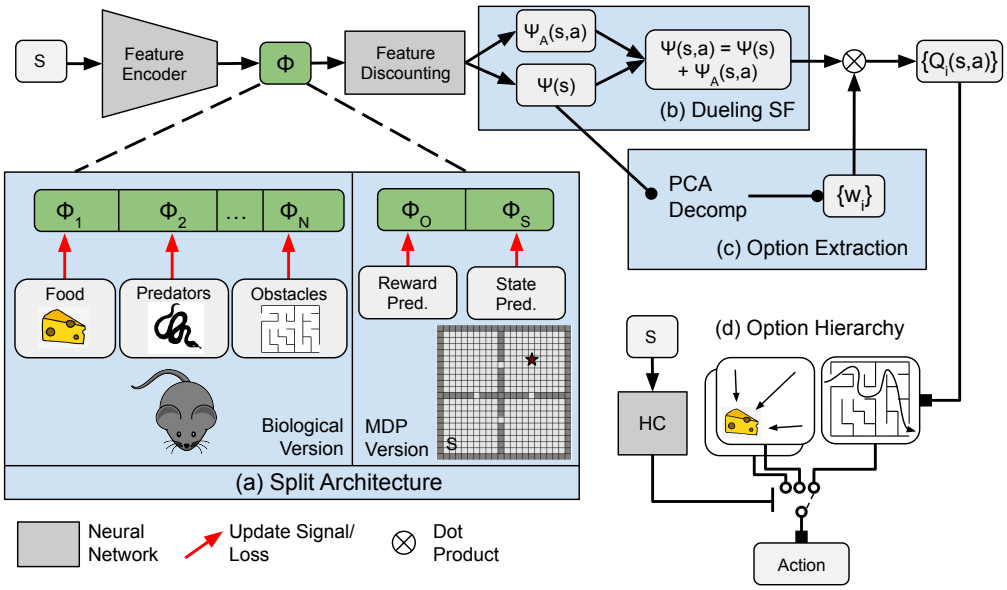

Figure 1: The SNAC decomposition and option generation pipeline. (a) A neural network feature encoder learns a split latent space, ϕ, broken into N distinct components, each encoding a unique aspect of the environment. In the generalized MDP case, we use state and reward prediction for the split. (b) The SFs, ψ, are learned with a dueling architecture defined in Equation (12) and jointly updated according to Equation (13). (c) The SF representation ψ is decomposed with PCA, creating a set of principle components that are combined with SFs to create options. (d) A generic HRL framework, where learned options are used to solve an overall task.

which spatially close states are encoded with similar representations [22]. These methods use random walk exploration to prevent learning a representation that is skewed by the exploration policy. An unbiased representation is required to ensure accurate representation of the environment's graph Laplacian, which is used for decomposition and defined by a policy-agnostic adjacency matrix [14, 19, 21, 22]. We discuss practical impacts of random walk exploration in Section 4.4. One limitation of existing methods used to learn these representations is that they focus on capturing state adjacency and struggle to model dynamic elements in the environment, such as adversaries.

The second step then decomposes the learned representation into a basis of pseudo-reward functions that are used to train options (using any RL method) [19, 21, 22]. This decomposition is typically performed using eigendecomposition methods, which produce pseudo-rewards leading to minimum cuts, or bottlenecks, in the geometric space of the environment. This approach has been shown to be useful in navigation problems, as bottleneck states can be used to bridge between sparsely connected states [17, 29]. However, options defined by eigendecomposition struggle to solve environments where a goal is not in a bottleneck [19], and thus have only been fully demonstrated in simple navigation problems [21]. Recent works have also begun to address this limitation by exploring alternative decomposition methods, such as k-means clustering, but they are still only applied to navigation problems [27].

3 OUR APPROACH

Our approach, SNAC, proposes three key ideas to improve current eigenoption methods for HRL. Our first idea improves the process of learning a graph-like representation, or topological map, of an environment by using a split SF architecture that better encodes dynamic elements through feature specialization. Our second idea improves the decomposition of a learned representation by using PCA to identify cluster-based options. Finally, our third idea improves training of a hierarchical policy by using a dueling SF architecture to learn a smoother and more accurate SF representation — this smooth representation enables us to skip the option learning phase used in other SF eigenoption methods[19, 21, 22]. Our approach is summarized in Figure 1 and our algorithm in Algorithm 1.

3.1 Bioinspired Feature Specialization through Split Successor Features

SFs provide a powerful way to represent an environment as a topological map that can be decomposed into options [22]. However, the quality of these options heavily relies on the underlying feature encoding $\boldsymbol{\phi}$, which is typically generated by state [22] or reward predictions [16, 18]. There are two main limitations imposed by this approach. First, there is no mechanism for enforcing specialization within these features, resulting in options that entangle all elements of the environment, such as spatial navigation, adversaries, and rewarding goals. Second, many agents operate in sparse reward environments where the state prediction loss dominates infrequent reward signals, leading to poor representation of rewarding elements like dynamic adversaries.

