**Reinforcement Learning Agents that Discover Structured Representations**

by

Wilka Carvalho

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2023

Doctoral Committee:

Professor Satinder Singh Baveja, Co-Chair
Associate Professor Honglak Lee, Co-Chair
Professor Joyce Chai
Professor Richard L. Lewis

Wilka Carvalho
wcarvalh@umich.edu
ORCID iD: 0000-0001-9350-9751

# DEDICATION

*To everyone that has supported me along the way. Thank you.*

*Also, to the Brain. Thank you for being such a fascinating source of curiosity.*

# ACKNOWLEDGMENTS

First, I'd like to thank my mother. My interests in reinforcement learning, cognition, and behavior were inspired by my childhood conversations with you. Afterwards, I would like to thank Alan Dion, a research professor at Stony Brook University that gave me so much time and attention. Those were truly transformative years and the beginning of my journey into being a scientist. I would also like to thank Bob Spunt and Damian Stanley at Caltech. Working with you in Ralph Adolph's group taught me that I'm interested in cognitive science and that I want to understand it through the lense of reinforcement learning. Additionally, I'd like to thank Yan Liu. After I learned that I wanted to pursue machine learning, you accepted me into your lab despite me having little to no relevant experience.

I would also like to thank my committee members. First, I'd like to thank Honglak Lee for all of your guidance and support, especially early on in my PhD. Our conversations have really shaped my views on the kinds of structured representations that we want reinforcement learning agents to have. Second, I'd like to thank Satinder Singh for accepting me into your lab and exposing me to reinforcement learning. You exposed me to an invaluable set of ideas and a focus on simplicity that I am grateful to have as I continue my career. Third, I'd like to thank Richard Lewis. Throughout my PhD, you have been a portal to connecting my machine learning research interests to cognitive science and psychology. Your guidance was invaluable in helping me form this bridge and now pursue a career as a computational cognitive scientist. Last, I'd like to thank my final committee member Joyce Chair for her helpful comments.

I would also like to thank various mentors throughout my PhD. First, I'd like to thank my DeepMind hosts Murray Shanahan, Daniel Zoran, and Danilo Rezende. Working with you was such a pleasure and transformed the journey of my PhD. I am forever grateful to all the DeepMinders I got a chance to collaborate with, talk with, and generally learn from: Andrew Lampinen, Angelos Filos, Kyriacos Nikiforou, Felix Hill, Andre Saraiva, Andre Barreto, Daved Abel, Alexander Lerchner. At Michigan, I'd like to thank Susan Gellman for providing me so much guidance and exposure to ideas in cognitive science. I'd like to thank John Laird, David Fouhey, Nikola Banovic, and Emily Provost for your support throughout my PhD.

Finally, I'd like to thank some of the amazing friends I've made in my PhD: Yves Nazon, Josie Granner, Mario and Diana Medina, Sophia Plata, Evvy and Ivan Dougherty, Eugenia Quintanilla. I learned how to think from this PhD. But I learned how to live from being your friend.

# TABLE OF CONTENTS

# LIST OF FIGURES

FIGURE

# LIST OF TABLES

TABLE

# LIST OF ACRONYMS

**RL** Reinforcement Learning

**ANN** Artificial Neural Networks

**DL** Deep Learning

**AI** Artificial Intelligence

# ABSTRACT

Deep reinforcement learning (Deep RL) has recently emerged as a powerful method for developing AI that can learn to select actions in the world. One key question in RL is how an agent should learn knowledge that can be transferred to new situations. In this dissertation, I hypothesize that one key to transferring knowledge is the ability to discover structured representations that permit relational reasoning over basic units describing the agent's experience. Recent research in computer vision and natural language processing has shown that structured neural networks with sparse and dynamic information flow enable the discovery of such structured representations, leading to faster learning and improved generalization. The thesis of this dissertation is that we can equip reinforcement learning agents with the ability to discover and exploit structured representations by incorporating structured neural networks with dynamic information flow into the core components of an RL learner. By equipping RL agents with the ability to discover structured representations, we can reduce the amount of experience the agent needs for learning and improve its ability to transfer behaviors across situations. To support this argument, I present the following evidence.

First, I incorporate structured neural networks into an RL agent's transition function and show that this enables the discovery of object representations that capture an object's category, properties, and attributes, while achieving performance comparable to an agent with access to ground-truth object information. Afterward, I incorporate structured neural networks into an RL agent's state function and demonstrate that this enables discovering object primitives that facilitate generalization across three diverse object-centric environments. Next, I incorporate structured neural networks into an agent's value function and show that this enables the discovery of features that enable generalization to combinations of tasks. Finally, I incorporate structured neural networks into an agent's policy and provide a method that transfers to new tasks with hundreds of millions fewer samples compared to other transfer learning baselines. Taken together, this thesis demonstrates that incorporating structured neural networks into the core components of an RL learner can enable structured representation learning that both reduces the amount of experience an agent requires for learning and improves its ability to transfer behaviors across situations.

# CHAPTER 1

# Introduction

Reinforcement Learning (RL) aims to develop autonomous systems that can leverage experience to learn behaviors for acting optimally in the world. One of the key challenges for RL is how an agent can gain knowledge about the world which permits transfer across situations. Once a household robot has learned to slice a fruit, say an apple, learning to slice other fruits (e.g. oranges and pears) should become easier. In this dissertation, I hypothesize that one key to transferring knowledge is the ability to discover *structured representations* that permit relational reasoning over discovered basic units that describe the agent's experience. Further, I hypothesize that defining the core components

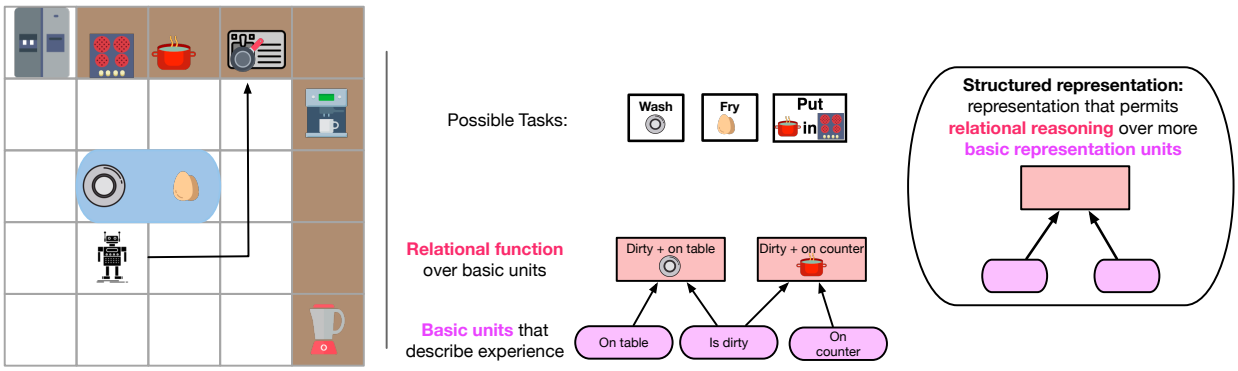


Figure 1.1: **An example of "structured representations" that an agent can discover**. On the left, we show a robot in toy kitchen environment. This robot has numerous tasks it can complete such as washing a plate, frying eggs, and putting a pot on the stove. As it completes these tasks, the robot may discover basic units that describe its experience such as an object being on the table, dirty, or on the counter. If the agent can learn relational functions over these basic units, it permits transferring knowledge across situations. For example, the function that describes the dirty plate on the table may be re-used or inform the function that describes the dirty pot on the counter.

of a reinforcement learning agent with relational functions over basic units will both lead to RL agents that learn more quickly and that can generalize their behaviors to novel situations. My hypothesis is that the ability to discover basic units which can recombined can enable faster learning and improve generalization to new settings. Returning to our example of slicing fruits, the ability to

discover structured representations could manifest as an agent discovering representations where objects are basic units. In such a setting, the agent might be able to transfer knowledge about slicing fruits from an apple to an orange by transferring knowldge from the function for relating an apple to the knife to the function for relating an orange to the knife. For example, the world model the agent learns for how the apple will evolve after being sliced by the knife may inform the world model is learns for how the orange will be sliced by the knife. So even if the agent's task is to learn to slice an orange, if it slices an apple, this may still lead to faster learning on this task.

In RL, an agent learns a behavioral policy $\pi$ for producing actions that maximizes the cumulative reward it will experience in its lifetime. In a large and complex world, individual observations commonly do not have enough information to select actions $a$ which maximize future reward $r$. Therefore, an RL agent commonly learns a subjective "state" representation $s_t = f_s(o_t, o_{t-1}, \ldots, o_1)$ which summarizes the agent's experienced observations into a representation useful for selecting actions that maximize reward. Here, $o$ refers to observations and the subscript $t$ refers to a time-step. The agent's goal is thus to

$$\text{learn} \qquad \pi(a|s) \qquad \text{that maximizes} \qquad \sum_{t=0}^{\infty} r_t. \tag{1.1}$$

In addition to state, RL defines a few key components that an agent can also learn. First, an agent can learn to estimate the sum of future rewards it will experience from following a particular behavioral policy $\pi$ and starting at a state $s$. This is the RL definition of the "value" of that state, $V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $\gamma$ is a discount factor which prioritizes immediate rewards for defining the value of the current state. The agent can then use this value function to select actions which lead to states with higher value. In addition to the value function, an agent can learn a transition function $T$ which predicts a future state and reward given an initial state and action, $(\hat{s}_{t+1}, \hat{r}_{t+1}) = T(s_t, a_t)$.

Structured representations can aid each of these component in an RL agent. Structured representations can enable a state function that is comprised of basic units which can be recombined (for example objects or other aspects of the environment the agent commonly experiences). The agent's transition function can be defined with relational functions that permit re-using parameters across components. Likewise, the agent's value function can support recombining more basic component-wise value functions. For example, the value of slicing an apple and an orange could be expressed as sum of the values for slicing the individual objects. I emphasize that the structure that an agent can discover may not necessarily be object-centric. However, objects provide familiar and intuitive examples for thinking about how such structured representations may be helpful for RL agents.

So how do we equip RL agents with the ability to discover structured representations? Recently,

the field of "Deep Reinforcement Learing" or "Deep RL" has emerged which equips RL agents with the ability to learn representations from their experience. Key to Deep RL is the use of large, multi-layer artificial neural networks or "Deep Learning" for function approximation. With the ability of RL to converge in the limit to an optimal behavior policy (Watkins and Dayan, 1992; Jaakkola et al., 1993; Singh et al., 2000) and the ability of neural networks to approximate any function (Hornik et al., 1989; Funahashi, 1989; Lu and Lu, 2020), this has enabled a powerful class of AI systems that can learn representations and behaviors for very complex tasks from experience. For example, RL agents have surpassed human-level performance in Atari video games (Mnih et al., 2015), the game of Go (Schrittwieser et al., 2020), and an RL system has enabled autonomous control over a nuclear fusion reactor (Degrave et al., 2022). However, traditional neural networks are limited in their ability to do set-based relational reasoning. This is because each "layer" typically produces a single vector output. If they do any set-based relational reasoning over dynamically chosen subsets of their representations, they do so completely implicitly. In contrast, more recent neural networks have layers that produce a set of vector outputs (He et al., 2020; Vaswani et al., 2017). Subsets of these vectors can then be dynamically re-combined by downstream functions.

Recently, in Natural Language Processing (NLP) and Computer Vision (CV), structured neural networks with sparse and dynamic information flow have rise to prominence. Researchers have shown that neural networks which dynamically flow information between sparsely-connected and repetitive internal structures can facilitate the discovery of structured representations that enable faster learning and improved generalization. In CV, "ResNets" that employ residual blocks with residual connections are known to discover object parts and shapes while additionally improving generalization across visual settings (He et al., 2020; Huang et al., 2020a; Li et al., 2020). In NLP, "Transformers" use attention "heads" that dynamically flow information and have led to impressive generalization results (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2018). Interestingly, researchers have found that attention heads can discover functions for attending to adjacent words and for tracking specific syntactic relations (Voita et al., 2019).

**Thesis**

We can equip reinforcement learning agents with the ability to discover structured representations by incorporating structured neural networks with dynamic information flow into the core components of an RL learner. Equipping RL agents with the ability to discover structured representations can reduce the amount of experience the agent needs for learning and improve its ability to transfer behaviors across situations.

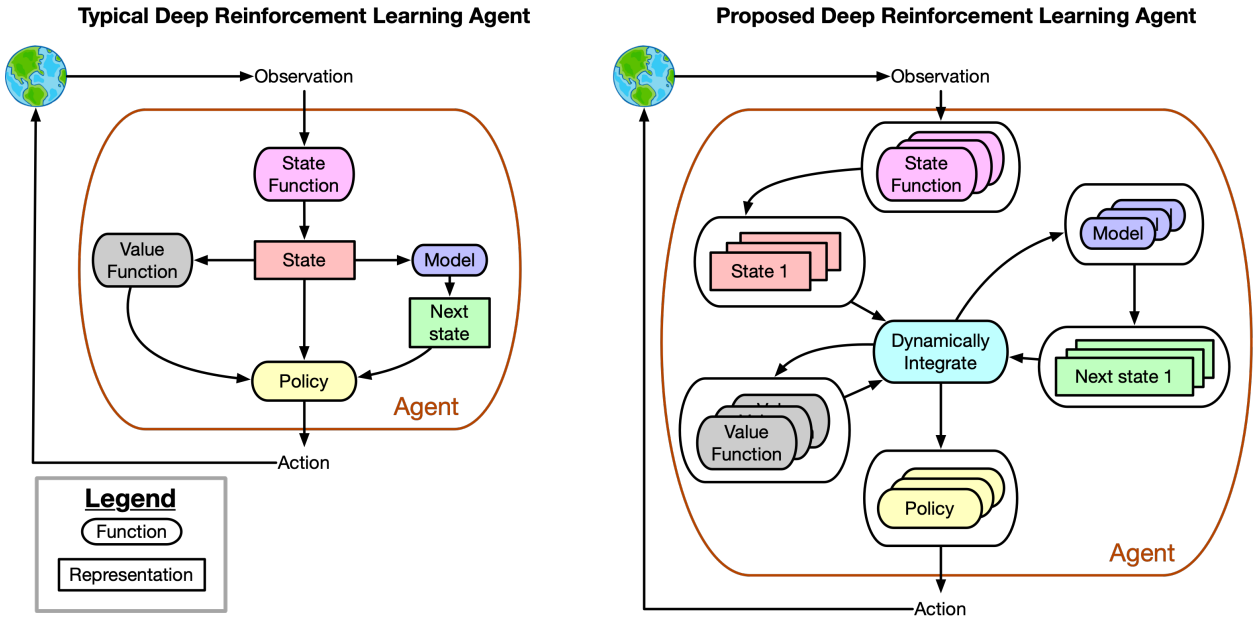


Figure 1.2: **Left**: a typical Deep RL agent which fixed information flow. **Right**: a schematic of future Deep RL agent which incorporate structured neural networks with dynamic information into their core components. I hypothesize this will enable an RL agent to discover and exploit structured representations for faster learning and improved generalization.

## 1.1 Contributions

Below, I summarize how each chapter of this dissertation contributes to my thesis that structured neural networks enable the discovery of structured representations that reduce learning time and improve generalization. In §1.1.1, I study structured neural networks within the context of learning a structured transition function, in §1.1.2 within the context of learning structured state function, in §1.1.3 within the context of learning structured value function, and in §1.1.4 within the context of learning structured policy.

### 1.1.1 Reinforcement Learning for Sparse-Reward Object-Interaction Tasks in First-person Simulated 3D Environments

In this chapter, we consider the problem of learning sparse-reward object-interaction tasks in a photorealistic 3D environment. Learning sparse-reward object-interaction tasks in a 3D environment efficiently requires that an agent learn environment abstractions which capture object-pertinent information such as an object's category (is it a toaster?), its properties (is it on?), and object-relations (is something inside of it?) from a sparse-reward signal. Object-interaction tasks are particularly challenging because the size of the state-space grows exponentially with the number

of objects, object-states, and object-relationships that exist. In order to study this problem, we adopt the virtual home-environment AI2Thor (Kolve et al., 2017) because no prior work had yet learned sparse-reward object-interaction tasks without expert demonstrations or shaped rewards in this domain.

We propose the *Learning Object Attention & Dynamics* (or *LOAD*) agent. LOAD is composed of a base object-centric relational policy that leverages inter-object attention to incorporate object-relationships when estimating object-interaction action-values. Without ground-truth information to identify object categories, properties, or relationships, LOAD discovers useful object-representations by learning a structured object-centric transition model in an (initially random) embedding space. LOAD learns to structure the space by treating the learning problem as a classification problem, where random embeddings are treated as incorrect labels. We compare our method to other methods for learning object representations. We find that our method best closes the performance gap with an oracle that has access to ground-truth object-information. We also perform a quantitative analysis indicating that our method best learns representations that capture the ground-truth object information. This work was published in IJCAI, 2021 (Carvalho et al., 2021b).