In SNAC, we instead propose splitting the encoding of $\boldsymbol{\phi}$ into *specialized* components which independently encode specific elements of the environment, such as obstacles, goal locations, or

adversaries. These specialized features are then decomposed separately to generate specialized policies relating to each element. Such specializations are present throughout the brain, from macroscale hemispheric specialization [35] to specialization in specific regions like the visual cortex, where language and spatial location are processed independently [33]. We specifically draw inspiration from the well-documented specialization in the entorhinal cortex of humans and rodents, where specialized cells encode positional information, obstacle proximity, and object location. Of particular interest are observations of object cells in [34], where dynamic cells were observed to fire in proximity to objects that were moved throughout an environment. We hypothesize that specialized features which capture a similar dynamic encoding will generate options relating specifically to dynamic objects.

We mimic such specialization in SFs by separating $\boldsymbol{\phi}$ into two components, which are learned using the two most prevalent signals available to RL agents: state prediction and reward dynamics. We expect that training via state prediction will encode a dense feature representation of the environment's spatial features, similar to place cells. More interestingly, we expect that training via reward dynamics will encode a feature representation that captures all objects an agent can interact with to generate reward, similar to object cells [34]. We implement this idea by dividing the latent $\boldsymbol{\phi}$ vector into two components,

$$\boldsymbol{\phi} = (\boldsymbol{\phi}_O, \boldsymbol{\phi}_S), \tag{5}$$

where $\boldsymbol{\phi}_O$ represents all rewarding features, which we call object features, and $\boldsymbol{\phi}_S$ represents spatial features of the environment. Object features are learned with the following reward loss,

$$L_{\text{reward},\theta_{\boldsymbol{\phi}_O}} = \|r - \boldsymbol{\phi}_O(s) \cdot \mathbf{w}\|^2, \tag{6}$$

where $\mathbf{w}$ is the SF weight vector from Equation (4). Spatial features are learned with the following state prediction loss,

$$L_{\text{spatial},\theta_{\boldsymbol{\phi}_S}} = \left\|s' - \mathcal{D}(\boldsymbol{\phi}_S(s), a)\right\|^2, \tag{7}$$

where $\mathcal{D}$ is a standard decoder trained to predict s'. The specialized feature vector $\boldsymbol{\phi}$ is then discounted to create specialized SFs,

$$\boldsymbol{\psi} = (\boldsymbol{\psi}_O, \boldsymbol{\psi}_S), \tag{8}$$

using Equation (3) with no modification.

We again note that using reward prediction to train the feature basis does not effect the ability of SFs to separate dynamics and reward structure [3, 17, 18]. The dynamics of the environment are still represented in SFs by the feature discounting $\boldsymbol{\psi}(s, a)$, with reward structure still captured by $\mathbf{w}$. This separation is independent of how we learn the feature space $\boldsymbol{\phi}$.

3.2 Decomposing Environments into a Task Subspace with PCA

Eigenoption methods have successfully solved navigation tasks in environments by generating options that lead to unique states [22]. However, options that only lead to unique states cannot easily handle dynamic environments that require complex movements, such as positioning around mobile enemies.

In SNAC , we propose replacing eigendecomposition of the environment, in this case represented through SFs, with *cluster*-based decomposition of the environment. In the context of SFs, such clustering could identify spatial clusters within the environment, such as rooms, to be used as option goals for high-level navigation. This decomposition could also identify intra-cluster relationships, which are useful for defining options to navigate within a cluster. For example, such options would allow for navigation to specific positions within a room, which is missing from current eigenoption methods. We implement this idea by decomposing our SF representation of the environment with PCA. We use PCA because it is proven to recover a basis for the cluster subspace of the input data that can identify cluster centroids and intra-cluster connections [10]. We reformulate this proof within the context of our SF decomposition problem in the following lemma using ideas from [10]:

Lemma 1: *The cluster centroid subspace of an SF topological map is spanned by the first $K-1$ principal components of PCA.*

We use PCA to independently decompose each component, $\boldsymbol{\psi}_O$ and $\boldsymbol{\psi}_S$, of our specialized SFs. This decomposition creates sets of principle components $\{\mathbf{w}_{O,i}\}$ and $\{\mathbf{w}_{S,i}\}$ that capture the specialization found in our features. We then use these principle components to generate sets of object and spatial options using Equation (14).

We note that clusters within SF topological maps could directly serve as goals for individual options, for example through k-means clustering. However, directly using such clusters for options would limit the ability of the hierarchical policy to perform micro-level movements within a cluster. We show in Section 4.3 that our PCA cluster subspace outperforms direct use of k-means clusters as options.

3.3 Improving SF Learning with Dueling SFs

SFs are traditionally learned with the following Bellman-style update [16, 22],

$$L_{\text{SF},\theta_{\boldsymbol{\psi}}} = E\left\|\boldsymbol{\phi}(s_t) + \gamma\boldsymbol{\psi}(s_{t+1}, a') - \boldsymbol{\psi}(s_t, a_t)\right\|^2, \tag{9}$$

where $a' = \text{argmax}_a(\boldsymbol{\psi}(s_{t+1}, a)^\top \mathbf{w})$. There are two issues with this update, pertaining to our SFs implementation. First, following from [22], we use a random walk exploration to learn SFs that are not biased by a policy. However, this results in a double sampling issue as a' would be randomly sampled. Second, these Bellman updates are often noisy, specifically when multiple state-action pairs have similar values, which leads to poor evaluation of the effect of actions [36].

We propose a new SFs architecture, based on the dueling architecture introduced in [36], to address both of these issues. The dueling architecture represents the action value function through two components,

$$Q(s, a) = V(s) + A(s, a), \tag{10}$$

where $V(s)$ is the state value function and $A(s, a)$ is an advantage function evaluating the value of individual actions. This parameterization allows for a powerful separation of state and action evaluation. While dueling can be implemented in the above manner, it is naive, as there is ambiguity in the value assignment: a constant could be subtracted from $A(s, a)$ and added to $V(s)$ without changing the value. Wang et al. [36] instead proposes using,

$$Q(s, a) = V(s) + (A(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s, a')), \tag{11}$$

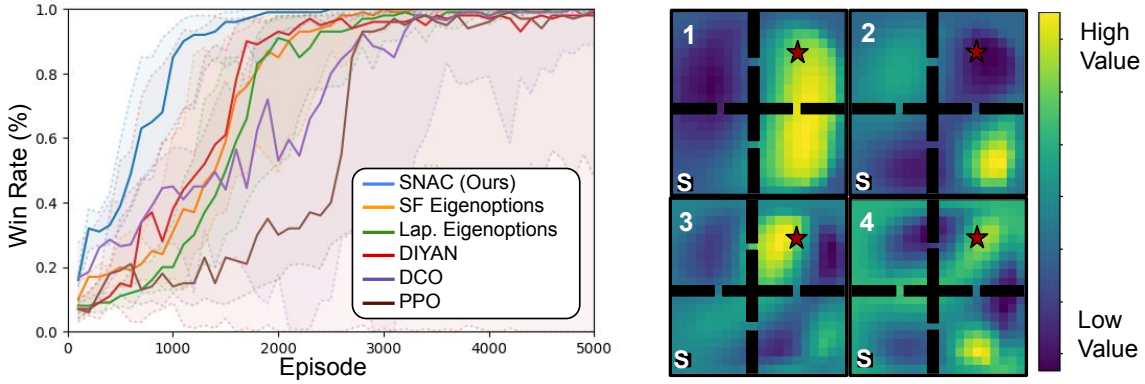

Figure 2: (left) Comparing performance during training in the 4 rooms domain. Bold lines are median runs with shaded regions representing 25-75 percentiles [25 replicates] (right) Four of 16 options generated by SNAC in 4 rooms, with the initial state shown by "S". The first principal component (1) represents high-level movement in the environment, i.e., moving to right/left side of environment. The next two options (2 & 3) resemble cluster centroids in the right hand rooms. Subsequent options (4) are complex options that enable micro-level movement, e.g., moving to the top right corner of the top right quadrant.

Algorithm 1: SNAC Algorithm

Initialize feature encoder parameterized by $\theta_{\boldsymbol{\phi}}$:
$F_{\theta_{\boldsymbol{\phi}}}(s) = \boldsymbol{\phi}(s)$;
Initialize feature discounting parameterized by θ_{SF}:
$G_{\theta_{\boldsymbol{\psi}}}(s) = \boldsymbol{\psi}(s,a)$;
Initialize HC parameterized by θ_{HC}: $H_{\theta_{HC}}(s) = \pi(s)$;
for *i in SF exploration episodes* **do**
  Explore environment with random uniform policy $\rightarrow [(s_0, a_1, s_1), ...(s_{n-1}, a_n, s_n)]$;
  Update feature encoder with Equation (6) and Equation (7) ;
  Update feature discounting with Equation (13);
end
Sample N states from environment $\rightarrow M \in R^{NxP}$;
Split M into $M_O \in R^{NxS}$ and $M_S \in R^{NxP-S}$ based on feature split;
Decompose M_O and M_S with PCA to create n options $\rightarrow$ $\{\mathbf{w}_{O,i}\}$ and $\{\mathbf{w}_{S,i}\}$;
repeat
  Reset environment $\rightarrow s_0$;
  for *i in steps* **do**
    Select option $\rightarrow j = \pi(s_i)$;
    Select action for specific option $\rightarrow a_i = argmax_a(\boldsymbol{\psi}(s_i, a)^\top \mathbf{w}_j)$;
    Step environment with $a_i \rightarrow s_i$
  end
  Update HC according to learning algorithm (PPO) ;

to fix a zero advantage reference point at the selected action.