### 1.1.2 Feature-Attending Recurrent Modules for Generalization in Reinforcement Learning

While LOAD enabled discovering object-centric abstractions that supported faster learning for sparse-reward tasks, it relied on access to an "objectness"-detector. In this chapter, we remove this assumption and study weak inductive biases for discovering abstractions that support generalization in object-centric environments. We study three diverse environments: (1) one, where an agent must remember object motions, (2) a 3D pick and place environment, and (3) a navigation environment where the agent must navigate around object-induced obstacles.

We propose Feature Attending Recurrent Modules (FARM), a structured neural network architecture for representing state. FARM learns a state representation that is distributed across multiple modules that each attend to spatiotemporal features with an expressive feature attention mechanism. This enables FARM to represent diverse object-induced spatial and temporal regularities across subsets of modules. We compare FARM against other methods with weak inductive biases for enabling objects to emerge in a state representation. We find that FARM enables generalizing (a) memory to longer combinations of object motions; (b) navigation to 3D objects in larger environments; and (c) memory of goal information to longer tasks with more distractors. This work was published in the ICML Workshop on Unsupervised RL, 2021 (Carvalho et al., 2021a).

### 1.1.3 Composing Task Knowledge With Modular Successor Feature Approximators

FARM was a structured neural network for discovering abstractions in an agent's state representation. In this chapter, we develop a structured neural network for the agent's value function. Specifically, we investigate whether the abstractions learned by FARM can form prediction targets for the generalization "successor features" framework. Successor features are a type of value function that estimate how much features known as "cumulants" will be experienced by an agent when following a particular behavioral policy. One key challenge to using the successor features framework for generalization is that it commonly requires carefully designed cumulants and their discovery is currently an open problem. Here, we hypothesize that a simple inductive bias for learning structured cumulants can enable an agent to learn abstractions that capture regularities across multiple tasks.

We present Modular Successor Feature Approximators (MSFA) and we show that we can improve reward-driven discovery of cumulants through an architectural bias alone. MSFA learns a set of prediction modules that are responsible for discovering state features (i.e. cumulants) and predictions (i.e. successor features) for different aspects of the environment. We compare against using an unstructured architecture for discovering cumulants and against using a reward-based value function architecture. We first show that MSFA is able to recover the generalization results of prior work that has leveraged oracle cumulants. Afterwards, we show that MSFA enables cumulant discovery and faster learning on a more complex suite of longer horizon entity-interaction tasks. This work was published in ICLR, 2023 (Carvalho et al., 2023a).

### 1.1.4 Discovering Representations for Transfer with Successor Features and the Deep Option Keyboard

MSFA was able to discover cumulants that enabled transfer; however, MSFA relies on hand-designed task encodings and we only showed generalization within 2D gridworld environments. A more general solution would discover all features necessary and support tasks in 3D environments. Discovering all features necessary for transfer with successor features is challenging for a number of reasons. First, tasks in 3D environments typically have long-horizons and sparse-rewards. This makes it challenging to discover both cumulants and task encodings from the reward signal. The horizon of tasks can then make learning successor features challenging because of high-variance in the return estimates. These challenges can compound with additionally learning successor features over a non-stationary return-target (the changing cumulants).

To address the issues, we present two novel methods: the Deep Option Keyboard (Deep OK) and Categorical Successor Feature Approximators (CSFA). CSFA is an algorithm for jointly estimating

successor features while learning the cumulants and task encodings that support them. CSFA addresses challenges with estimating a non-stationary return by estimating SFs with a variant of the categorical two-hot representation introduced by MuZero (Schrittwieser et al., 2020). CSFA discretizes the space of cumulant-return values into bins and learns a probability mass function (pmf) over them. Modelling cumulant-returns with a pmf is more robust to outliers and can better accomodate a shifting magnitude. At transfer time, we leverage Deep OK to adaptively combine subsets of known behaviors using the machinery of successor features. With Deep OK and CSFA, we achieve the first demonstration of transfer with SFs in a challenging 3D environment where all the necessary representations are discovered. We first compare CSFA against other methods for approximating SFs and show that only CSFA discovers representations compatible that support the SF framework at this scale. We then compare Deep OK against transfer learning baselines and show that it transfers most quickly to long-horizon tasks. This work is under submission (Carvalho et al., 2023b).

# CHAPTER 2

# Reinforcement Learning for Sparse-Reward Object-Interaction Tasks in First-person Simulated 3D Environments

First-person object-interaction tasks in high-fidelity, 3D, simulated environments such as the AI2Thor virtual home-environment pose significant sample-efficiency challenges for reinforcement learning (RL) agents learning from sparse task rewards. To alleviate these challenges, prior work has provided extensive supervision via a combination of reward-shaping, ground-truth object-information, and expert demonstrations. In this work, we show that one can learn object-interaction tasks from scratch without supervision by learning an attentive object-model as an auxiliary task during task learning with an object-centric relational RL agent. Our key insight is that learning an object-model that incorporates object-attention into forward prediction provides a dense learning signal for unsupervised representation learning of both objects and their relationships. This, in turn, enables faster policy learning for an object-centric relational RL agent. We demonstrate our agent by introducing a set of challenging object-interaction tasks in the AI2Thor environment where learning with our attentive object-model is key to strong performance. Specifically, we compare our agent and relational RL agents with alternative auxiliary tasks to a relational RL agent equipped with ground-truth object-information, and show that learning with our object-model best closes the performance gap in terms of both learning speed and maximum success rate. Additionally, we find that incorporating object-attention into an object-model's forward predictions is key to learning representations which capture object-category and object-state.

## 2.1 Introduction

Consider a robotic home-aid agent that learns *object-interaction tasks* that involve using multiple objects together to accomplish various tasks such as chopping vegetables or heating meals. Such tasks are important for artificial intelligence (AI) to make progress on because of their large potential

to impact our everday world: nursing robots can serve healthcare workers in hospitals, and home-aid robots can help busy families, the disabled, and the elderly.

Prior work on object-interaction tasks has focused on achieving strong training performance using expert demonstrations (Zhu et al., 2017; Shridhar et al., 2019). Unfortunately, Zhu et al. (2017) found they were unable to learn relatively simple pick and place tasks when only learning from a sparse task-completion signal. Other work has relaxed the learning problem by relying on domain knowledge in the form of shaped rewards or object-affordance knowledge (Jain et al., 2019; Gordon et al., 2018).

Unfortunately, expert demonstrations and shaped rewards can be challenging to obtain for tasks novel to an agent. Additionally, it can be tedious or impossible to obtain ground-truth information about all novel objects an agent may encounter. Ideally, agents are capable of learning object-interaction tasks without this information. To work towards this, we focus on the setting where none of these are available.

Learning object-interaction tasks without expert demonstrations or shaped rewards is challenging because selecting between object-interactions induces a branching factor that scales with the number of visible objects, leading the agent to choose from 50-100 possible actions at a given time-step. This leads the agent to infrequently experience a successful episode. When the agent does, task completion typically occurs after many hundred time-steps. Consider learning to toast bread. The agent should learn to turn on the toaster after a bread slice is placed inside, i.e. it needs to learn to represent *containment relationships* (the bread is inside the toaster) and *object properties* (the toaster is on or off). Without domain knowledge about objects, task-completion alone provides a weak learning signal for learning both to represent 3D object categories, properties, and relationships. When episodes last for hundreds of time-steps and the agent interacts with many objects, this makes it challenging to learn about about how the agent's object-interactions led to reward.

In this work, we find that we can achieve strong training performance on object-interaction tasks without expert demonstrations, shaped rewards, or ground-truth object-knowledge by incorporating inter-object attention and an object-centric model into a reinforcement learning agent.We call our agent the *Learning Object Attention & Dynamics* (or *LOAD*) agent. LOAD is composed of a base object-centric relational policy (*Attentive Object-DQN*, §2.4.1) that leverages inter-object attention to incorporate object-relationships when estimating object-interaction action-values. Without ground-truth information to identify object categories, properties, or relationships, LOAD learns object-representations with a novel learning objective that frames learning an object-model as a classification problem, where random object-embeddings are incorrect labels (*Attentive Object-Model*, §2.4.2). By doing so, we provide the object-model with a dense learning signal for learning to represent both object categories and changes in object-properties caused by different object-interactions. Additionally, by sharing inter-object attention between the policy and the model,

learning the model helps drive learning of inter-object attention helpful for speedening task learning.

In order to study object-interaction tasks and evaluate our agent, we adopt the virtual home-environment AI2Thor (Kolve et al., 2017) (or *Thor*). Thor is an open-source environment that is high-fidelity, 3D, partially observable, and enables object-interactions. We show that LOAD is able to significantly reduce sample complexity in this domain where no prior work has yet learned sparse-reward object-interaction tasks without expert demonstrations or shaped rewards.

In our main evaluation, we compare pairing Attentive Object-DQN with our Attentive Object-Model to alternative representation learning methods, and show that learning with our object-model best closes the performance gap to an agent supplied with ground-truth information about object categories, properties, and relationships (§2.5.1). Through an analysis of the learned object-representations and inter-object attention learned by each auxiliary task, we provide quantitative evidence that our Attentive Object-Model best learns representations that capture the ground-truth information present in our oracle (§2.5.2). We hypothesize that this is the source of our strong performance. Afterwards, we perform a series of ablations to study the importance of object-representations which capture object-properites and object-relations for reducing sample-complexity (§2.5.3).

In summary, the key contributions of this work are: (1) LOAD: an RL agent that demonstrates how to learn sparse-reward object-interaction tasks with first-persosn vision without expert demonstrations, shaped rewards, or ground-truth object-knowledge. (2) A novel Attentive Object-Model auxilliary task, which frames learning an object-model as a classification problem. With our analysis, we provide evidence that for our 3D, high-fidelty domain and our architecture, it is key to learn object-representations which not only capture object-categories but also object-properties and object-relations.

## 2.2 Related Work

**Learning object-interaction tasks in 3D, first-person environments**. Due to the large branching factor induced by object-interactions, most work here has relied extensively on expert demonstrations (Zhu et al., 2017; Shridhar et al., 2019; Xu et al., 2019) or avoided this problem by hard-coding object-selection (Jain et al., 2019; Gordon et al., 2018). The work most closely related to ours is Oh et al. (2017a) (in Minecraft) and Zhu et al. (2017) (in Thor). Both develop a hierarchical reinforcement learning agent where a meta-controller provides goal object-interactions for a low-level controller to complete using ground-truth object-information. Both provide agents with knowledge of all objects and both assume lower-level policies pretrained to navigate to objects and to select interactions with a desired object. In contrast, we do not provide the agent with any ground-truth object information; nor do we pretrain navigation to objects or selection of them.

**Object-Centric Relational RL**. An intutive approach to tasks with objects is object-centric relational RL. Most work here has used hand-designed representations of objects and their relations, showing things like improved sample-efficiency (Xu et al., 2020), improved policy quality (Zaragoza et al., 2010), and generalization to unseen objects (Van Hoof et al., 2015). In contrast, we seek to learn object-representations and object-relations implicitly with our network. Most similar to our work is Zambaldi et al. (2018)–which applies attention to the feature vector outputs of a CNN. In this work, Attentive Object-DQN is a novel architecture extension for a setting with an object-centric observation- and action-space. Additionally, we show that learning an object-model as an auxilliary task can help drive learning of attention.

**Learning an object-model as an auxiliary task**. Most prior work here has focused on how an object-model can be used in model-based reinforcement learning by enabling superior planning (Ye et al., 2020; Veerapaneni et al., 2020; Watters et al., 2019). In contrast, we do not use our object-model for planning and instead show that it can be leveraged to learn object-representation and inter-object attention to support faster policy learning in a model-free setting. Additionally, other work focused on domains where representation-learning only had to differntiate object-categories. We show that our method can additionally differentiate object-properties and does so significantly better than the object-model of Watters et al. (2019). Our attentive object-model is most similar to the Contrastive Structured World Model (CSWM) (Kipf et al., 2019), which uses a maximum margin contrastive learning objective (Hadsell et al., 2006) to learn an object-model. Instead, we formulate a novel object-model contrastive objective as learning a classification problem. We note that they applied their model towards video-prediction and not reinforcement learning.

## 2.3 Sparse-Reward Object-Interaction Tasks in a First-Person Simulated 3D environment

**Observations**. We focus on an agent that has a 2D camera for experiencing *egocentric* observations $x^{\texttt{ego}}$ of the environment. Our agent also has a pretrained vision system that enables it to extract bounding box image-patches corresponding to the visible objects in its observation $X^o = \{x^{o,i}\}$. Besides boxes around objects, no other information is extracted (i.e. no labels, identifiers, poses, etc.). We assume the agent has access to its $(x, y, z)$ location and body rotation $(\varphi_1, \varphi_2, \varphi_3)$ in a global coordinate frame, $x^{\texttt{loc}} = (x, y, z, \varphi_1, \varphi_2, \varphi_3)$.

**Actions**. In this work, we focus on the Thor environment. Here, the agent has $8$ base object-interactions: $\mathcal{I} = \{$*Pickup, Put, Open, Close, Turn on, Turn off, Slice, Fill*$\}$. The agent interacts with objects by selecting (object-image-patch, interaction) pairs $a = (b, x^{o,c}) \in \mathcal{I} \times X^o$, where $x^{o,c}$ corresponds to the *chosen* image-patch. For example, the agent can turn on the stove by selecting the

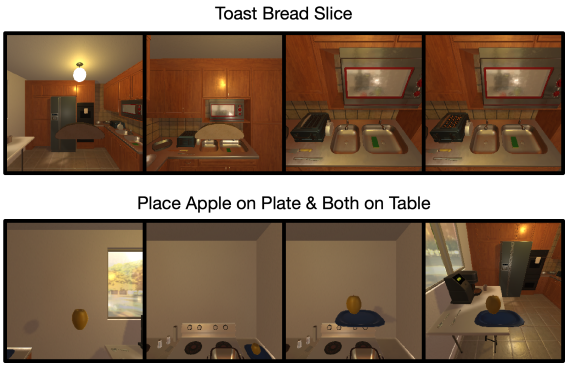


Figure 2.1: We present the steps required to complete two of our tasks. In "Toast Bread Slice", an agent must pickup a bread slice, bring it to the toaster, place it in the toaster, and turn the toaster on. In order to complete the task, the agent needs to recognize the toaster across angles, and it needs to recognize that when the bread is inside the toaster, turning the toaster on will cook the bread. In "Place Apple on Plate & Both on Table", agent must pickup an apple, place it on a plate, and move the plate to a table. It must recognize that because the objects are combined, moving the plate to the table will also move the apple. We observe that learning to use objects together such as in the tasks above poses a representation learning challenge – and thus policy learning challenge – when learning from only a task-completion reward.

| Slice $\{X_i\}, n \in [1, 3]$ | Make Tomato & Lettuce Salad | Place Apple on Plate, Both on Table | Cook Potato on Stove | Fill Cup with Water | Toast Bread Slice |
|---|---|---|---|---|---|
| (A) recognize knife across angles<br>(B) recognize 2-4 objects | (B) recognize 3 objects<br>(C) use containment: plate with tomato/lettuce slice | (B) recognize 3 objects<br>(C) use containment: apple on plate | (B) recognize 2 objects<br>(C) use containment: potato on stove<br>(D) changing properites: cooked potato | (A) recognize translucent cup across backgrounds<br>(B) recognize 2 objects<br>(C) use containment: cup in sink<br>(D) changing properites: filled cup | (A) recognize toaster across angles<br>(B) recognize 2 objects<br>(C) use containment: bread inside toaster<br>(D) changing properites: cooked bread |

Table 2.1: Description of challenges associated with the tasks we study. See Figure 2.1 for example panels of 2 tasks.

image-patch containing the stove-knob and the *Turn on* interaction (see Figure 2.2 for a diagram). Each action is available at every time-step and can be applied to all objects (i.e. no affordance information is given/used). Interactions occur over one time-step, though their effect may occur over multiple. For the example above, when the agent applies "Turn on" to the stove knob, food on the stove will take several time-steps to heat.

In addition to object-interactions, the agent can select from 8 base navigation actions: $\mathcal{A}_N =$

{*Move ahead, Move back, Move right, Move left, Look up, Look down, Rotate right, Rotate left*}. With {*Look up, Look down*}, the agent can rotate its head up or down in increments of $30°$ between angles $\{0°, \pm 30°, \pm 60°\}$. $0°$ represents looking straight ahead. With {*Rotate Left, Rotate Right*}, the agent can rotate its body by $\{\pm 90^o\}$.

**Tasks**. We construct 8 tasks with the following 4 challenges. Challenge (A): the visual complexity of task objects (e.g. the cup is translucent). Challenge (B): the number of objects to be interacted with (e.g., "Slice Apple, Potato, Lettuce" requires the agent interact with 4 objects). Challenge (C): whether object-containment must be recognized and used (e.g. toasting bread in a toaster). Challenge (D): whether object-properties change (e.g. bread gets cooked). See Figure 2.1 for a description of the challenges associated with each task and Figure 2.1 for example panels of 2 tasks.