SFs use the same Bellman-style update as Q functions, so any paramaterization or updates valid for the Q function Bellman equation, like the dueling architecture, can be applied to learn $\boldsymbol{\psi}$ [3]. Similar to [36], we paramaterize $\boldsymbol{\psi}(s,a)$ with the advantage formulation from Equation (11),

$$\boldsymbol{\psi}(s,a) = \boldsymbol{\psi}_S(s) + \boldsymbol{\psi}_A(a,s) = \boldsymbol{\psi}_S(s) + (\boldsymbol{\psi}_A(a,s) - \frac{1}{|\mathcal{A}|}\sum_{a'}\boldsymbol{\psi}_A(a',s)), \tag{12}$$

where $\boldsymbol{\psi}_S(s)$ represents the expected features induced by the state s and $\boldsymbol{\psi}_A(a,s)$ represents the expected features induced by a specific future action a. We use this dueling format of the advantage function in SNAC to improve our $\boldsymbol{\psi}(s,a)$ evaluation "in the presence of many similar-valued actions [36]." We also use this parameterization to simplify the feature discounting loss to,

$$L_{\mathrm{SF},\theta_{\boldsymbol{\psi}}} = E\,\|\boldsymbol{\phi}(s_{t+1}) + \gamma\boldsymbol{\psi}_S(s_{t+1}) - \boldsymbol{\psi}(s_t,a_t)\|^2, \tag{13}$$

which eliminates the double sampling issue as we can reparameterize the Bellman update with $\boldsymbol{\psi}_S(s_{t+1})$, eliminating the need to evaluate a'.

Our experiments show that our dueling SF architecture reduces errors in our estimation of $\boldsymbol{\psi}(s,a)$ relative to traditional SF methods. This accurate SF representation then allows us to simply define an option policy as,

$$\pi_i(s) = \underset{a}{\mathrm{argmax}}(Q_i(s,a)) = \underset{a}{\mathrm{argmax}}(\boldsymbol{\psi}(s,a)^\top \mathbf{w}_i). \tag{14}$$

This approach reduces training time and network complexity relative to existing methods which, likely due to noisy SF representations, require additional learning of π_i using $\mathbf{w}_i$ [11, 14, 19, 21, 22, 27].

4 RESULTS

We compare SNAC against five baselines, a traditional PPO RL algorithm [28], and four HRL methods. Our primary HRL baselines are decomposition-based eigenoption methods using the SF (SF Eigenoptions) [22] and deep Laplacian (Lap. Eigenoptions) [21, 37] representations. We also compare to the exploration-based option generation methods DIYAN [11] and DCO [14].

The HC in all HRL methods (Eigenoptions, DIYAN, DCO, and SNAC) is a PPO implementation which selects the options to use,

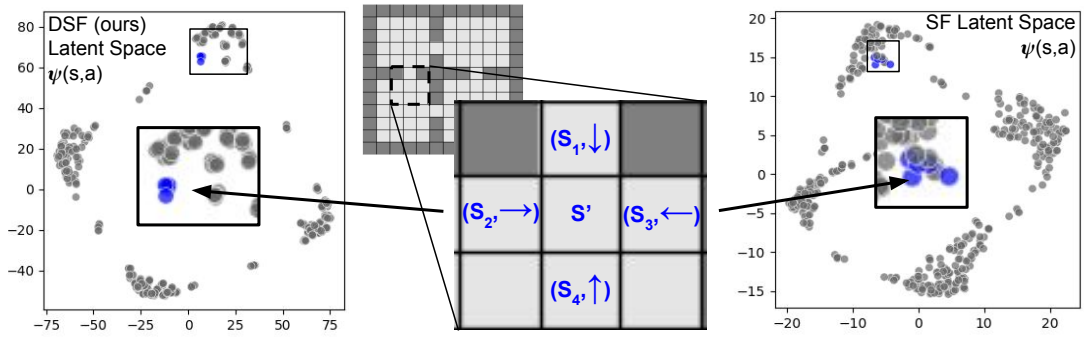

Figure 3: PCA visualization of $\psi(s,a)$ for all state-action pairs in 4 rooms. (center) state-action pairs that terminate in the same s' should have similar $\psi(s,a)$ representations, as they will have similar future trajectories. (left) *The dueling SF architecture (DSF)* produces clusters of (s,a) pairs that all terminate in the same s'. (right) *The traditional SF architecture* produces noisier (s,a) representations that often overlap with (s,a) pairs terminating in different states. Note: Each point is a linear combination, derived from PCA, of original data, and since the two sets of points come from two different learned representations, we cannot compare them based on the axes values.

instead of selecting primitive actions as in traditional RL. The HC operates at a higher temporal scale than the options, i.e. the HC selects which option to use for the next N timesteps (N). We use the same HC architecture for all methods to focus our study on the effect of option generation architectures, as opposed to the HC implementation. Hyperparameters for all methods were selected to maximize training speed with a full factorial sweep over major hyperparameters, namely learning rate, entropy, batch-size, and the number of options in hierarchical methods.

4.1 Grid-world Navigation

We first test our algorithm in the 4 rooms domain [32], a benchmark grid-world navigation problem. This experiment focuses on the effects of our PCA decomposition and SF dueling architecture, since the lack of dynamic objects limits the impact of our feature specialization approach. Here, the agent must navigate through a series of bottlenecks to reach the goal, making it a challenging domain for RL agents. Figure 2 shows that in this environment, our method, SNAC, trains faster and has significantly less variation in performance than baselines. We note that decomposition-based methods have significantly less variation in performance when compared to exploration-based methods.