**Reward**. We consider a single-task setting where the agent receives a terminal reward of 1 upon task-completion.

## 2.4 LOAD: Learning Object Attention & Dynamics Agent

LOAD is a reinforcement learning agent composed of an object-centric relational policy, Attentive Object-DQN, and an Attentive Object-Model. LOAD uses 2 perceptual modules. The first, $f^o_{\texttt{enc}}$, takes in an observation $x$ and produces object-encodings $\{\boldsymbol{z}^{o,i}\}_{i=1}^n$ for the $n$ visible object-image-patches $X^o = \{x^{o,i}\}_{i=1}^n$, where $\boldsymbol{z}^{o,i} \in \mathbb{R}^d_o$. The second, $f^\kappa_{\texttt{enc}}$, takes in the egocentric observation and location $x^\kappa = [x^{\texttt{ego}}, x^{\texttt{loc}}]$ to produce the *context* for the objects $\boldsymbol{z}^\kappa \in \mathbb{R}^d_\kappa$. LOAD treats state as the union of these variables: $s = \{\boldsymbol{z}^{o,i}\} \cup \{\boldsymbol{z}^\kappa\}$. Given object encodings, Attentive Object-DQN computes action-values $Q(s, a = (b, x^{o,i}))$ for interacting with an object $x^{o,i}$ and leverages an attention module $A$ to incorporate information about other objects $x^{o,j \neq i}$ into this computation (see §2.4.1).

To address the representation learning challenge induced by a sparse-reward signal, object-representations $\boldsymbol{z}^{o,i}$ and object-attention $A$ are trained to predict object-dynamics with an attentive object-model (see §2.4.2). See Figure 2.2 for an overview of the full architecture.

### 2.4.1 Attentive Object-DQN

Attentive Object-DQN uses $\widehat{Q}(s, a)$ to estimate the action-value function $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_t | S_t = s, A_t = a]$, which maps state-action pairs to the expected return on starting from that state-action pair and following policy $\pi$ thereafter.

**Leveraging inter-object attention during action-value estimation**. In many tasks, an agent must integrate information about multiple objects when estimating $Q$-values. For example, in the "toast bread" task, the agent needs to integrate information about the toaster and the bread when

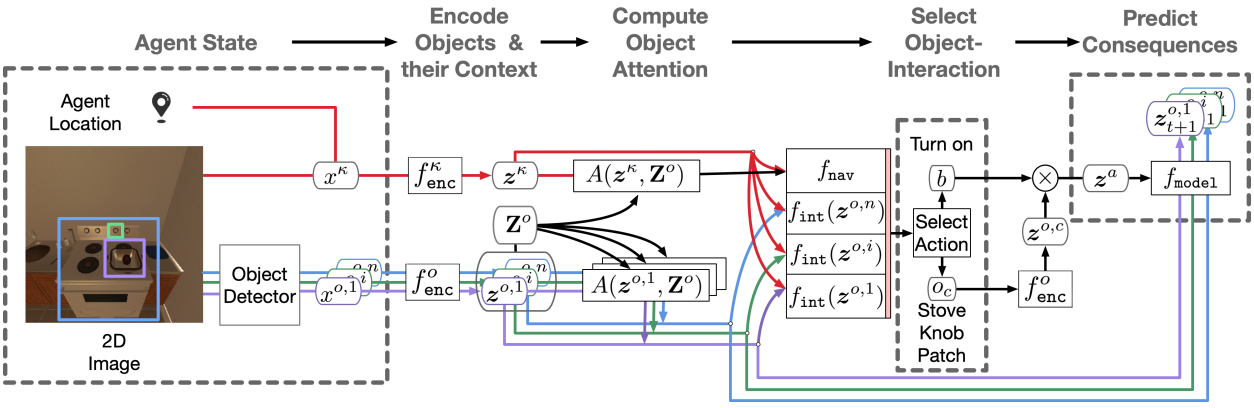


Figure 2.2: Full architecture and processing pipeline of LOAD. A scene is broken down into object-image-patches $\{x^{o,j}\}$ (e.g. of a pot, potato, and stove knob). The scene image is combined with the agent's location to define the *context* of the objects, $x^\kappa$. The objects $\{x^{o,j}\}$ and their context $x^\kappa$ are processed by different encoding branches and then recombined by an attention module $A$ that selects relevant objects for computing Q-value estimates. Here, $A$ might select the pot image-patch when computing $Q$-values for interacting with the stove-knob image-patch. Actions are selected as (object-image-patch, base action) pairs $a = (b, x^{o,c})$. The agent then predicts the consequences of its interactions with our attentive object-model $f_{\texttt{model}}$ which reuses $A$.

deciding to turn on the toaster. To accomplish this, we exploit the object-centric observations-space and employ attention (Vaswani et al., 2017) to incorporate inter-object attention into $Q$-value estimation.

More formally, given an object-encoding $\boldsymbol{z}^{o,i}$, we can use attention to select relevant objects $A(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^o) \in \mathbb{R}^{d_o}$ for estimating $Q(s, a = (b, x^{o,i}))$. With a matrix of object-encodings, $\boldsymbol{Z}^o = [\boldsymbol{z}^{o,i}]_i \in \mathbb{R}^{n \times d_o}$, we can perform this computation efficiently for each object-image-patch via:

$$\begin{pmatrix} A(\boldsymbol{z}^{o,1}, \boldsymbol{Z}^o) \\ \vdots \\ A(\boldsymbol{z}^{o,n}, \boldsymbol{Z}^o) \end{pmatrix} = \text{Softmax}\left(\frac{\left(\boldsymbol{Z}^o W^{q_o}\right)\left(\boldsymbol{Z}^o W^k\right)^\top}{\sqrt{d_k}}\right)\boldsymbol{Z}^o. \tag{2.1}$$

Here, $\boldsymbol{Z}^o W^{q_o}$ projects each object-encoding to a "query" space and $\boldsymbol{Z}^o W^k$ projects each encoding to a "key" space, where their dot-product determines whether a key is selected for a query. The softmax acts as a soft selection-mechanism for selecting an object-encoding in $\boldsymbol{Z}^o$.

**Estimating action-values**. We can incorporate attention to estimate $Q$-values for selecting an interaction $b \in \mathcal{I}$ on an object $x^{o,i}$ as follows:

$$\widehat{Q}(s, a = (b, x^{o,i})) = f_{\texttt{int}}([\boldsymbol{z}^{o,i}, A(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^o), \boldsymbol{z}^\kappa]) \tag{2.2}$$

Importantly, this enables us to compute $Q$-values for a variable number of *unlabeled* objects. We

can similarly incorporate attention to compute $Q$-values for navigation actions by replacing $\boldsymbol{Z}^o W^{q_o}$ with $(W^{q_\kappa}\boldsymbol{z}^\kappa)^\top$ in equation 2.1. We estimate $Q$-values for navigation actions $b \in \mathcal{A}_N$ as follows:

$$\widehat{Q}(s, a = b) = f_{\texttt{nav}}([\boldsymbol{z}^\kappa, A(\boldsymbol{z}^\kappa, \boldsymbol{Z}^o)]) \tag{2.3}$$

**Learning**. We estimate $\widehat{Q}(s, a)$ as a Deep Q-Network (DQN) by minimizing the following temporal difference objective:

$$\mathcal{L}_{\texttt{DQN}} = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}} \left[ ||y_t - \widehat{Q}(s_t, a_t; \theta)||^2 \right], \tag{2.4}$$

where $y_t = r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}; \theta_{\texttt{old}})$ is the target Q-value, and $\theta_{\texttt{old}}$ is an older copy of the parameters $\theta$. To do so, we store trajectories containing transitions $(s_t, a_t, r_t, s_{t+1})$ in a replay buffer that we sample from Mnih et al. (2015). To stabilize learning, we use Double-Q-learning Van Hasselt et al. (2016) to choose the next action: $a_{t+1} = \arg\max_a \widehat{Q}(s_{t+1}, a; \theta)$.

### 2.4.2 Attentive Object-Dynamics Model

Consider the global set of objects $\{o^g_{t,i}\}_{i=1}^m$, where $m$ is the number of objects in the environment. At each time-step, each object-image-patch the agent observes corresponds to a 2D projection of $o^g_{t,i}$, $\rho(o^g_{t,i})$ (or $\rho^{g,i}_t$ for short) and encodes it as $\boldsymbol{z}^{g,i}_t$. Given, an object-image-patch encoding $\boldsymbol{z}^{g,i}_t$ and a performed interaction $a_t$, we can define an object-dynamics model $D(\boldsymbol{Z}^o_t, \boldsymbol{z}^{g,i}_t, a_t)$ which produces the resultant encoding for $\rho^{g,i}_{t+1}$. We want $D(\boldsymbol{Z}^o_t, \boldsymbol{z}^{g,i}_t, a_t)$ to be closer to $\boldsymbol{z}^{g,i}_{t+1}$ than to encodings of other object-image-patches.

**Classification problem**. We can formalize this by setting up a classification problem. For an object-image-patch encoding $\boldsymbol{z}^{g,i}_t$, we define the *prediction* as the output of our object-dynamics model $D(\boldsymbol{Z}^o_t, \boldsymbol{z}^{g,i}_t, a_t)$. We define the *label* as the encoding of a visible object-image-patch at the next time-step with the highest cosine similarity to the original encoding $\boldsymbol{z}^{g,i}_+ = \arg\max_{\boldsymbol{z}^{g,j}_{t+1}} \cos(\boldsymbol{z}^{g,i}_t, \boldsymbol{z}^{g,j}_{t+1})$. We can then select $K$ random object-encodings $\{\boldsymbol{z}^o_{k,-}\}_{k=1}^K$ as *incorrect labels*. Rewriting $D(\boldsymbol{Z}^o_t, \boldsymbol{z}^{g,i}_t, a_t)$ as $D$, this leads to:

$$p(\boldsymbol{z}^{g,i}_{t+1} | \boldsymbol{Z}^o_t, a_t) = \frac{\exp(D^\top \boldsymbol{z}^{g,i}_+)}{\exp(D^\top \boldsymbol{z}^{g,i}_+) + \sum_k \exp(D^\top \boldsymbol{z}^o_{k,-})}. \tag{2.5}$$

The set of indices corresponding to visible objects at time $t$ is $v_t = \{i : \rho^{g,i}_t \text{ is visible at time } t\}$. The set of observed object-image-patch encodings is then $\boldsymbol{Z}^o_t = \{\boldsymbol{z}^{o,j}_t\} = \{\boldsymbol{z}^{g,i}_t\}_{i \in v_t}$. Assuming the probability of each object's next state is conditionally independent given the current set of objects

and the action taken, we arrive at the following objective:

$$\begin{aligned}\mathcal{L}_{\texttt{model}} &= \mathbb{E}_{z_t,a_t,z_{t+1}}\left[-\log p(\boldsymbol{Z}^o_{t+1}|\boldsymbol{Z}^o_t, a_t)\right] \\ &= \mathbb{E}_{z_t,a_t,z_{t+1}}\left[-\sum_{i\in v_{t+1}} \log p(z^{g,i}_{t+1}|\boldsymbol{Z}^o_t, a_t)\right].\end{aligned} \tag{2.6}$$

Our final objective becomes:

$$\mathcal{L} = \mathcal{L}_{\texttt{DQN}} + \beta^{\texttt{model}}\mathcal{L}_{\texttt{model}}. \tag{2.7}$$

**Leveraging inter-object attention for improved accuracy.** Consider slicing an apple with a knife. When selecting "slice" on the apple patch, learning to attend to the knife patch **both** enables more accurate estimation of $Q$-values and higher model-prediction accuracy. We can accomplish this by incorporating $A(\boldsymbol{z}^{g,i}, \boldsymbol{Z}^o)$ into our object-model as follows:

$$D(\boldsymbol{Z}^o_t, \boldsymbol{z}^{g,i}_t, a_t) = f_{\texttt{model}}([\boldsymbol{z}^{g,i}_t, A(\boldsymbol{z}^{g,i}_t, \boldsymbol{Z}^o_t), \boldsymbol{z}^a_t]). \tag{2.8}$$

To learn an action encoding $\boldsymbol{z}^a_t$ for action $a_t$, following Oh et al. (2015); Reed et al. (2014), we employ multiplicative interactions so our learned action representation $\boldsymbol{z}^a_t$ compactly models the cartesian product of all base actions $b$ and object-image-patch selections $o_c$ as

$$\boldsymbol{z}^a_t = W^o \boldsymbol{z}^{g,c}_t \odot W^b b_t, \tag{2.9}$$

where $W^o \in \mathbb{R}^{d_a \times d_o}$, $W^b \in \mathbb{R}^{d_a \times |\mathcal{A}_I|}$, and $\odot$ is an element-wise hadamard product. In practice, $f_{\texttt{model}}$ is a small 1- or 2-layer neural network making this method compact and simple to implement.

## 2.5 Experiments

The primary aim of our experiments is to study how different auxilliary tasks for learning object-representations enable sample complexity comparable to an agent with oracle object-knowledge. We additionally study the degree to which each auxilliary task enables object-representation learning that captures the ground-truth knowledge present in our oracle agent. We conclude this section with ablation experiments studying the importance of different forms of object-knowledge in task learning.

**Evaluation Settings**. The agent's spawning location is randomized from $81$ grid positions. The agent receives a terminal reward of $1$ if its task is completed successfully and $0$ otherwise. It receives a time-step penalty of $-0.04$. Episodes have a time-limit of $500$ time-steps. The agent has a budget of $500K$ samples to learn a task. This was the budget needed by a relational agent with oracle

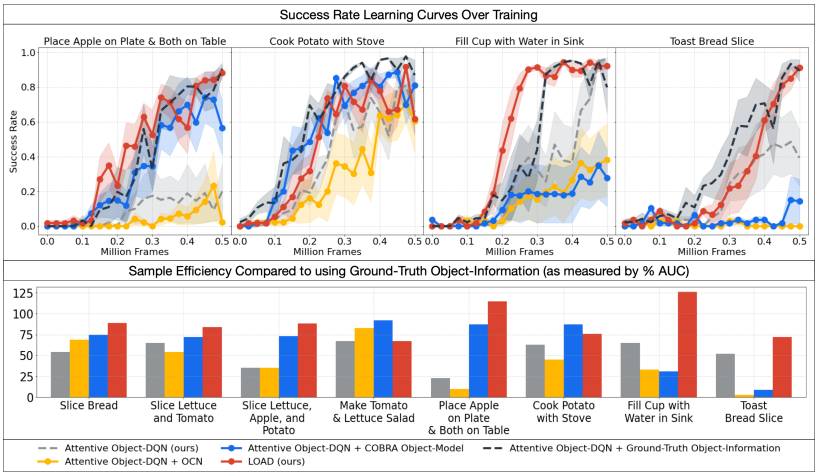


Figure 2.3: **Top-panel**: we present the success rate over learning for competing auxiliary tasks. We seek a method that best enables our Attentive Object-DQN (grey) to obtain the sample-efficiency it would from adding Ground-Truth Object-Information (black). We visually see that LOAD (red) is best able to learn more quickly on tasks that require using containment-relationships (e.g. a cup in a sink) or recognizing changing object properties (e.g. a toaster turning on with bread in it).
**Bottom-panel**: by measuring the % AUC achieved by each agent w.r.t to the agent with ground-truth information, we can measure how close each method is to the performance of an agent with ground-truth object-knowledge. We find LOAD (red), which learns an attentive object-model best closes the performance gap on $6/8$. We hypothesize that this is due to our object-model's ability to capture oracle object-information about object-categories, object-properties, and object-relations. We show evidence for this in Table 2.2.

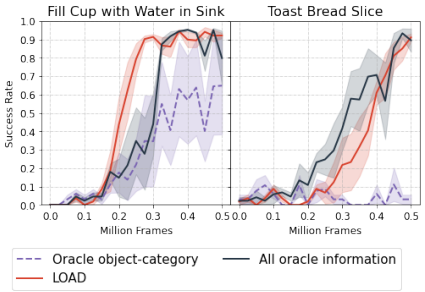


Figure 2.4: Ablation of object-properties and object-relations from oracle. With only oracle object-category information, the oracle can't learn these tasks in our sample budget.

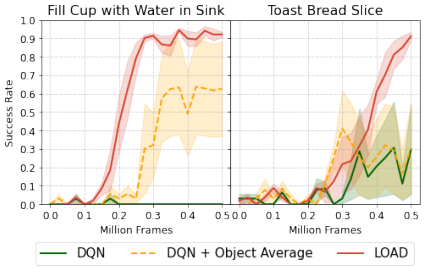


Figure 2.5: Ablation of inter-object attention in policy. Without this, DQN cannot learn these tasks in our sample-budget. See §2.5.3 for details.

object-information.

**Baseline methods for comparison**. In order to study the effects of competing object representation learning methods, we compare combining Attentive Object-DQN with the Attentive Object-Model against four baseline methods:

1. **Attentive Object-DQN**. This baseline has no auxiliary task and lets us study how well an agent can learn from the sparse-reward signal alone.
2. **Ground-Truth Object-Information**. This baseline has no auxiliary task. Instead, we supply the agent with 14 ground-truth features from the simulator. They roughly describe an object's category (is it a toaster?), its properties (e.g., is it on/off/etc.?), and relevant object-containment (e.g., what object is this object inside of?).
3. **OCN**. The Object Contrastive Network (Pirk et al., 2019). This method also employs a classification-like contrastive learning objective to cluster object-images across time-steps. However, it doesn't use an object-model or incorporate action-information. This enables us to study the importance of incoporating an object-model and action information.
4. **COBRA Object-Model**. This is the object-model employed by the COBRA RL agent (Watters et al., 2019). They also targeted improved sample-efficiency—though in a simpler, fully-observable 2D environment with shapes that only needed differentiation by category. Their model had no mechanism for incorporating inter-object relations into its predictions.