We also visualize four selected options generated by SNAC for this environment in Figure 2. A clear functional hierarchy is observed based on the principal component index, i.e., k in $\mathbf{w}_k$. The first few options capture global trends in the environment, specifically left-right and top-bottom position, which can be used to navigate to high-level clusters in the environment. The middle options resemble explicit cluster centroids but with complex merging; for example, option two shows two apparent centroids in the top-left and bottom-right quadrants. The last several options have more complex structures that allow for micro-level control, such as option four, which could be used to navigate to corners in the top-left quadrant.

We also examine the effect of our dueling SF architecture on the learned SF representation. Based on Equation (3), we expect there to be two major outcomes of an SF representation: (1) the representation should maintain geometric relationships with respect to the environment's topology; and (2) (s,a) pairs with the same terminal state s' should have the same $\boldsymbol{\psi}(s,a)$. We examine the benefits of our dueling SF architecture by plotting the SF representation of all state-action pairs in Figure 3 and highlighting a set of (s,a) pairs that all terminate in the same location. We see that our dueling SF and the traditional SF both capture general geometric relationships of the domain, as each have four large clusters of points (i.e., the rooms) and four distinct points in between those clusters (i.e., the bottlenecks). However, we see that the traditional SF learning architecture does not satisfy the second expected outcome of an SF representation, since it produces a noisy clustering that mixes representations of (s,a) pairs terminating in different states. In comparison, SNAC learns a smoother representation that more precisely clusters (s,a) pairs which terminate in the same state. Our results suggest that this improved representation helps with optimizing a hierarchical policy through improved value estimation.

4.2 Adversarial Grid-world

We also evaluate our algorithm in a more complex grid-world environment with dynamic adversaries, to demonstrate the effect of our specialized split SF encoding. Here, we use an open-source implementation of a grid-world version of Capture the Flag (CtF) [24]. The grid-world map in CtF is divided into two territories with two teams of agents, red and blue, competing to capture a flag in the adversary territory. The two teams interact in a simulated combat when two opposing agents come in close proximity. The combat outcome is probabilistic and biased by the territory on which it occurs, giving an advantage to the defending team. This interaction creates a challenging complexity in the environment, as the territory significantly affects interaction outcomes and future trajectories. In our experiments, the adversary team is controlled by a heuristic policy that patrols its own territory and occasionally ventures into enemy territory to capture flags.

We focus on the CtF map shown in Figure 5, which contains a large obstacle in the middle of the map that creates two key defensive (offensive) choke points. We first analyze the sensitivity of our specialized features $\boldsymbol{\phi}$ to specific objects in the environment. The activation patterns of four features of $\boldsymbol{\phi}$ are plotted in Figure 4,

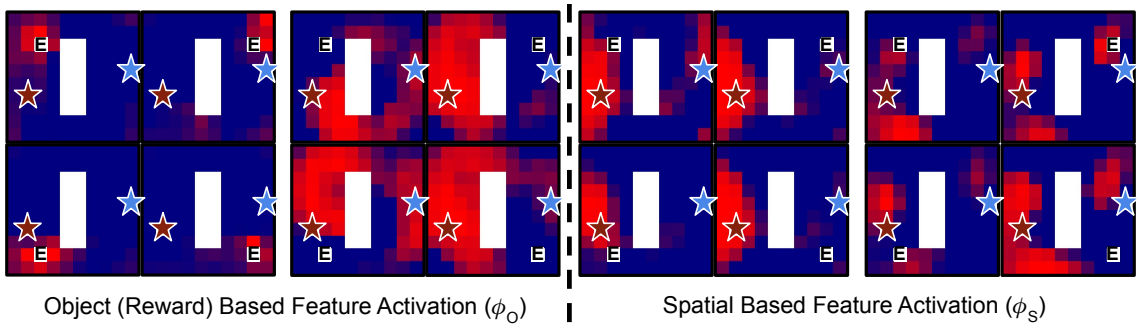

Figure 4: The ϕ representation learned by our split SF architecture in CtF. Red depicts the spatial activation of ϕ in a 1v1 game with the enemy, "E", at different positions. Several object (reward) features ϕ_O activate when proximal to enemies, similar to traces observed in the mouse entorhinal cortex in [34]. Other object features activate on a specific side of the map, since the territory impacts combat outcomes. Spatial features activate near walls and at specific spatial locations, similar to grid cells and boundary cells [4, 12, 30].

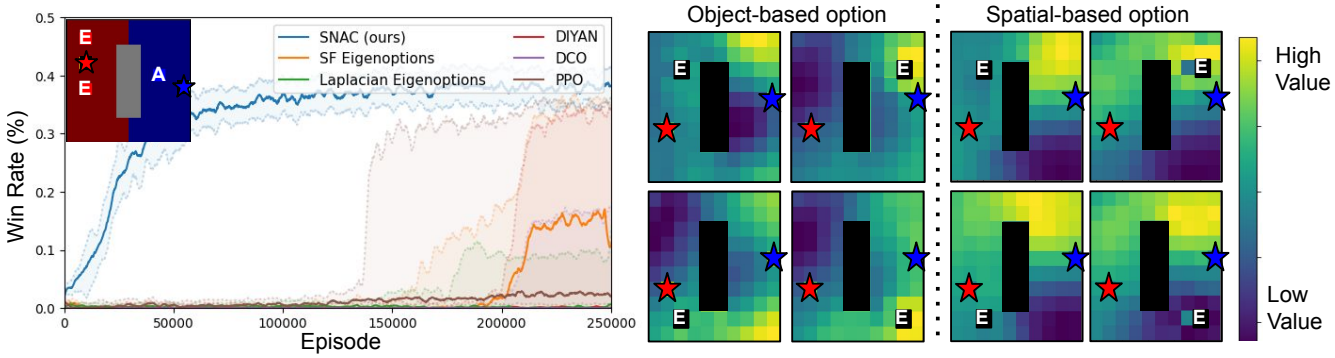

Figure 5: (left) SNAC shows improved performance relative to baselines during training in a 1v2 CtF game, where one controlled agent (A) competes against two adversarial enemies (E) to capture their flag (red star) while defending it's own (blue star). Bold line is the median, and shaded region is the 25-75 percentile [10 replicates]. (right) Two resultant options from SNAC, shown in response to one enemy agent for simplicity. The coloring represents the relative value of the position, where yellow indicates more value. The spatial-based option navigates to a specific spatial location, agnostic of the enemy position. The object-based option is highly reactive to the enemy position; this specific option appears to be defensive, as it moves to position the agent between the enemy and its own flag.

and show very different behaviors. Several object-based features, ϕ_O, show extreme sensitivity to enemy agents and only fire when in close proximity to them, similar to a mouse's sensitivity to moving objects [34]. We also see that several features are activated when on the adversary's territory, a major factor affecting rewards in CtF. Our spatial-based features, ϕ_S, behave similarly to boundary cells [30] and place cells [31], firing next to walls and at specific locations.

Figure 5 shows that SNAC significantly outperforms baselines in a 1v2 game of CtF with respect to training speed and steady-state performance. We expect these improvements are due to SNAC's ability to generate dynamic options that react to adversaries in the environment, based on specialized features as shown in Figure 4, as well as options that enable high-level and micro-level movements due to PCA decomposition. Figure 5 visualizes corresponding options generated by SNAC. We see that spatial options remain similar to those presented in Figure 2, and are constant regardless of the enemy's position. However, as expected, our object-based options show significant reactivity to dynamic attributes in the environment. For example, the option shown in Figure 5 resembles a defensive behavior, as the agent navigates to position itself between the adversary and its flag.

4.3 Adversarial Grid-world Ablation Study

We performed two ablation studies in the CtF environment to demonstrate the impact of different SNAC components. The first ablation in Figure 6 shows the performance of SNAC without PCA and without the split feature encoding. We see that both model variants perform worse than the full SNAC but still significantly outperform baselines, suggesting these two components independently improve performance and are constructively used together in SNAC. This ablation also demonstrates the value of our feature specialization, as "SNAC without PCA", which is essentially SF Eigenoptions with our split architecture, improves performance over the standard SF Eigenoptions method. The second ablation in Figure 6 shows the performance of alternative decomposition methods when using our dueling SF architecture. We observe that SNAC outperforms considered alternatives, while the dueling architecture has limited impact on the performance of SF Eigenoptions, as the 25-75th percentile ranges overlap significantly. We also see

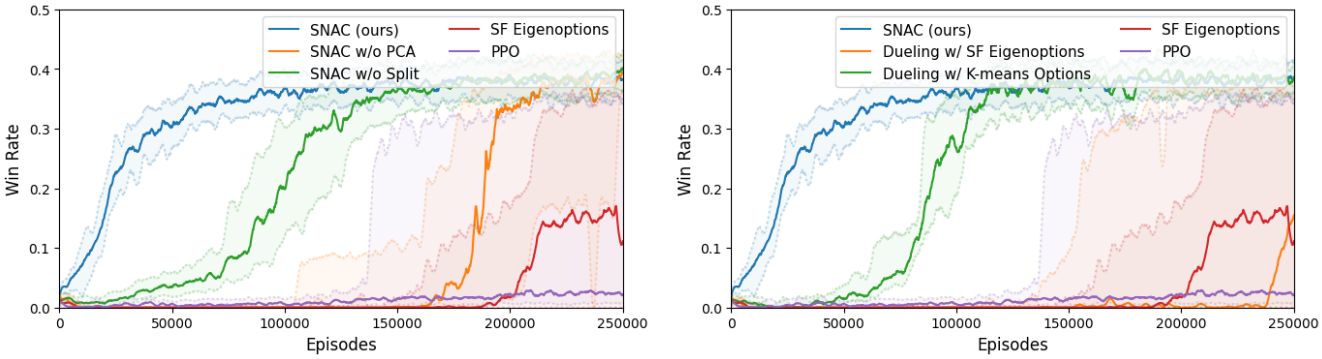

Figure 6: Ablation experiments performed in the 1v2 CtF game described in Figure 5. (left) Results from removing core components of SNAC demonstrate that its components independently improve performance over baselines. We observe that SNAC w/o PCA (i.e., SF Eigenoptions with a spit architecture) improves training speed and consistency over vanilla SF Eigenoptions, but remains slower than SNAC. Additionally, SNAC w/o Split maintains faster learning than SF Eigenoptions, showing that a PCA decomposition performs well even with reward-agnostic SFs. (right) Evaluating our SF dueling architecture with alternative decomposition techniques shows that cluster-based decompositions, like PCA and k-means, improve performance over traditional eigendecompositions but do not have the intra-cluster movement options needed to handle dynamic enemies.