To enable faster learning in a sparse-reward setting, all baselines sample training batches using a second self-imitation learning replay buffer of successful episodes (Oh et al., 2018).

### 2.5.1 Task Performance

**Metrics**. We evaluate agent performance by measuring the agent's success rate over 5K frames every 25K frames of experience. The success rate is the proportion of episodes that the agent completes. We compute the mean and standard error of these values across $5$ seeds. To study sample-efficiency, we compare each method to "Ground-Truth Object-Information" by computing what percent of the Ground-Truth Object-Information mean success rate AUC each method achieved.

We present sample-efficiency bar plots for all 8 of our tasks in Figure 2.3. We found that using containment relationships and recognizing changing object-properties (Challenges C & D in §2.3) were most indicative of task difficulty. We only present learning curve results for 4 tasks which match this criteria.

**Performance**. We find that using Ground-Truth Object-Information is able to get the highest success rate on all tasks. Attentive Object-DQN performs below all methods besides OCN on $7/8$ tasks. Surprisingly, Attentive Object-DQN outperforms OCN on $5/8$ tasks. OCN doesn't incorporate action-information when learning to represent object-images across time-steps. We

hypothesize that this leads it to learn degenerate object-representations that cannot discriminate object-propertiess that change due to actions, something important for our tasks.

In terms of sample-efficiency, our Attentive Object-Model comes closest to Ground-Truth Object-Information on $6/8$ tasks. For tasks that require using objects together, such as "Fill Cup with Water" where a cup must be used in a sink or "Toast Bread Slice" where bread must be cooked in a toaster, our Attentive Object-Model significantly improves over the COBRA Object-Model. Interestingly, sample-efficiency goes above 100% on 2 tasks. We suspect that this is because the object-model provides a learning signal for inter-object attention which is not provided by oracle information.

### 2.5.2 Analysis of Learned Object Representations

In Table 2.2, we explore our conjecture that the key to strong task-learning performance is an agent's ability to capture the information present in the oracle agent. To study this, we freeze the parameters of each encoding function, and add a linear layer to predict object-categories, object-properties, and containment relationships using a dataset of collected object-interactions we construct. We find that our object-model best captures the information present in the oracle agent.

| Representation Learning Method | Category | Object-Properties | Containment Relationship |
|---|---|---|---|
| OCN | $39.2 \pm 8.2$ | $66.5 \pm 8.5$ | $69.1 \pm 9.0$ |
| COBRA Object-Model | $79.8 \pm 2.8$ | $73.4 \pm 8.9$ | $83.1 \pm 5.8$ |
| Attentive Object-Model | $\mathbf{88.6 \pm 3.5}$ | $\mathbf{98.6 \pm 0.3}$ | $\mathbf{94.3 \pm 0.6}$ |

Table 2.2: Performance of different unsupervised learning methods for learning object-features. We find that our object-model best captures features present in the oracle agent, providing evidence that its strong object-representation learning is responsible for its strong task-learning performance.

### 2.5.3 Ablations

**Importance of object-properties & object-relations**. To verify that capturing object-properties and -relations is key, we train an agent with only oracle object-category information. We find that this agent is not able to learn tasks that require using objects together as object-properties change in our sample-budget (see Figure 2.4).

**Importance of inter-object attention**. In order to verify the utility of using attention as an inductive bias for capturing object-relations, we ablate attention from both Attentive Object-DQN and our Attentive Object-Model. First, we look at two variants of Attentive Object-DQN without attention. The first is a regular DQN. In the second, we incorporate inter-object information by

using the average of all present object-embeddings (DQN + Object Average). Neither learns our tasks in the sample-budget (see Figure 2.5).

Additionally, we look at performance where our policy can use inter-object attention but remove inter-object attention from our object-model. Without attention, we still get relatively good performance with 70% success rate; however, attention in the object-model helps increase this to 90%+ (TODO: add plot).

## 2.6 Conclusion

We have shown that learning an attentive object-model can enable sample-efficient learning in high-fidelity, 3D, object-interaction domains without access to expert demonstrations or ground-truth object-information. Further, when compared to strong unsupervised learning baselines, we have shown that our object-model best captures object-categories, object-properties, and containmennt-relationships. We believe that LOAD is a promising step towards agents that can efficiently learn complex object-interaction tasks.

# CHAPTER 3

# Feature-Attending Recurrent Modules for Generalization in Reinforcement Learning

To generalize in object-centric tasks, a reinforcement learning (RL) agent needs to exploit the structure that objects induce. Prior work has either hard-coded object-centric features, used complex object-centric generative models, or updated state using local spatial features. However, these approaches have had limited success in enabling general RL agents. Motivated by this, we introduce "Feature-Attending Recurrent Modules" (FARM), an architecture for learning state representations that relies on simple, broadly applicable inductive biases for capturing spatial and temporal regularities. FARM learns a state representation that is distributed across multiple modules that each attend to spatiotemporal features with an expressive feature attention mechanism. This enables FARM to represent diverse object-induced spatial and temporal regularities across subsets of modules. We hypothesize that this enables an RL agent to flexibly recombine its experiences for generalization. We study task suites in both 2D and 3D environments and find that FARM better generalizes compared to competing architectures that leverage attention or multiple modules.

## 3.1 Introduction

Objects are key to real-world tasks. For example, a self-driving car needs to represent the movements of other cars, and a household robot needs to recognize and use kitchen items. In order to generalize across tasks with objects, a reinforcement learning (RL) agent should capture and exploit object-induced structure present across the tasks.

One way to capture this structure is in an agent's state representation. Unfortunately, flexibly capturing objects in a state representation is challenging because objects manifest in a multitude of ways. Consider a household robot tasked with cooking. Completing the task might require memory about objects that range in size, shape, and color (e.g. a stove vs. a tomato). Additionally, other agents may be in motion requiring that the agent represent temporal information about

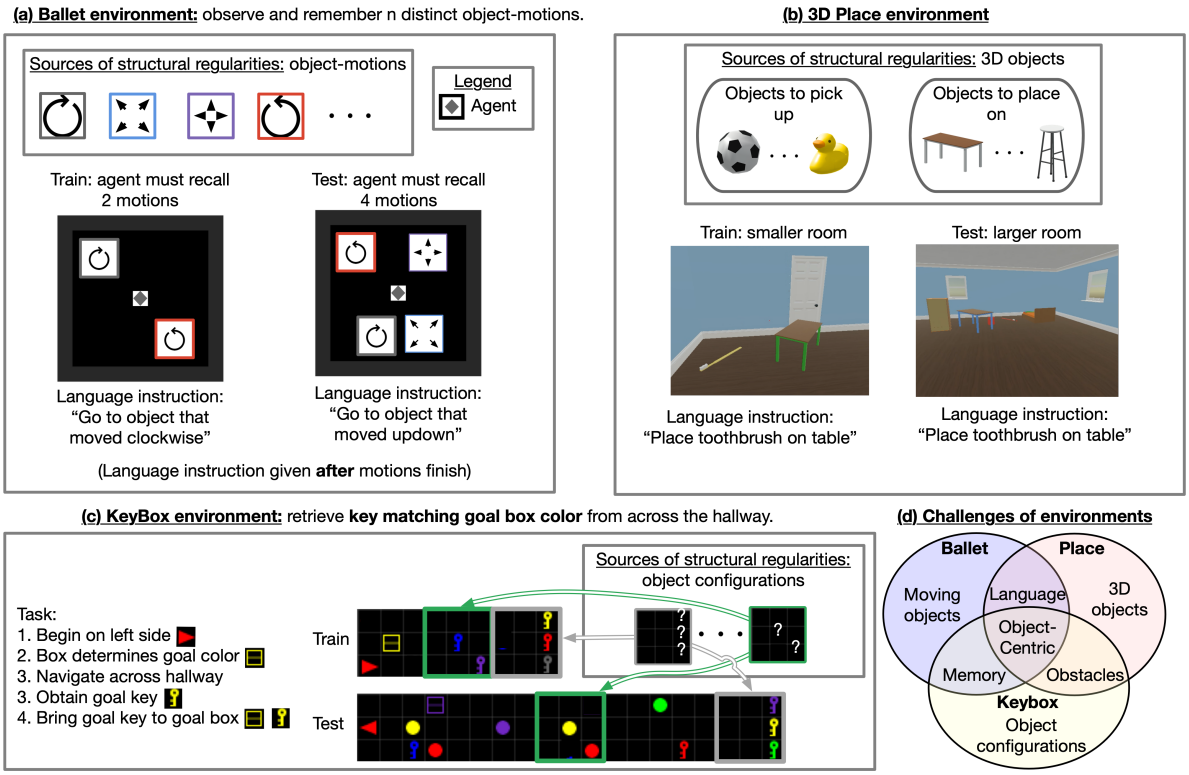


Figure 3.1: **We study three environments with different structural regularities induced by objects.** In the Ballet environment, tasks share regularities induced by object motions; in the KeyBox environment, they share regularities induces by object configurations; and in the Place environment, tasks share regularities induces by 3D objects. The Ballet and KeyBox environments pose learning challenges for long-horizon memory and require generalizing to more objects. The KeyBox and Place environments pose learning challenges in obstacle navigation and requires generalizing to a larger map. Videos of our agent performing these tasks: https://bit.ly/3kCkAqd.

objects in order to coordinate navigation. It is unclear how to best incorporate objects into a state representations to enable generalization.

Prior work has attempted to capture object-induced task structure by hand-designing object-centric state features (Diuk et al., 2008; Carvalho et al., 2021b; Borsa et al., 2019; Marom and Rosman, 2018). The "COBRA" agent (Watters et al., 2019) avoids hand-designing features by learning an object-centric generative model. However, these methods are limited in their generality because they rely on relatively strong inductive biases. For example, COBRA relies on environments being fully-observable and objects having regular shapes to learn representations by predicting object segmentations. We focus on weak inductive biases in order to maximize an architecture's flexibility.

Objects can be described by subsets of features over space and time. We conjecture that weak inductive biases for capturing subsets of features over space and time may enable agents that can

flexibly incorporate objects into state across a wide range of environments.

We propose *Feature Attending Recurrent Modules (FARM)*, a simple but flexible architecture for learning state representations when tasks share object-induced structural regularities. FARM learns state representations that are *distributed* across multiple, smaller recurrent modules. To help motivate this, consider word embeddings. A word embedding can represent more information than a one-hot encoding of the same dimension because subsets of dimensions can coordinate activity to represent different patterns of word usage. Analogously, learning multiple modules enables FARM to coordinate subsets of modules to represent different temporal segments in an agent's experiences. To capture general object-induced patterns, modules select observation information to update with by weighting observation features.

We study FARM across three diverse object-centric environments, each with their own suite of tasks that share object-induced structural regularities. Tasks in the Ballet environment share regularities induced by object motions; tasks in the Place environment share regularities induced by navigating towards and around 3D objects; and tasks in the KeyBox environment share regularities induced by object configurations. These environments present a number of challenges. First, their state-space grows exponentially with the number of objects. The more distractor objects an environment has, the larger the chance an object will obstruct an agent's path. This requires learning a policy that can navigate around distractor-based obstacles. When task objects appear in sequence, this can require long-horizon memory of object information (e.g. of goal information). Finally, tasks defined by language can require an agent learn a complex mapping (e.g. to object motions and to irregular shapes in our tasks). Across these environments, we study an agent's ability to recombine object-oriented memory, obstacle-avoidance, and navigation to longer tasks with more objects.

We compare against methods with weak inductive biases for enabling objects to emerge in a state representation. Recent work has shown that spatial attention is a simple inductive bias for strong performance on object-centric vision tasks because it enables attending to individual objects (Greff et al., 2020; Locatello et al., 2020; Goyal et al., 2020a). Thus, we compare against recent RL agents that leverage spatial attention for object-centric state-update functions (Goyal et al., 2020b; Mott et al., 2019).

**Summary of contributions**.

1. We introduce FARM, a simple architecture for learning an agent's state representation when an environment has object-induced structural regularities (§3.4).
2. We show that FARM's simple inductive biases–feature attention and multiple modules–synergistically enable generalizing (a) memory to longer combinations of object motions (§3.5.1); (b) navigation to 3D objects in larger environments (§3.5.2); and (c) memory of goal information to longer tasks with more distractors (§3.5.3).

3. We compare against spatial attention, which has been shown to enable object-centric state updates. We find mixed benefits in for our diverse object-centric tasks, including interference with learning multiple modules.
4. We analyze FARM's state representations and provide evidence that object-induced regularities are represented in patterns that are flexibly distributed across subsets of modules (§3.5.3.1).

## 3.2 Related work on generalization in deep RL

The key question for generalization is how to capture structure in the problem in a flexible way. How much structure do you build in? How much do you let the agent discover? Some work takes a data-driven approach (Tobin et al., 2017; Packer et al., 2018; Hill et al., 2020; Justesen et al., 2019). Others have a policy that captures task structure with either hierarchical RL (Oh et al., 2017b; Zhang et al., 2018; Sohn et al., 2018, 2021; Brooks et al., 2021) or successor features (Borsa et al., 2019; Barreto et al., 2020). A final strand focuses on learning invariant representations (Higgins et al., 2017; Chaplot et al., 2018; Lee et al., 2020; Zhang et al., 2021) or building in inductive biases (Mott et al., 2019; Goyal et al., 2020b). In this work we focus on weak inductive biases for capturing structure. Below we review approaches most closely related to ours.

**Generalizing across object-centric tasks** dates back at least to object-oriented MDPs (Džeroski et al., 2001; Diuk et al., 2008) which enabled generalization by representing dynamics with logical object attributes (Kansky et al., 2017; Marom and Rosman, 2018). Successor features have also leveraged objects for generalization by formulating rewards as linear with object-centric features (Borsa et al., 2019; Barreto et al., 2020). A common thread among these directions is that they relied on hand-designed object features. Watters et al. (2019) avoided hand-designing features by learning an object-centric generative model (Burgess et al., 2019). However, they focused on fully-observable top-down environments with regular shapes, which allowed them to predict future object masks. This is incompatible with our environments. While research on object-centric models (Kabra et al., 2021; Zoran et al., 2021) has progressed, these methods commonly add training complexity (more objective terms, extra modules, etc.) and make stronger assumptions (e.g. on the number of objects). We differ from this work because we focus on simple, broadly applicable inductive biases for capturing object-induced task regularities.

**Generalizing with feature attention** has also been studied by Chaplot et al. (2018). They showed that mapping language instructions to non-linear feature coefficients enabled generalizing to tasks specified over unseen feature combinations in a 3D environment. While FARM also learns non-linear feature coefficients, our work has two important differences. First, we develop a multi-head version where individual feature coefficients are produced by their own recurrent modules. This enables FARM to leverage this form of attention in settings where language instructions don't

indicate what to attend to (this is true in $2/3$ of our tasks). Second, we are the first to show that feature attention enables generalizing memory of object motions and of goal information to longer tasks (Figure 3.4 and Figure 3.6, respectively).

**Generalizing with top-down spatial attention**. Most similar to FARM are the Attention Augmented Agent (AAA) (Mott et al., 2019) and Recurrent Independent Mechanisms (RIMs) (Goyal et al., 2020b). Both are recurrent architectures that leverage spatial attention to learn an object-centric state-update function. Both showed generalization to novel distractors. The major difference between AAA, RIMs, and FARM is that FARM attends to an observation with feature attention as opposed to spatial attention. Our experiments indicate that spatial attention has limited utility in updating state during reinforcement learning of tasks defined by object motions (Figure 3.4) or 3D objects (Figure 3.5). In terms of modularity, we also show different results from RIMs who showed that their modules "specialize". Our experiments suggest that in FARM, a modular state instead leads subset of modules to *jointly* represent regularities in an agent's experience (§3.5.3.1).

## 3.3 Problem setting

We study generalization across tasks within deterministic, partially-observable, pixel-based environments. Within an environment, a task is defined by a Partially Observable Markov decision processes (POMDP): $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, R, T, \psi \rangle$. $\mathcal{S}$ corresponds to environment states, $\mathcal{A}$ corresponds to actions that agent can take, $\mathcal{O}$ corresponds to the agent's observations, $r = R(s, a) \in \mathbb{R}$ is the reward function, $s' = T(s, a) \in \mathcal{S}$ is the environment transition function, and $o = \psi(s) \in \mathcal{O}$ is an observation function that maps the underlying environment state to an RGB image.