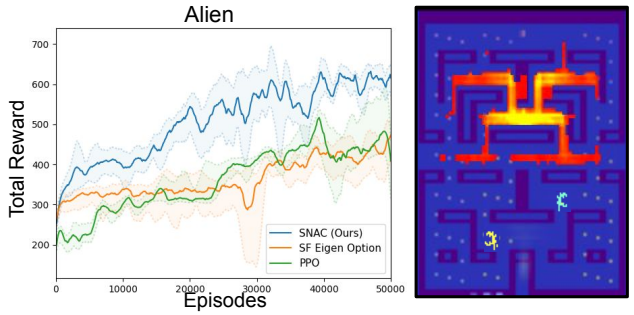

Figure 7: Comparing performance in the Atari game Alien. Both option generation methods significantly outperform traditional RL in the first 5000 episodes, after which only SNAC continues to show strong performance. However, there is a performance plateau near the end of training, likely due to limited exploration when learning the underlying SFs. State visitation during SF training is highlighted on the right and shows less than half the map is visited.

that while k-means clustering [27] is outperformed by SNAC, it does outperform other baselines, further supporting the notion that cluster-based options better handle more complex, dynamic environments than eigendecomposition.

4.4 Atari

We also tested our method in the Alien game from the open-source Atari learning environment [6], which requires complex spatial planning to win due to moving adversaries. We see in Figure 7 that there are significant early performance gains during learning for SNAC and SF Eigenoptions relative to vanilla PPO. However, only SNAC maintains higher performance further into learning, while SF Eigenoptions quickly plateaus. This plateauing is also seen in our architecture (to a lesser extent) near the end of training, and demonstrates the primary limitation of current SF methods: they perform well on tightly constrained environments, like grid-worlds, where it is reasonable to explore a majority of the states during preliminary SF training. However, in larger environments such as this one, the exploration of states distant from starting locations is limited, leading to an inaccurate SF representation and options that struggle to generalize to the rest of the environment. We highlight the states explored during SF training in Figure 7, which shows that over half of the environment is never explored during training. We expect that implementations that better explore large action spaces [13] will improve performance.

5 CONCLUSIONS

Successor features is a bioinspired framework that provides a strong approach for predicting the underlying dynamics of an environment, which can be leveraged for decomposition of a complex task into subtasks. In this work, we show that augmenting SF decomposition with feature specialization and clustering, inspired by cell specialization and clustering in the hippocampus, significantly improves learning performance and speed in benchmark spatial planning tasks. More specifically, we empirically show that the options learned by our approach, SNAC (Specialized Neurons And Clustering Architecture), outperform baselines due to high reactivity to dynamic objects such as adversaries, options that more completely cover the task subspace, and smoother learning of SFs.

Regarding future work, our specialized encoding architecture could be improved through creation of a more generalized architecture to develop splits that better mimic that of the hippocampus. Techniques could also be implemented to update and optimize learned options over long horizons, similar to continual learning in humans. However, the most pressing limitation of our approach is the exploration limitation of current SF methods due to random walk sampling, as seen in our Atari results.

ACKNOWLEDGMENTS

This work was supported in part by ONR N00014-20-1-2249, ONR N00014-19-1-2373, and ARL W911NF2020184.

REFERENCES

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. 2018. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299* (2018).

[2] Pierre Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (2017), 1726–1734. arXiv:arXiv:1609.05140v2

[3] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. 2017. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 2017-Decem. 4056–4066. arXiv:1606.05312

[4] Caswell Barry, Colin Lever, Robin Hayman, Tom Hartley, Stephen Burton, John O'Keefe, Kate Jeffery, and Neil Burgess. 2006. The boundary vector cell model of place cell firing and spatial memory. , 71–97 pages. https://doi.org/10.1515/REVNEURO.2006.17.1-2.71

[5] Timothy EJ Behrens, Timothy H Muller, James CR Whittington, Shirley Mark, Alon B Baram, Kimberly L Stachenfeld, and Zeb Kurth-Nelson. 2018. What is a cognitive map? Organizing knowledge for flexible behavior. *Neuron* 100, 2 (2018), 490–509.

[6] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[7] Andrej Bicanski and Neil Burgess. 2020. Neuronal vector coding in spatial cognition. *Nature Reviews Neuroscience* 21, 9 (2020), 453–470.

[8] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. 2020. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*. PMLR, 1317–1327.

[9] Peter Dayan. 1993. Improving Generalisation for Temporal Difference Learning: The Successor Representation. *Neural Computation* 5, 4 (1993), 613–624.