We seek an RL agent that learns to perform tasks by finding a policy $\pi$ that maximizes the expected discounted sum of rewards it obtains starting at a state $s$: $V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)\right]$—also known as the *value* of a state. In a POMDP, the agent doesn't have access to the environment state. A common strategy is to instead learn an *"agent state"* representation, $s_t^A$, that compresses the full history $(o_1, a_1, r_1, \ldots, a_{t-1}, o_t)$ into a sufficient statistic suitable for selecting actions. The agent state is commonly learned with a recursive function $s_t^A = \eta(o_t, a_{t-1}, r_{t-1}, s_{t-1}^A)$.

**Object-induced structural regularities**. We study object-centric environments, where objects induce structural regularities across tasks in the reward functions $R$, transition functions $T$, and observation functions $\psi$. For example, consider the KeyBox environment in Figure 3.1 (c). First, $R(s, a)$ always specifies the goal key based on a goal box. Second, whenever the agent has to navigate around an obstacle (see Figure 3.2, b), the agent always sees the sprite it controls move closer to an object and then around it. This is true regardless of *where* in the hallway the agent observes the obstacle because of regularities in the transition function $T(s, a)$ and observation function $\psi(s)$. We want an agent that captures these regularities in its representation for state to

enable zero-shot generalization to new tasks.

## 3.4 Architecture: FARM

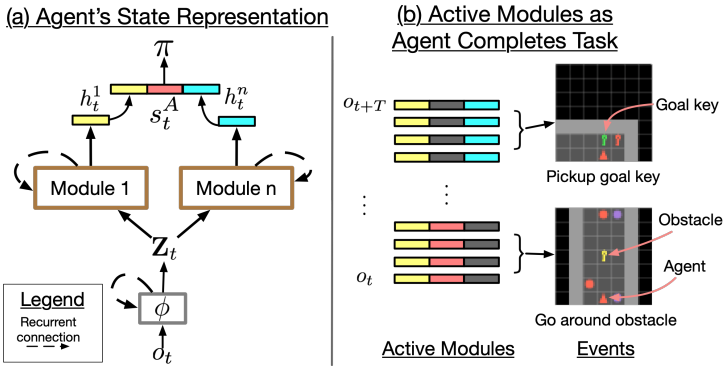


Figure 3.2: **Overview of FARM**. (a) FARM learns an agent state representation that is distributed across $n$ recurrent modules. (b) By distributing agent state across multiple modules, FARM is able to represent different object-centric task regularities, such as navigating around obstacles or picking up goal keys, across subsets of modules. We hypothesize that this enables a deep RL agent to flexibly recombine its experience for generalization. See §3.4 for details on the architecture and §3.5.3.1 for supporting analysis.

We propose a new architecture, "Feature Attending Recurrent Modules" (FARM) for learning an agent's state representations when an environment has object-induced structural regularities. We provide an overview of the architecture in Figure 3.2. Instead of representing agent state with a single recurrent function, FARM learns a state representation that is distributed across $n$ recurrent functions $\{\eta^k\}_{k=1}^n$, which we call modules (Figure 3.2, a). Distributing state across modules allows subsets of modules to jointly represent different regularities in the agent's experience (Figure 3.2, b). We hypothesize that having subsets of modules represent different regularities in the agent's experience enables the agent to flexibly recombine its experience for generalization.

At each time-step $t$, each module updates with both observation features and information from other modules. First, the agent computes observation features with a recurrent observation encoder, $\boldsymbol{Z}_t = \phi(o_t, \boldsymbol{Z}_{t-1})$. Afterward, each module creates a *query* vector by combining its previous module-state with the previous action and reward, $q_{t-1}^k = [h_{t-1}^k, a_{t-1}, r_{t-1}]$. The query is used to attend to observation features via a dynamic feature attention mechanism $u_t^k = f_{\texttt{att}}^k(\boldsymbol{Z}_t, q_{t-1}^k)$. The query is also used to retrieve information from other modules with a transformer-style attention mechanism $\nu_t^k = f_{\texttt{share}}^k(s_{t-1}^A, q_{t-1}^k)$. (We explain both attention mechanisms in more detail below). Each module updates with both attention outputs to produce the next module-state $h_t^k = \eta^k(u_t^k, \nu_t^k, q_{t-1}^k)$. Agent state is then defined by the combination of these module-states $s_t^A = [h_t^1, \ldots, h_t^n]$. We provide an

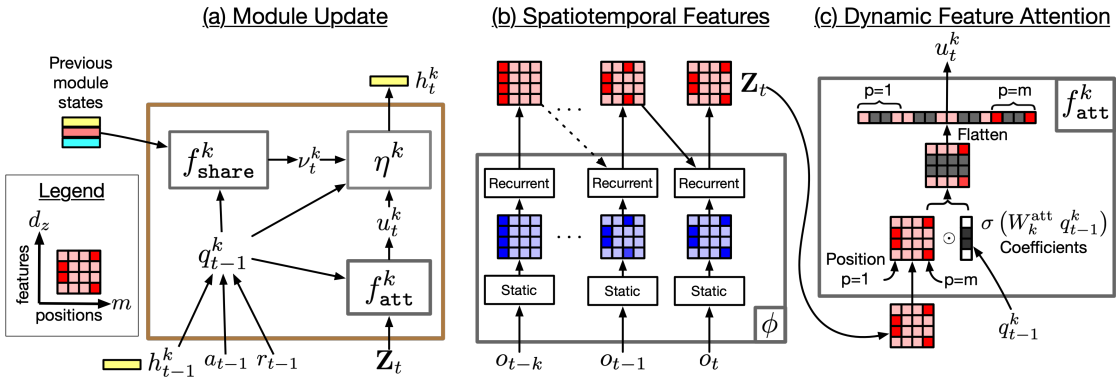


Figure 3.3: **Computations of FARM**. (a) Schematic of updates. See 2nd paragraph in §3.4 for details. (b) In order to update with features that describe both visual and temporal regularities, the agent learns structured spatiotemporal features $\mathbf{Z}_t \in \mathbb{R}^{m \times d_z}$ that share $d_z$ spatio-temporal features across $m$ spatial positions. Here we show toy computations where static observations features (blue) on the top and bottom row move to spatial positions to the right. The resultant spatio-temporal features (red) also include temporal information about the features (here, that the features came from leftward spatial positions). (c) $f^k_{\texttt{att}}$ computes coefficients for features and applies them uniformly across all spatial positions. This allows the agent to attend to all spatial position that possess desired features.

overview in Figure 3.3 (a) and summarize the computations below:

$$\boldsymbol{Z}_t = \phi(o_t, \boldsymbol{Z}_{t-1}) \qquad \text{obs features} \tag{3.1}$$

$$q^k_{t-1} = [h^k_{t-1}, a_{t-1}, r_{t-1}] \qquad \text{query} \tag{3.2}$$

$$u^k_t = f^k_{\texttt{att}}(\boldsymbol{Z}_t, q^k_{t-1}) \qquad \text{obs attention} \tag{3.3}$$

$$\nu^k_t = f^k_{\texttt{share}}(s^A_{t-1}, q^k_{t-1}) \qquad \text{share info} \tag{3.4}$$

$$h^k_t = \eta^k(u^k_t, \nu^k_t, q^k_{t-1}) \qquad \text{module update} \tag{3.5}$$

$$s^A_t = [h^1_t, \ldots, h^n_t] \qquad \text{agent state} \tag{3.6}$$

We now describe each computation in more detail.

**Structured spatiotemporal observation features.** Our first insight is that modules can attend to features describing an object's motion if an agent learns observation features that describe both spatial and temporal regularities. An agent can accomplish this by learning structured spatiotemporal features with a recurrent observation encoder $\boldsymbol{Z}_t = \phi(x_t, \boldsymbol{Z}_{t-1}) \in \mathbb{R}^{m \times d_z}$ that share $d_z$ features across $m$ spatial positions[1]. At each spatial position, these features both describe what is there visually along with temporal information about the recent dynamics of these features. We show example toy computations in Figure 3.3 (b).

[1]One can convert height by width observation features as follows: $\mathbb{R}^{h \times w \times d_z} \to \mathbb{R}^{hw \times d_z}$

**Dynamic feature attention.** Our second insight is that feature attention is an expressive attention function that can focus on desired information present across all spatial positions in observation features. An agent accomplishes this by having a module predict feature coefficients that it applies to uniformly across all spatial positions in $\boldsymbol{Z}_t$ (Perez et al., 2018; Chaplot et al., 2018). We show example toy computations in Figure 3.3 (c). We found it useful to linearly project the features before and after using shared parameters as in Andreas et al. (2016); Hu et al. (2018). The operations are summarized below:

$$f^k_{\texttt{att}}(\boldsymbol{Z}_t, q^k_{t-1}) = \left(\boldsymbol{Z}_t W_1 \odot \sigma(W^{\texttt{att}}_k q^k_{t-1})\right) W_2 \tag{3.7}$$

where $\odot$ denotes an element-wise product over the feature dimension and $\sigma$ is a sigmoid non-linearity. Since our features capture dynamics information, this allows a module to attend to object motion (§3.5.1). When updating, we flatten the attention output. Flattening leads all spatial positions to be treated uniquely and allows a module to represent aspects of the observation that span multiple positions, such as 3D objects (§3.5.2) and spatial arrangements of objects (§3.5.3). Since the feature-coefficients for the next time-step are produced with observation features from the current time-step, modules can *dynamically shift* their attention when task-relevant events occur (see Figure 3.7, b for an example).

**Sharing information.** Similar to RIMs (Goyal et al., 2020b), before updating, each module retrieves information from other modules using transformer-style attention (Vaswani et al., 2017). We define the collection of previous module-states as $\boldsymbol{H}_{t-1} = \left[h^{(1)}_{t-1}; \ldots; h^{(n)}_{t-1}; \boldsymbol{0}\right] \in \mathbb{R}^{(n+1)\times d_h}$, where $\boldsymbol{0}$ is a null-vector used to retrieve no information. A module computes a "retrieval query" to search for information as $q^k_r = q^k_{t-1} W^{\texttt{que}}_k \in R^{d_h}$. That module computes "retrieval keys and values" as $K^k = \boldsymbol{H}_{t-1} W^{\texttt{key}}_k \in R^{n+1\times d_h}$ and $V^k = \boldsymbol{H}_{t-1} W^{\texttt{val}}_k \in R^{n+1\times d_h}$, respectively. Each module then retrieves information as follows:

$$f^k_{\texttt{share}}(s^A_{t-1}, q^k_{t-1},) = \text{softmax}\left(\frac{q^k_r {K^k}^\top}{\sqrt{d_h}}\right) V^k. \tag{3.8}$$

Intuitively, the dot-product inside the softmax is computing $n+1$ scores (one for each "key"), which then form probabilities. The outter dot-product multiplies each "value" by its probability and sums them to perform soft-selection.

## 3.5 Experiments

In this section, we study the following questions:

1. Can LOAD generalize memory to longer spatiotemporal combinations of object motions?

2. Can LOAD generalize navigation towards and avoidance of 3D objects to larger environments?
3. Can LOAD generalize memory of goal-information to larger maps with more distractor-based obstacles?

**Baselines.** Our first baseline is a common choice for learning state-representations, a **Long Short-term Memory (LSTM)** (Hochreiter and Schmidhuber, 1997). We study two other baselines that also attend to observation features: **Attention Augmented Agent (AAA)** (Mott et al., 2019) and **Recurrent Independent Mechanisms (RIMs)** (Goyal et al., 2020b). Both employ transformer-style attention (Locatello et al., 2020; Vaswani et al., 2017) to attend to individual *spatial positions* by reducing observation features to a weighted average over spatial positions. We instead attend to *features shared across all spatial positions*. RIMs, like FARM, represents state with a set of recurrent modules.

| Method | Observation Attention | Modular State |
|---|---|---|
| LSTM | ✗ | ✗ |
| AAA | Spatial | ✗ |
| RIMs | Spatial | ✓ |
| FARM (Ours) | Feature | ✓ |

Table 3.1: Baselines.

**Implementation details.** We implement our recurrent observation encoder, $\phi$, as a ResNet (He et al., 2016) followed by a Convolutional LSTM (ConvLSTM) (Shi et al., 2015). We implement the update function of each module with an LSTM. We used multihead-attention (Vaswani et al., 2017) for $f^k_{\texttt{share}}$. We trained the architecture with the IMPALA algorithm (Espeholt et al., 2018) and an Adam optimizer (Kingma and Ba, 2014).

### 3.5.1 Generalizing memory to more object motions

We study this with the "Ballet" grid-world (Lampinen et al., 2021) shown in Figure 3.1 (a). **Tasks**. The agent controls a white square that begins in the middle of the grid. There are $m$ other "ballet-dancer" objects that move with a one of 15 distinct object motions. The dances move in sequence for 16 time-steps with a 48-time-step delay in between. After all dancers finish, the agent is given a language instruction indicating the correct ballet dancer to navigate towards. All shapes and colors are randomized making motion the only feature indicating the goal object. **Observations**. The agent observes a top-down RBG image of the environment. **Actions**. The agent can move left, right, up, and down. **Reward** is 1 if it touches the correct dancer and 0 otherwise. **Tasks split**. Training tasks always consists of seeing $m = \{2, 4\}$ dancers; testing tasks always consists of seeing $m = \{8\}$ dancers. All agents learn with a sample budget of 2 billion frames. A poorly performing agent will obtain chance performance, $1/m$.

We present the training and generalization success rates in Figure 3.4. We learned spatiotemporal observation features with RIMs and AAA for a fair comparison. We found that only FARM is able to obtain above chance performance for training and testing. In order to understand the source of

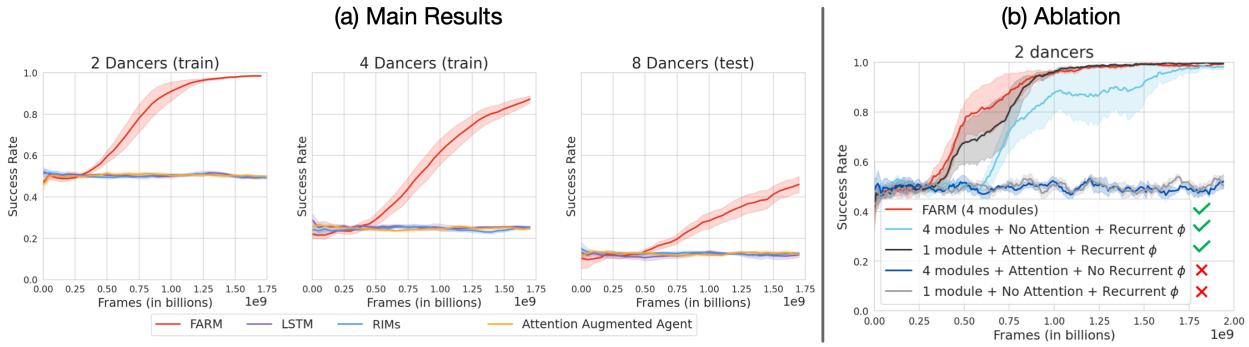


Figure 3.4: **FARM enables generalizing memory to longer spatiotemporal combinations of object motions**. We present the success rate means and standard errors computed using 5 seeds. (a) Only FARM is able to go above chance performance for each setting. (b) Given spatiotemporal features, we find that *either* using multiple modules *or* feature attention enables learning memory of object motions. These results suggest that spatial attention removes the benefits of using multiple modules for learning to remember object motions. Encouragingly, feature attention by itself can support it.

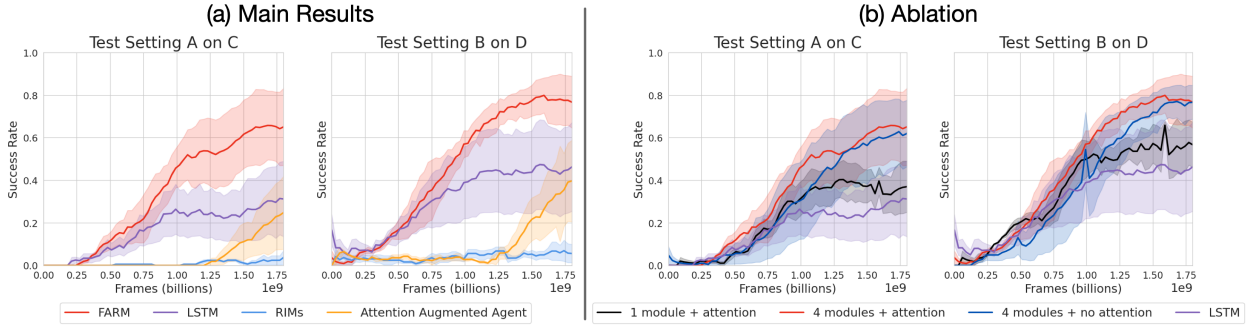


Figure 3.5: **FARM enables generalizing navigation towards and avoidance of 3D objects to a larger environment.** We present the success rate means and standard errors computed using 3 seeds. (a) FARM generalizes best. (b) Our performance benefits mainly come from learning multiple modules, though feature attention slightly improves performance and lowers variance. These results suggest that spatial attention interferes with reinforcement learning of 3D objects.

our performance, we ablate using a recurrent observation encoder, using multiple modules, and using feature-attention. We confirm that a recurrent encoder is required. Interestingly, we find that either using multiple modules or using our feature-attention enables task-learning, with our feature-attention mechanism being slightly more stable.