[10] Chris Ding and Xiaofeng He. 2004. K-means clustering via principal component analysis. In *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*. 225–232. https://doi.org/10.1145/1015330.1015408

[11] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2018. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070* (2018).

[12] James R. Hinman, G. William Chapman, and Michael E. Hasselmo. 2019. Neuronal representation of environmental boundaries in egocentric coordinates. *Nature Communications* 10, 1 (dec 2019), 1–8. https://doi.org/10.1038/s41467-019-10722-y

[13] Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. 2021. Learning and Planning in Complex Action Spaces. *arXiv preprint arXiv:2104.06303* (2021).

[14] Yuu Jinnai, Jee Won Park, Marlos C Machado, and George Konidaris. 2019. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*.

[15] Nicholas K Jong, Todd Hester, and Peter Stone. 2008. The utility of temporal abstraction in reinforcement learning.. In *AAMAS (1)*. Citeseer, 299–306.

[16] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems* Nips (2016), 3682–3690.

[17] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. 2016. Deep Successor Reinforcement Learning. (2016). arXiv:1606.02396

[18] Lucas Lehnert and Michael L Littman. 2020. Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning. *J. Mach. Learn. Res.* 21 (2020), 196–1.

[19] Libin Liu and Jessica Hodgins. 2017. Learning to schedule control fragments for physics-based characters using deep Q-learning. *ACM Transactions on Graphics* 36, 3 (2017). https://doi.org/10.1145/3083723

[20] Miao Liu, Marlos C. Machado, Gerald Tesauro, and Murray Campbell. 2017. The Eigenoption-Critic Framework. (2017). arXiv:1712.04065 http://arxiv.org/abs/1712.04065

[21] Marlos C Machado, Marc G Bellemare, and Michael Bowling. 2017. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2295–2304.

[22] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. 2017. Eigenoption discovery through the deep successor representation. arXiv:1710.11089

[23] I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, and S. J. Gershman. 2017. The successor representation in human reinforcement learning. *Nature Human Behaviour* 1, 9 (sep 2017), 680–692. https://doi.org/10.1038/s41562-017-0180-8

[24] N. Van Stralen, S. Kim, H. T. Tran and G. Chowdhary. 2020. Evaluating Adaptation Performance of Hierarchical Deep Reinforcement Learning. In *2020 International Conference on Robotics and Automation (ICRA)*.

[25] Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics* 36, 4 (2017). https://doi.org/10.1145/3072959.3073602

[26] Kaspar Podgorski, Derek Dunfield, and Kurt Haas. 2012. Functional Clustering Drives Encoding Improvement in a Developing Brain Network during Awake Visual Learning. *PLoS Biology* 10, 1 (jan 2012), e1001236. https://doi.org/10.1371/journal.pbio.1001236

[27] Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. 2019. Successor options: An option discovery framework for reinforcement learning. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (2019).

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (2017), 1–12. arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[29] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

[30] Trygve Solstad, Charlotte N. Boccara, Emilio Kropff, May Britt Moser, and Edvard I. Moser. 2008. Representation of geometric borders in the entorhinal cortex. *Science* 322, 5909 (dec 2008), 1865–1868. https://doi.org/10.1126/science.1166466

[31] Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. 2017. The hippocampus as a predictive map. *Nature Neuroscience* 20, 11 (oct 2017), 1643–1653. https://doi.org/10.1038/nn.4650

[32] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning.* Technical Report. 181–211 pages.

[33] Marcin Szwed, Stanislas Dehaene, Andreas Kleinschmidt, Evelyn Eger, Romain Valabrègue, Alexis Amadon, and Laurent Cohen. 2011. Specialization for written words over objects in the visual cortex. *NeuroImage* 56, 1 (may 2011), 330–344. https://doi.org/10.1016/j.neuroimage.2011.01.073

[34] Albert Tsao, May Britt Moser, and Edvard I. Moser. 2013. Traces of experience in the lateral entorhinal cortex. *Current Biology* 23, 5 (mar 2013), 399–405. https://doi.org/10.1016/j.cub.2013.01.036

[35] Danhong Wang, Randy L. Buckner, Hesheng Liu, Danhong Wang, Randy L. Buckner, and Randy L. Buckner. 2014. Functional specialization in the human brain estimated by intrinsic hemispheric interaction. *Journal of Neuroscience* 34, 37 (sep 2014), 12341–12352. https://doi.org/10.1523/JNEUROSCI.0787-14.2014

[36] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Frcitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *33rd International Conference on Machine Learning, ICML 2016*, Vol. 4. 2939–2947. arXiv:1511.06581 https://www.youtube.com/playlist?list=

[37] Yifan Wu, George Tucker, and Ofir Nachum. 2019. The Laplacian in RL: Learning Representations with Efficient Approximations. *Seventh International Conference on Learning Representations* (2019).

[38] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. 2017. Deep reinforcement learning with successor features for navigation across similar environments. In *IEEE International Conference on Intelligent Robots and Systems*, Vol. 2017-Septe. 2371–2378. https://doi.org/10.1109/IROS.2017.8206049 arXiv:1612.05533