### 3.5.2 Generalizing navigation with more 3D objects

Here, we study the 3D Unity environment from Hill et al. (2020) shown in Figure 3.1 (b). **Tasks**. The agent is an embodied avatar in a room filled with task objects and distractor objects. The agent receives a language instruction of the form "*X on Y*" —e.g., "toothbrush on bed". We partition

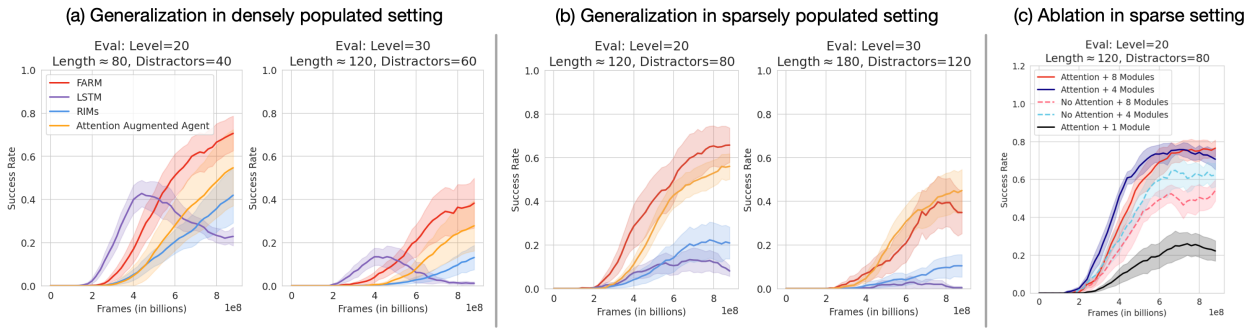


Figure 3.6: **FARM enables generalizing memory of goal-information and avoidance of obstacles to larger maps with more objects**. We show the the success rate mean and standard error computed using 10 seeds. (a) In the densely populated setting, FARM better generalizes to longer hallways with more distractors. (b) In the sparsely populated setting, FARM has slightly better performance than AAA for $2n_{\texttt{max}}$ but comparable performance for $3n_{\texttt{max}}$. (c) Using multiple modules and feature attention both improve generalization. These results suggest that spatial attention interferes with generalization benefits of learning multiple modules. Learning feature attention and multiple modules, instead, act synergistically.

objects into two sets as follows: pickup-able objects $O_1 = A \cup B$ and objects to place them on $O_2 = C \cup D$. **Observations.** The agent receives first-person egocentric RGB images. **Actions.** The agent has 46 actions that allow it to navigate, pickup and place objects. **Reward** is 1 if it completes the task and 0 otherwise. **Tasks split**. During training the agent sees $A \times D$ and $B \times C$ in a $4m \times 4m$ room with 4 distractors, along with $A \times C$ and $B \times D$ in a $3m \times 3m$ room with 0 distractors. We test the agent on $A \times C$ and $B \times D$ in a $4m \times 4m$ room with 4 distractors. We also train with "Go to X" and "Lift X".

We present the generalization success rate in Figure 3.5. We find that baselines which used spatial attention learn more slowly than an LSTM or FARM. Additionally, both models that use spatial attention have poor performance until the end of training where AAA begins to improve. FARM achieves relatively good performance, achieving a success rate of $60\%$ and $80\%$ on the two test settings, respectively.

### 3.5.3 Generalizing to larger maps with more objects

To study this, we create the "keyBox" environment depicted in Figure 3.1 (c). **Tasks** are defined with $n$ levels. Each level is a hallway with a single box and a *key of the same color* that the agent must retrieve. The agent and the box always starts in the left-most end and the goal key always starts in the right-most end. The agent always begins in the first level. It is teleported to the next level after placing the goal key next to the goal box. The hallway for level $n$ consists of a length-$n$ sequence of $w \times w$ environment subsections. Each subsection contains $d$ distractor objects. **Observations**.

The agent observes egocentric top-down images over a short segments of the hallway. **Actions**. The agent can move forward, turn left, turn right, pick up objects and drop them. **Rewards**. When completing a level, the agent gets a reward of $n/n_{\texttt{max}}$ where $n_{\texttt{max}}$ is the maximum level. **Tasks split**. Learning tasks include levels $1$ to $n_{\texttt{max}} = 10$. Test tasks only use levels $2n_{\texttt{max}}$ and $3n_{\texttt{max}}$. We study two generalization settings: a *densely populated setting* with subsections of area $w^2 = 9$ and $d = 2$ distractors, and a *sparsely populated setting* with subsections of area $w^2 = 25$ and $d = 4$ distractors.

We present the generalization success rates in Figure 3.6. In the dense setting, we see an LSTM quickly overfits in both settings. All architectures with attention continue to improve in generalization performance as they continue training. In the dense setting, we find that FARM generalizes better (by about $20\%$ for AAA and about $30\%$ for RIMs). In the sparse setting, both RIMs and an LSTM fail to generalize above $30\%$. FARM generalizes better than AAA for level $20$ but gets comparable performance for level $30$. In some ways, this is our most surprising result since it is not obvious that either learning multiple modules or using feature attention should help with this task. In the next section we study possible sources of our generalization performance.

#### 3.5.3.1 Analysis of state representations

We study the state representations FARM learns for categories of regularly occurring events. We collect $2000$ generalization episodes in level $20$. We segment these episodes into $6$ categories: pickup ball, drop ball, pickup wrong key, drop wrong key, pickup correct key, and drop correct key. We study the time-series of the L2 norm of each module-state and their attention coefficients. For reference, we also show the L2 norm for the entire episode. We note that we observed consistent activity that was not captured by our simple programmatic classification of states; for example, salient activity from module $0$ when the agent moved around obstacles. We show an example in Figure 3.7 a.

Due to space constraints, we present a subset of results in Figure 3.7. While some modules are selective for different recurring events such as attending to goal information (Figure 3.7, b), it seems that subsets of modules jointly represent different aspects of state. We hypothesize that this enables FARM to leverage overlapping sets of modules to store goal-information or to navigate around obstacles in a decoupled way that supports recombination. This is further supported by our ablation where we find that having $4$ or $8$ modules significantly outperforms using a single large module (all had about 8M params) (Figure 3.6, (c)). Feature attention consistently improves performance.

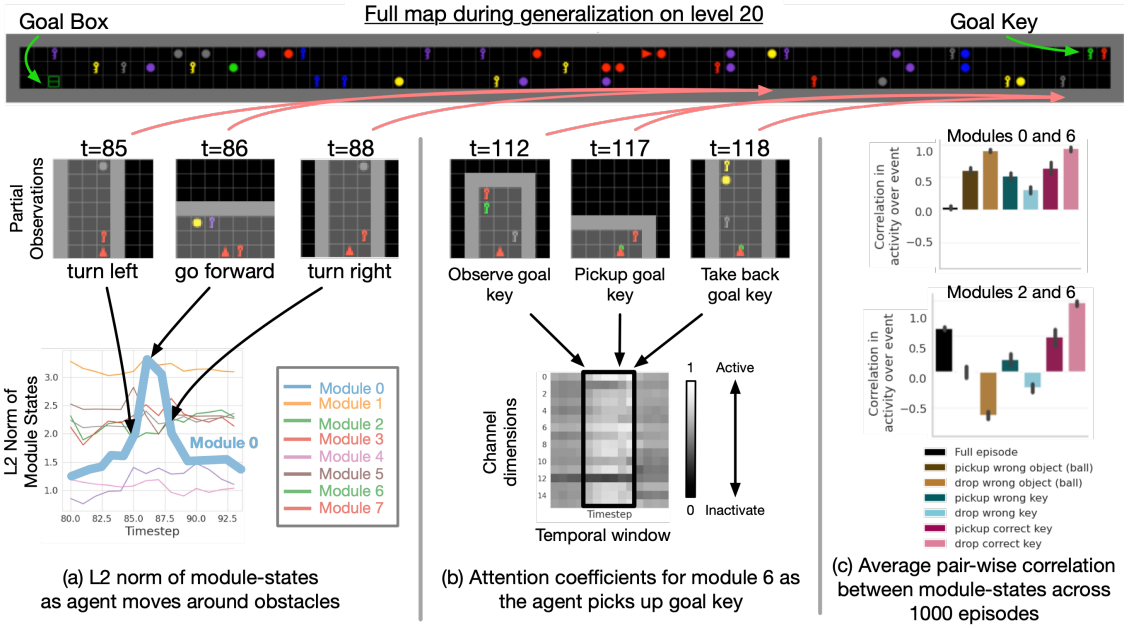


Figure 3.7: **We show evidence that different subsets of modules jointly represent object-induced task regularities.** (a) Module 0 commonly exhibits salient activity when the agent moves around an obstacle. (b) Module 6 activates its attention coefficients as the agent picks up the goal key. (c) Modules 2 and 6 correlate for picking up the correct key but anti-correlate for dropping the wrong object. This is similar to when neurons in word embeddings correlate for some words (e.g. man and king), but anti-correlate for other words (e.g. man and woman). In general, we find rich patterns of correlation and anti-correlation between the modules. These results suggest that FARM is representing task regularities across the modules in complicated and interesting ways. Videos of the state-activity and attention coefficients: https://bit.ly/3qCxatr.

## 3.6 Discussion and conclusion

We have presented FARM, a novel state representation learning architecture for environments that have object-induced structural regularities. Our results show that learning feature attention and multiple modules acts synergistically to support generalizing (a) memory to longer combinations of object-motions (§3.5.1); (b) navigation around 3D objects to larger environments (§3.5.2); and (c) memory of goal information to longer sequences of obstacles (§3.5.3). Our ablations suggest that feature attention mainly helps with long-horion memory. Interestingly, learning multiple modules helped across all conditions (memory, obstacle-avoidance, and language learning). Our analysis suggests that learning multiple modules enables subsets to represent object-centric task-relevant events in flexible ways. We hypothesize that this enables a deep RL agent to flexibly recombine its experience for generalization.

We compared FARM to other architectures that used spatial attention as a weak inductive bias for enabling objects to emerge in a state representation. We found that spatial attention hindered

learning tasks with object motions and 3D objects. In the KeyBox task, spatial attention seemed to help AAA most in the sparse setting with many objects. This makes sense since spatial attention has been shown to help with distractors and the agent mainly needed to ignore objects and move forward. Interestingly, pairing spatial attention with multiple modules (RIMs) removed the benefits of both.

One limitation of FARM is that feature attention is not spatially invariant since it treats all positions as unique. Future work can look to adapt this attention for something that still describes multiple positions but in a spatially invariant way. Another limitation of FARM is the length of temporal regularities it can capture. Transformers (Vaswani et al., 2017) have shown strong performance for representing long sequences. An interesting next-step might be to integrate FARM with a transformer. We hope that our work contributes to future RL algorithms that leverage weak inductive biases for capturing object-centric task regularities.

# CHAPTER 4

# Composing Task Knowledge With Modular Successor Feature Approximators

Recently, the Successor Features and Generalized Policy Improvement (SF&GPI) framework has been proposed as a method for learning, composing, and transferring predictive knowledge and behavior. SF&GPI works by having an agent learn predictive representations (SFs) that can be combined for transfer to new tasks with GPI. However, to be effective this approach requires state features that are useful to predict, and these state-features are typically hand-designed. In this work, we present a novel neural network architecture, "Modular Successor Feature Approximators" (MSFA), where modules both discover what is useful to predict, and learn their own predictive representations. We show that MSFA is able to better generalize compared to baseline architectures for learning SFs and modular architectures for learning state representations.

## 4.1 Introduction

Consider a household robot that needs to learn tasks including picking up dirty dishes and cleaning up spills. Now consider that the robot is deployed and encounters a table with both a spill and a set of dirty dishes. Ideally this robot can combine its training behaviors to both clean up the spill and pickup the dirty dishes. We study this aspect of generalization: combining knowledge from multiple tasks.

Combining knowledge from multiple tasks is challenging because it is not clear how to synthesize either the behavioral policies or the value functions learned during training. This challenge is exacerbated when an agent also needs to generalize to novel appearances and environment configurations. Returning to our example, our robot might need to additionally generalize to both novel dirty dishes and to novel arrangements of chairs.

Successor features (SFs) and Generalized Policy Improvement (GPI) provide a mechanism to combine knowledge from multiple training tasks (Barreto et al., 2017, 2020). SFs are predictive

representations that estimate how much state-features (known as "cumulants") will be experienced given a behavior. By assuming that reward has a linear relationship between cumulants and a task vector, an agent can efficiently *compute* how much reward it can expect to obtain from a given behavior. If the agent knows multiple behaviors, it can leverage GPI to compute which behavior would provide the most reward (see Figure 4.2 for an example). However, SF&GPI commonly assume hand-designed cumulants and don't have a mechanism for generalizing to novel environment configurations.

Modular architectures are a promising method for generalizing to distributions outside of the training distribution (Goyal et al., 2020b; Madan et al., 2021). In chapter §3, I presented "FARM" and showed that learning multiple state modules enabled generalization to environments with unseen environment parameters (e.g. to larger maps with more objects).In this work, we hypothesize that modules can further be leveraged to discover state-features that are useful to predict.

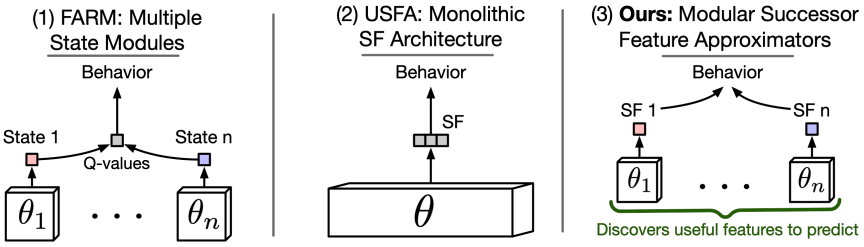


Figure 4.1: (1) FARM learns multiple state modules. This promotes generalization to novel environments. However, it has no mechanism for combining task solutions. (2) USFA learns a single monolithic architecture for predicing SFs and can combine task solutions. However, it relies on hand-designed state features and has no mechanism for generalization to novel environments. (3) We combine the benefits of both. We leverage modules for reward-driven discovery of state features that are useful to predict. These form the basis of their own predictive representations (SFs) and enables combining task solutions in novel environments.

We present "Modular Successor Feature Approximators" (MSFA), a novel neural network for discovering, composing, and transferring predictive knowledge and behavior via SF&GPI. MSFA is composed of a set of modules, which each learn their own state-features and corresponding predictive representations (SFs). **Our core contribution** is showing that an inductive bias for modularity can enable reward-driven discovery of state-features that are useful for zero-shot transfer with SF&GPI. We exemplify this with a simple state-feature discovery method presented in Barreto et al. (2018) where the dot-product between state-features and a task vector is regressed to environment reward. This method enabled transfer with SF&GPI in a continual learning setting but had limited success in the zero-shot transfer settings we study. While there are other methods for state-feature discovery, they add training complexity with mutual information objectives (Hansen et al., 2019) or meta-gradients (Veeriah et al., 2019). With MSFA, by adding *only an architectural bias for*

*modularity*, we discover state-features that (1) support zero-shot transfer competitive with hand-designed features, and (2) enable zero-shot transfer in visually diverse, procedurally generated environments. We are hopeful that our architectural bias can be leveraged with other discovery methods in future work.

## 4.2 Related Work on Generalization in RL

**Hierarchical RL** (HRL) is one dominant approach for combining task knowledge. The basic idea is that one can sequentially combine policies in time by having a "meta-policy" that sequentially activates "low-level" policies for protracted periods of time. By leveraging hand-designed or pre-trained low-level policies, one can generalize to longer instructions (Oh et al., 2017a; Corona et al., 2020), to new instruction orders (Brooks et al., 2021), and to novel subtask graphs (Sohn et al., 2021, 2022). We differ in that we focus on combining policies *concurrently* in time as opposed to sequentially in time. To do so, we develop a modular neural network for the SF&GPI framework.

**SFs** are predictive representations that represent the current state as a summary of the *successive* features to follow (see §4.3 for a formal definition). By combining them with Generalized Policy Improvement, researchers have shown that they can transfer behaviors across object navigation tasks (Borsa et al., 2019; Zhang et al., 2017; Zhu et al., 2017), across continuous control tasks (Hunt et al., 2019), and within an HRL framework (Barreto et al., 2019). However, these works tend to require hand-designed cumulants which are cumbersome to design for every new environment. In our work, we integrate SFs with Modular RL to facilitate reward-driven discovery of cumulants and improve successor feature learning.

**Modular RL** (MRL) (Russell and Zimdars, 2003) is a framework for generalization by combining value functions. Early work dates back to (Singh, 1992), who had a mixture-of-experts system select between separately trained value functions. Since then, MRL has been applied to generalize across robotic morphologies (Huang et al., 2020b), to novel task-robot combinations (Devin et al., 2017; Haarnoja et al., 2018), and to novel language instructions (Logeswaran et al., 2021). MSFA, is the first to integrate MRL with SF&GPI. This integration enables combining task solutions in novel environment configurations.

**Generalizing to novel environment configurations with modules**. Goyal et al. (2020b) showed that leveraging modules to learn a *state function* improved out-of-distribution generalization. Carvalho et al. (2021a) showed that a modified attention mechanism led to strong generalization improvements with RL. MSFA differs from both in that it employ modules for learning *value functions* in the form of SFs. This enables a principled way to compose task knowledge while additionally generalizing to novel environment configurations.

## 4.3 Problem Setting and Background

We study a reinforcement learning agent's ability to transfer knowledge between tasks in an environment. During training, the experiences $n_{\texttt{train}}$ tasks $\mathbb{M}_{\texttt{train}} = \{\mathcal{M}_i\}_{i=1}^{n_{\texttt{train}}}$, sampled from a training distribution $p_{\texttt{train}}(\mathcal{M})$. During testing, the agent is evaluated on $n_{\texttt{test}}$ tasks, $\{\mathcal{M}_i\}_{i=1}^{n_{\texttt{test}}}$, sampled from a testing distribution $p_{\texttt{test}}(\mathcal{M})$. Each task $\mathcal{M}_i$ is specified as Partially Observable Markov Decision Process (POMDP), $\mathcal{M}_i = \langle \mathcal{S}^e, \mathcal{A}, \mathcal{X}, R, p, f_x \rangle$. Here, $\mathcal{S}^e$, $\mathcal{A}$ and $\mathcal{X}$ are the environment state, action, and observation spaces. $p(\cdot|s_t^e, a_t)$ specifies the next-state distribution based on taking action $a_t$ in state $s_t^e$, and $f_x(s_t^e)$ maps the underlying environment state to an observation $x_t$. We focus on tasks where rewards are parameterized by a task vector $w$, i.e. $r_t^w = R(s_t^e, a_t, s_{t+1}^e, w)$ is the reward obtained for transition $(s_t^e, a_t, s_{t+1}^e)$ given task vector $w$. Since this is a POMDP, we need to learn a state function that maps histories to agent state representations. We do so with a recurrent function: $s_t = s_\theta(x_t, s_{t-1}, a_{t-1})$. Given this learned state, we want to obtain a behavioral policy $\pi(s_t)$ that best maximises the expected reward it will obtain when taking an action $a_t$ at a state $s_t$: $Q_t^{\pi,w} = Q^{\pi,w}(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t^w\right]$.

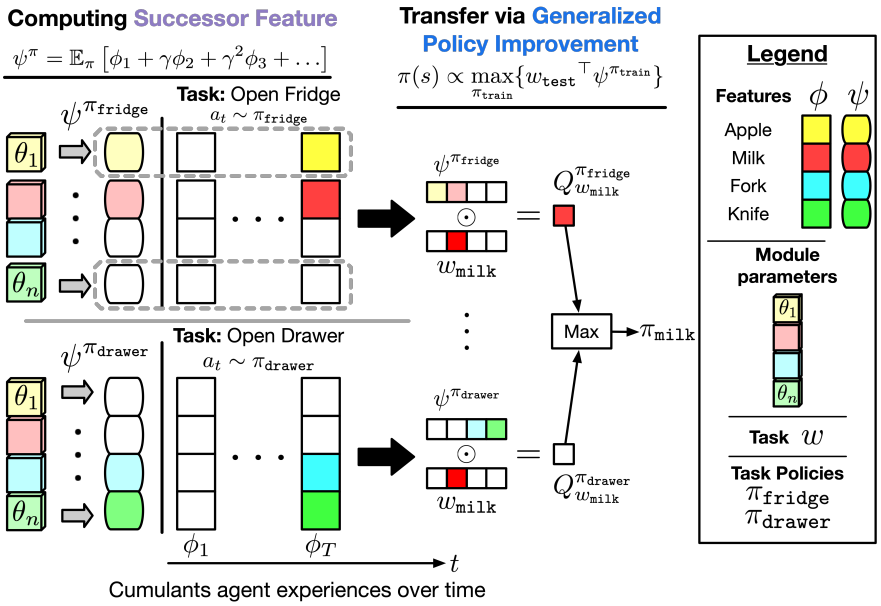


Figure 4.2: **High-level diagram of how MSFA can be leveraged for transfer with SF&GPI**. During training, we can have the agent learn policies for tasks—e.g. "open drawer" and "open fridge". Each task leads the agent to experience different aspects of the environment—e.g. a "fork" during "open drawer" or an "apple" during "open fridge". We can leverage MSFA to have different modules learn different "cumulants", $\phi$, and SFs, $\psi$. For example, module 1 ($\theta_1$) can estimate SFs for apple cumulants. Module SFs are combined to form the SF for a policy. When the agent wants to transfer its knowledge to a test task—e.g., "get milk"—it can compute Q-values for that task as a dot-product with the SFs of each training task. The highest Q-value is then used to select actions.

**Transfer with SF&GPI.** In order to leverage SFs (Barreto et al., 2017), one assumes an agent has access to state features known as "cumulants", $\phi_t = \phi(s_t, a_t, s_{t+1})$. Given a behavioral policy $\pi(a|s)$, SFs are a type of value function that use $\phi_t$ as pseudo-rewards:

$$\psi_t^\pi = \psi^\pi(s_t, a_t) = \mathbb{E}_\pi \left[ \sum_{i=0}^{\infty} \gamma^i \phi_{t+i} \right] \tag{4.1}$$

If reward is (approximately) $r_t^w = \phi_t^\top w$, then action-values can be decomposed as $Q_t^{\pi,w} = {\psi_t^\pi}^\top w$. This is interesting because it provides an easy way to *reuse* task-agnostic features $\psi_t^\pi$ for new tasks.

We can re-use the SFs we've learned from training tasks $\mathbb{M}_{\texttt{train}}$ for transfer with GPI. Assume we have learned (potentially optimal) policies $\{\pi_i\}_{i=1}^{n_{\texttt{train}}}$ and their corresponding SFs $\{\psi^{\pi_i}(s,a)\}_{i=1}^{n_{\texttt{train}}}$. Given a test task $w_{\texttt{test}}$, we can obtain a new policy with GPI in two steps: (1) compute Q-values using the training task SFs (2) select actions using the highest Q-value. This operation is summarized as follows:

$$\pi(s_t; w_{\texttt{test}}) \in \arg\max_{a \in \mathcal{A}} \max_{i \in \{1,\ldots,n_{\texttt{train}}\}} \{Q_t^{\pi_i, w_{\texttt{test}}}\} = \arg\max_{a \in \mathcal{A}} \max_{i \in \{1,\ldots,n_{\texttt{train}}\}} \{{\psi_t^{\pi_i}}^\top w_{\texttt{test}}\} \tag{4.2}$$

This is useful because the GPI theorem states that $\pi$ will perform as well as all of the training policies, i.e. that $Q^{\pi,w_{\texttt{test}}}(s,a) \geq \max_i Q^{\pi_i,w_i}(s,a) \forall (s,a) \in (\mathcal{S} \times \mathcal{A})$ (Barreto et al., 2017).

SF&GPI enable transfer by exploiting structure in the RL problem: a policy that maximizes a value function is guaranteed to perform at least as well as the policy that defined that value function. However, SF&GPI relies on combining a fixed set of SFs. Another form of transfer comes from "Universal Value Function Approximators" (UVFAs) (Schaul et al., 2015), which add the task-vector $w$ as a parameter to a Q-approximator parameterized by $\theta$, $Q_\theta(s,a,w)$. If $Q_\theta$ is smooth with respect to $w$, then $Q_\theta$ should generalize to test tasks nearby to train tasks in task space. Borsa et al. (2019) showed that one could combine the benefits of both with "Universal Successor Feature Approximators". Since rewards $r^w$, and therefore task vectors $w$, reference deterministic task policies $\pi_w$, one can parameterize successor feature approximators with task-vectors $\tilde{\psi}^{\pi_w} = \tilde{\psi}^w \approx \psi_\theta(s,a,w)$. However, USFA assumed hand-designed cumulants. We introduce an architecture for reward-driven discovery of cumulants and improved function approximation of universal successor features.

## 4.4 Modular Successor Feature Approximators

We propose a new architecture *Modular Successor Feature Approximators* (MSFA) for approximating SFs, shown in Figure 4.3. Our hypothesis is that learning cumulants and SFs with modules

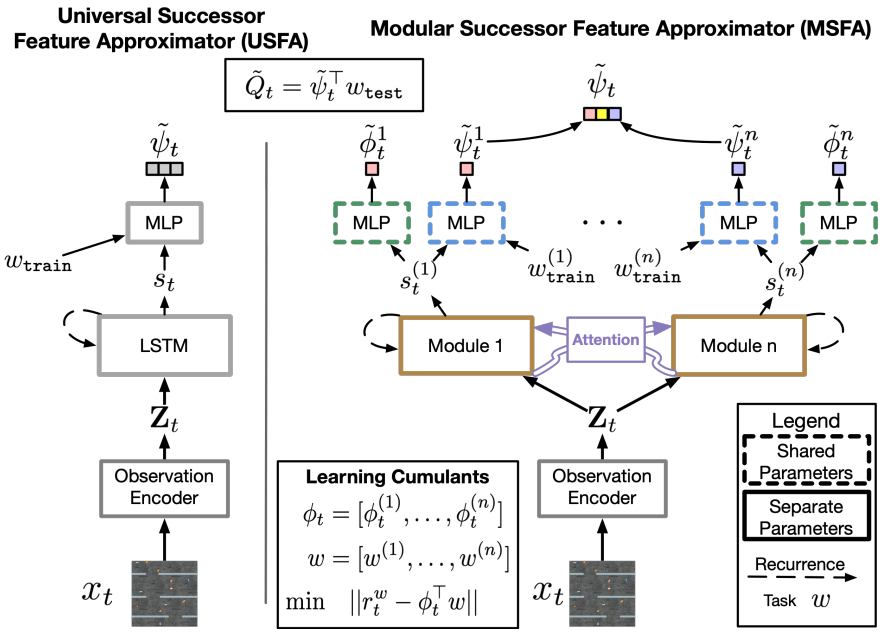


Figure 4.3: **Left: Universal Successor Feature Approximator (USFA)** learns a single, monolithic successor feature estimator that uses **hand-designed** cumulants. **Right: Modular Successor Feature Approximator (MSFA)** learns a set of successor feature modules, each with their own functions for (a) updating module-state, (b) computing cumulants, and (c) estimating successor features. Modules then share information with an attention mechanism. We hypothesize that isolated module computations facilitate learning cumulants that suppot generalization with GPI.

improves zero-shot composition of task knowledge with SF&GPI. MSFA accomplishes this by learning $n$ state modules $\{s_{\theta_k}\}_{k=1}^n$ that evolve with independent parameters $\theta_k$ and have sparse inter-module information flow. MSFA then produces modular cumulants $\{\tilde{\phi}_t^{(k)}\}_{k=1}^n$ and SFs $\{\tilde{\psi}_t^{\pi,k}\}_{k=1}^n$ by having their computations depend **only** on individual module-states. For example, a cumulant may correspond to information about apples, and would be a function **only** of the module representing state information related to apples. This is in contrast to prior work, which learns a single monolithic prediction module for computing cumulants and SFs (see Figure 4.3).

The rest of section is structured as follows. In section §4.4.1, we derive Modular Successor Feature Learning within the Modular RL framework. We then describe our architecture, MSFA, for learning modular successor features in §4.4.2. In §4.4.3, we describe how to generate behavior with MSFA. Finally, we describe the learning algorithm for MSFA in section §4.4.4.

### 4.4.1 Modular Successor Feature Learning

Following the Modular RL framework (Russell and Zimdars, 2003), we assume that reward has an additive structure $R(s_t, a_t, s_{t+1}) = \sum_k R^k(s_t, a_t, s_{t+1}) = \sum_k R_t^{(k)}$, where $R_t^{(k)}$ is the reward of

the $k$-th module. We enforce that every module decomposes reward into an inner-product between its own task-description $w^{(k)}$ and task-agnostic cumulants $\phi^{(k)} \in \mathbb{R}$: $R_t^{(k)} = \phi_t^{(k)} \cdot w^{(k)}$. Here, we simply break up the task vector into $n$ pieces so individual modules are responsible for subsets of the task vector. This allows us to decompose the action-value function as

$$Q^{\pi}(s_t, a_t, w) = \sum_{k=1}^{n} Q^{\pi,k}(s_t, a_t, w^{(k)}) = \sum_{k=1}^{n} \psi^{\pi,k}(s_t, a_t) \cdot w^{(k)} \tag{4.3}$$

where we now have *modular SFs* $\{\psi^{\pi,k}(s,a)\}_{k=1}^{n}$ (see the Appendix for a derivation). Rather than hand-designing modules or cumulants, we aim to discover them from the environment reward signal.

### 4.4.2 Architecture

We learn a set of modules with states $\mathbb{S}_t = \{s_t^{(k)}\}_{k=1}^{n}$. They update at each time-step $t$ with the observation $x_t$, the previous module-state $s_{t-1}^{(k)}$, and information from other modules $A_\theta(s_{t-1}^{(k)}, \mathbb{S}_{t-1})$. Following prior work (Santoro et al., 2018; Goyal et al., 2020b; Carvalho et al., 2021a), we have $A_\theta$ combine transformer-style attention (Vaswani et al., 2017) with a gating mechanism (Parisotto et al., 2020) to enforce that inter-module interactions are sparse. Since $A_\theta$ is not the main contribution of this paper, we describe these computations in more detail with our notation in the Appendix. We summarize the high-level update below.

$$s_t^{(k)} = s_{\theta_k}(x_t, s_{t-1}^{(k)}, A_\theta(s_{t-1}^{(k)}, \mathbb{S}_{t-1})) \tag{4.4}$$

**We learn modular cumulants and SFs** by having sets of cumulants and SFs depend on individual module-states. Module cumulants depend on the module-state from the current and next time-step. Module SFs depend on the current module-state and on their subset of the task description. We summarize this below:

$$\tilde{\phi}_t^{(k)} = \phi_\theta(s_t^{(k)}, a_t, s_{t+1}^{(k)}) \qquad \tilde{\psi}_t^{w,k} = \psi_\theta(s_t^{(k)}, a_t, w^{(k)}) \tag{4.5}$$

We highlight that cumulants share parameters but differ in their input. This suggests that the key is not having cumulants and SFs with separate parameters but that they are functions of sparse subsets of state (rather than all state information). We show evidence for this hypothesis in Figure 4.6.

We concatenate module-specific cumulants and SFs to form the final outputs: $\tilde{\phi}_t = \left[\tilde{\phi}_t^{(1)}, \ldots, \tilde{\phi}_t^{(n)}\right]$ and $\tilde{\psi}_t^{w} = \psi_\theta(s_t, a_t, w) = \left[\tilde{\psi}_t^{w,1}, \ldots, \tilde{\psi}_t^{w,n}\right]$. Note that cumulants, are only used during learning, update with the module-state from the next time-step.

### 4.4.3 Behavior

During **training**, actions are selected in proportion to Q-values computed using task SFs as $\pi(s_t, w) \propto \tilde{Q}(s_t, a, w) = \psi_\theta(s_t, a, w)^\top w$. In practice we use an epsilon-greedy policy, though one can use other choices. During **testing**, we compute policies with GPI as $\pi(s_t, w_{\texttt{test}}) \in \arg\max_a \max_{z \in \mathbb{M}_{\texttt{train}}} \{\psi_\theta(s_t, a, z)^\top w_{\texttt{test}}\}$, where $\mathbb{M}_{\texttt{train}}$ are train task vectors.

### 4.4.4 Learning Algorithm

MSFA relies on three losses. The first loss, $\mathcal{L}_Q$, is a standard Q-learning loss, which MSFA uses to learn optimal policies for the training tasks. The main difference here is that MSFA uses a particular parameterization of the Q-function $Q^{\pi_w, w}(s, a) = \psi^{\pi_w}(s, a)^\top w$. The second loss, $\mathcal{L}_\psi$, is an SF learning loss, which we use as a regularizer to enforce that the Q-values follow the structure in the reward function $r_t^w = \phi_t^\top w$. For this, we again apply standard Q-learning but using SFs as value functions and cumulants as pseudo-rewards. The final loss, $\mathcal{L}_\phi$ is a loss for learning cumulants that grounds them in the environment reward signal. The losses are summarised as follows

$$\mathcal{L}_Q = ||r_t + \gamma \psi_\theta(s_{t+1}, a', w)^\top w - \psi_\theta(s_t, a_t, w)^\top w||^2 \tag{4.6}$$

$$\mathcal{L}_\psi = ||\tilde{\phi}_t + \gamma \psi_\theta(s_{t+1}, a', w) - \psi_\theta(s_t, a_t, w)||^2 \tag{4.7}$$

$$\mathcal{L}_\phi = ||r_t^w - \tilde{\phi}_t^\top w||^2 \tag{4.8}$$

where $a' = \arg\max_a \psi_\theta(s_{t+1}, a, w)^\top w$. Selecting the next action via the combination of all modules ensures they individually convergence to optimal values (Russell and Zimdars, 2003). The final loss is $\mathcal{L} = \alpha_Q \mathcal{L}_Q + \alpha_\psi \mathcal{L}_\psi + \alpha_\phi \mathcal{L}_\phi$.

## 4.5 Experiments

We study generalization when training behaviors must be combined concurrently in time in the presence of novel object appearances and layouts. The need to combine training behaviors tests how well MSFA can leverage SF&GPI. Generalization to novel object appearances and layouts tests how well MSFA's modular construction supports generalization to novel environment configurations.

**Baselines.** (1) **Universal Value Function Approximator (UVFA)** (Schaul et al., 2015), which takes the task as input: $Q_\theta(s, w)$. This comparison shows shows the transfer benefits of SF&GPI. (2) **UVFA with Feature-Attending Recurrent Modules (UVFA+FARM)** instead takes state-factors as input $Q_\theta(s^{(1)}, \ldots, s^{(n)}, w)$. Each state-factor $s^{(k)}$ is the output of a FARM module. (3) **Modular Value Function Approximator(MVFA)** is an adaptation of (Haarnoja et al., 2018) where modules

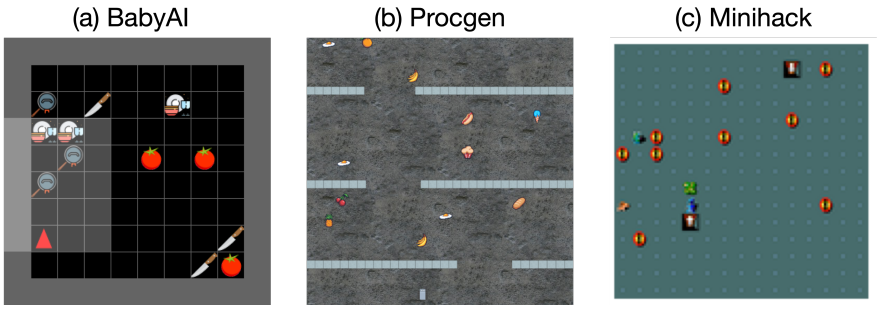


Figure 4.4: **We study an agent's ability to combine task knowledge in three environments.** (a) In BabyAI, an agent learns to pick up one object type at a time during training. During testing, the agent must pickup combinations of object types while avoiding other object types. This is the setting used by USFA which assumed **hand-designed** cumulants. (b) In Procgen, we study extending this form of generalization to a visually diverse, procedurally generated environment. (c) In Minihack, we go beyond combining object navigation skills. Here, an agents needs to combine (1) avoiding teleportation traps, (2) avoiding monsters, and (3) partial visibility around the agent.

learn individual Q-values $Q_\theta^{(i)}(s^{(i)}, w^{(i)})$. Comparing to UVFA+FARM and MVFA enables us to study the benefits of leveraging modules for learning value functions in the form of SFs. (4) **Universal Successor Function Approximator (USFA)** (Borsa et al., 2019) leverages a single monolithic function for successor features with **hand-designed cumulants**. USFA is an upper-bound baseline that allows us to test the quality of cumulants and successor features that MSFA learns. We also test a variant of USFA with learned cumulants, **USFA-Learned-$\phi$**, which shows how the architecture degrades without oracle cumulants.

**Implementation.** We implement the state modules of MSFA with FARM modules (Carvalho et al., 2021b). For UVFA and USFA, we learn a state representation with an LSTM (Hochreiter and Schmidhuber, 1997). We implement all $\phi$, $\psi$, and $Q$ functions with Mutli-layer Perceptrons. We train UVFA and UVFA+FARM with n-step Q-learning (Watkins and Dayan, 1992). When learning cumulants, USFA and MSFA have the exact same losses and learning alogirthm. They both learn Q-values and SFs with n-step Q-learning. We use $n = 5$. When not learning cumulants, following (Borsa et al., 2019), USFA only learns SFs with n-step Q-learning. All agents are built with JAX (Bradbury et al., 2018) using the open-source ACME codebase (Hoffman et al., 2020) for reinforcement learning.

### 4.5.1 Combining object navigation task knowledge

MSFA learns modular functions for computing $\phi$ and estimating $\psi$. We hypothesize that this facilitates learning cumulants that respond to different aspects of the environment (e.g. to different object categories). This leads to the following research questions. **R1**: Can we recover prior

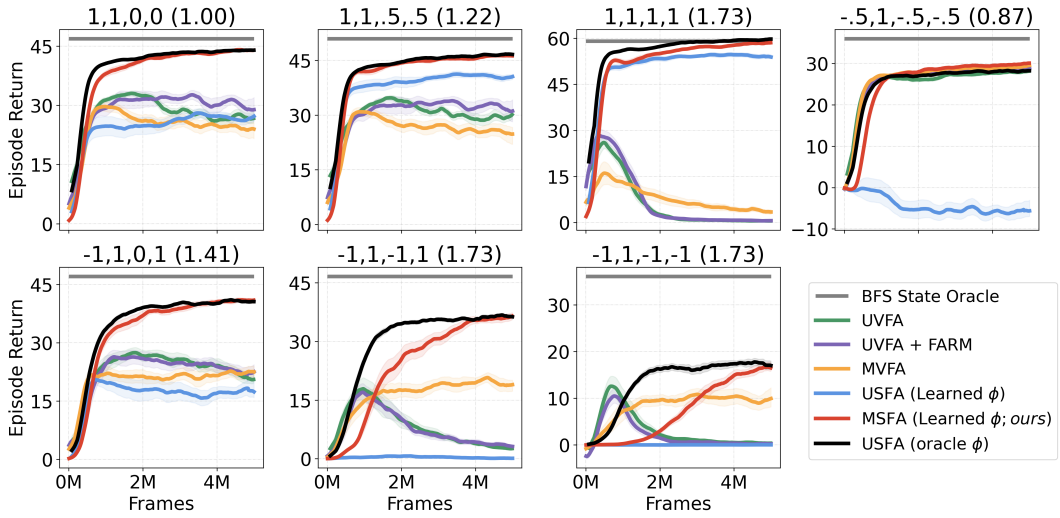


Figure 4.5: **MSFA matches USFA, which has hand-designed cumulants**. We show mean and standard error generalization episode return across 10 runs. We put a task's L2 distance to the closest train task in parenthesis. USFA best generalizes to novel combinations of picking up and avoiding objects. Once USFA learns cumulants, its performance degrades significantly. UVFA-based methods struggle as more objects should be avoided or tasks are further in distance to train tasks.

generalization results that relied on hand-designed cumulants for different object categories? **R2**: How important is it to learn modular functions for $\phi$ and $\psi$? **R3**: Without GPI, do learning modular functions for $\phi$ and $\psi$ still aid in generalization?

**Setup.** We implement a simplified version of the object navigation task of (Borsa et al., 2019) in the (Chevalier-Boisvert et al., 2019) BabyAI environment. The environment consists of 3 instances of 4 object categories. **Observations** are partial and egocentric. **Actions**: the agent can rotate left or right, move forward, or pickup an object. When it picks up an object, following (Borsa et al., 2019), the object is respawned somewhere on the grid. **Task** vectors lie in $w \in \mathbb{R}^4$ with training tasks being the standard unit vectors. For example, $[0, 1, 0, 0]$ specifies the agent must obtain objects of category 2. **Generalization** tasks are linear combinations of training tasks. For example, $[-1, 1, -1, 1]$ specifies the agent must obtain categories 2 and 4 while *avoiding* categories 1 and 3. Borsa et al. (2019) showed that USFA could generalize with *hand-designed* cumulants that described whether an object was picked up. We describe challenges for this task in detail in the Appendix.

**R1: MSFA is competitive with USFA, which uses oracle $\phi$.** Figure 4.5 shows USFA with a similar generalization trend to (Borsa et al., 2019). Tasks get more challenging as they are further from train tasks or involve avoiding more objects. For simply going to combinations of objects, USFA-Learned-$\phi$ does slightly worse than MSFA. However, with more objects to avoid, all methods except MSFA (including USFA-Learned-$\phi$) degrades significantly. For comparison, we

show performance by an oracle bread-first-search policy with access to ground-truth state (**BFS State Oracle**). All methods have room for improvement when objects must be avoided. In the appendix, we present heat-maps for how often object categories were picked up during different tasks. We find that MSFA most matches USFA, while USFA-Learned-$\phi$ commonly picks up all objects regardless of task.

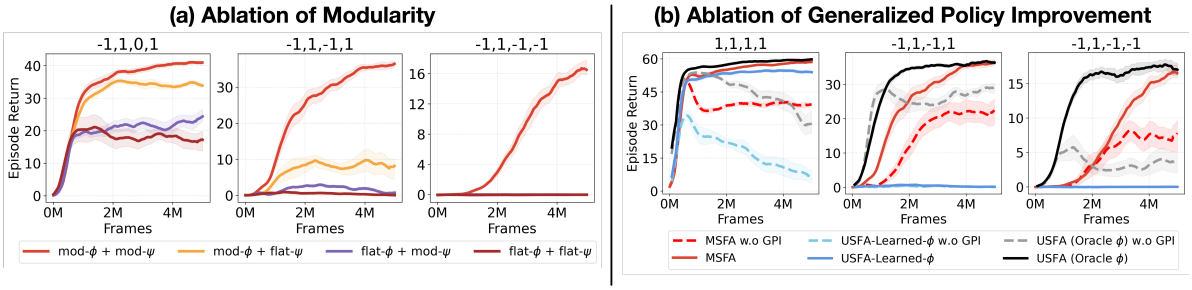


Figure 4.6: **Learning modular $\phi$ and $\psi$ is key to generalization and improves generalization even without GPI.** We show mean and standard error generalization episode return across 10 runs. (a) We ablate having modular functions for $\phi_\theta$ and $\psi_\theta$. Generalization results degrade significantly. (b) We ablate leveraging GPI for generalization from all SF-based methods. MSFA without GPI can outperform both USFA-Learned-$\phi$ with GPI and USFA without GPI. This shows the utility of modularity for generalization.

**R2: Learning modular $\phi$ and $\psi$ functions is critical for generalization.** Learning an entangled function corresponds to learning a monolithic function for $\psi$ or $\phi$ where we concatenate module-states, e.g. $\tilde{\psi}^w_t = \psi_\theta(s^{(1)}_t, \ldots, s^{(n)}_t, a, w)$. Modular functions correspond to equation 4.5. Figure 4.6 (a) shows that without modular functions for $\phi$ **and** $\psi$, performance severely degrades. This also highlights that a naive combination of USFA+FARM—with entangled functions for $\phi$ and $\psi$—does not recover our generalization performance.

**R3: Modularity alone improves generalization.** For all SF-based methods, we remove GPI and select actions with a greedy policy: $\pi(s) = \arg\max_a \psi_\theta(s_t, a, w)^\top w$. Figure 4.6 (b) shows that GPI is critical for generalization with USFA as expected. USFA-Learned-$\phi$ benefits less from GPI (presumably because of challenges in learning $\phi$). Interestingly, MSFA can generalize relatively well without GPI, sometimes doing better than USFA without GPI.

### 4.5.2 Combining object navigation task knowledge with novel appearances and environment configurations

Beyond generalizing to combinations of tasks, agents will need to generalize to different layouts and appearances of objects. **R4**: Can MSFA enable combining task knowledge in a visually diverse, procedurally generated environment?

**Setup.** We leverage the "Fruitbot" environment within ProcGen (Cobbe et al., 2020). Here, an agent controls a paddle that tries to obtain certain categories of objects while avoiding others. When the agent hits a wall or fence, it dies and the episode terminates. If the agent collects a non-task object, nothing happens. **Observations** are partial. **Actions**: At each time-step the agent moves one step forward and can move left or right or shoot pellets to open fences. **Training and generalization tasks** follow the same setup as §4.5.1.

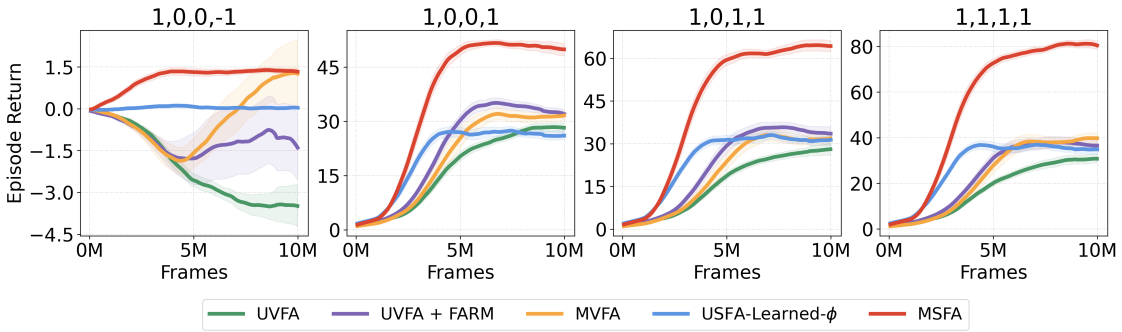


Figure 4.7: **MSFA is able to combine task knowledge in a visually diverse, procedurally generated ProcGen environment.** We find that no method is able to do well when there are objects to avoid ($w = [1, 0, 0, -1]$) in this setting (see text for more). However, as more objects need to be collected MSFA best generalizes (10 runs).

**R4: MSFA enables combining task knowledge of object navigation tasks in a visually diverse, procedurally generated environment.** Figure 4.7 shows that when an agent has to generalize to collecting more objects, modular architectures generalize best, with MSFA doing best. When objects have to be avoided, we see that no architecture does well, though MSFA tends to do better. We observe that avoiding objects leads agents to hit walls. Since the agent always moves forward at each time-step in Fruibot, this makes avoiding objects a particularly challenging type of generalization.

## 4.6 Discussion and Conclusion

We have presented "Modular Successor Feature Approximators", a modular neural network for learning cumulants and SFs produced by their own modules. We first showed that MSFA is competitive with prior object navigation generalization results that relied on hand-designed cumulants (§4.5.1). Afterwards, we showed that MSFA improves an agent's ability to combine task knowledge in a visually diverse, procedurally generated environment (§4.5.2). We also show that MSFA can combine solutions of heterogeneous tasks. Our ablations show that learning modular cumulants and SFs is critical for generalization with GPI.

We compared MSFA to (1) USFA, a monolithic architecture for learning cumulants and SFs; (2) FARM, an architecture which learns multiple state-modules but combines them with a monolithic Q-value function. Our results show that when learning cumulants, MSFA improves generalization with SF&GPI compared to USFA. Additionally, without GPI, MSFA as an architecture improves generalization as compared to both FARM and USFA.

**Limitations**. While we demonstrated reward-driven discovery of cumulants for transfer with SF&GPI, we focused on relatively simple task encodings. Future work can extend this to more expressive encodings such as language embeddings. Another limitation is that we did not explore more sophisticated state-feature discovery methods such as meta-gradients (Veeriah et al., 2019). Nonetheless, we think that MSFA provides an important insight for future work: that modularity is a simple but powerful inductive bias for discovering state-features that enable zero-shot transfer with SF&GPI.

**Future directions**. SFs are useful for exploration (Janz et al., 2019; Machado et al., 2020); for discovering and combining options (Barreto et al., 2019; Hansen et al., 2019); for transferring policies across environments (Zhang et al., 2017); for improving importance sampling (Fujimoto et al., 2021); and for learning policies from other agents (Filos et al., 2021). We hope that future work can leverage MSFA for improved state-feature discovery and SF-learning in all of these settings.

# CHAPTER 5

# Discovering Representations for Transfer with Successor Features and the Deep Option Keyboard

The Option Keyboard (OK) was recently proposed as a method for transferring behavioral knowledge across tasks. OK transfers knowledge by adaptively combining subsets of known behaviors using Successor Features (SFs) and Generalized Policy Improvement (GPI). However, it relies on hand-designed state-features and task encodings. In this work, we propose the Deep Option Keyboard (Deep OK), which enables transfer with *discovered* state-features and task encodings. To enable discovery, we propose the Categorical Successor Feature Approximator (CSFA), a novel learning algorithm for estimating SFs while jointly discovering state-features and task encodings. With Deep OK and CSFA, we achieve the first demonstration of transfer with SFs in a challenging 3D environment where all the necessary representations are discovered. We first compare CSFA against other methods for approximating SFs and show that only CSFA discovers representations compatible with SF&GPI at this scale. We then compare Deep OK against transfer learning baselines and show that it transfers most quickly to long-horizon tasks.

## 5.1 Introduction

Consider a household robot that learns tasks for interacting with objects such as finding and moving them around. When this robot is deployed to a house and needs to perform combinations of these tasks, collecting data for reinforcement learning (RL) will be expensive. Thus, ideally this robot can effectively *transfer* its knowledge to efficiently learn these novel tasks with minimal interactions in the environment. We study this form of transfer in Deep RL.

One promising method for transfer is the Option Keyboard (OK)(Barreto et al., 2019, 2020), which transfers to new tasks by adaptively combining subsets of known behaviors (see Figure 5.1). OK combines known behaviors by leveraging Successor Features (SFs) and Generalized Policy Improvement (GPI) (Barreto et al., 2017, 2018). SFs are predictive representations for behaviors.

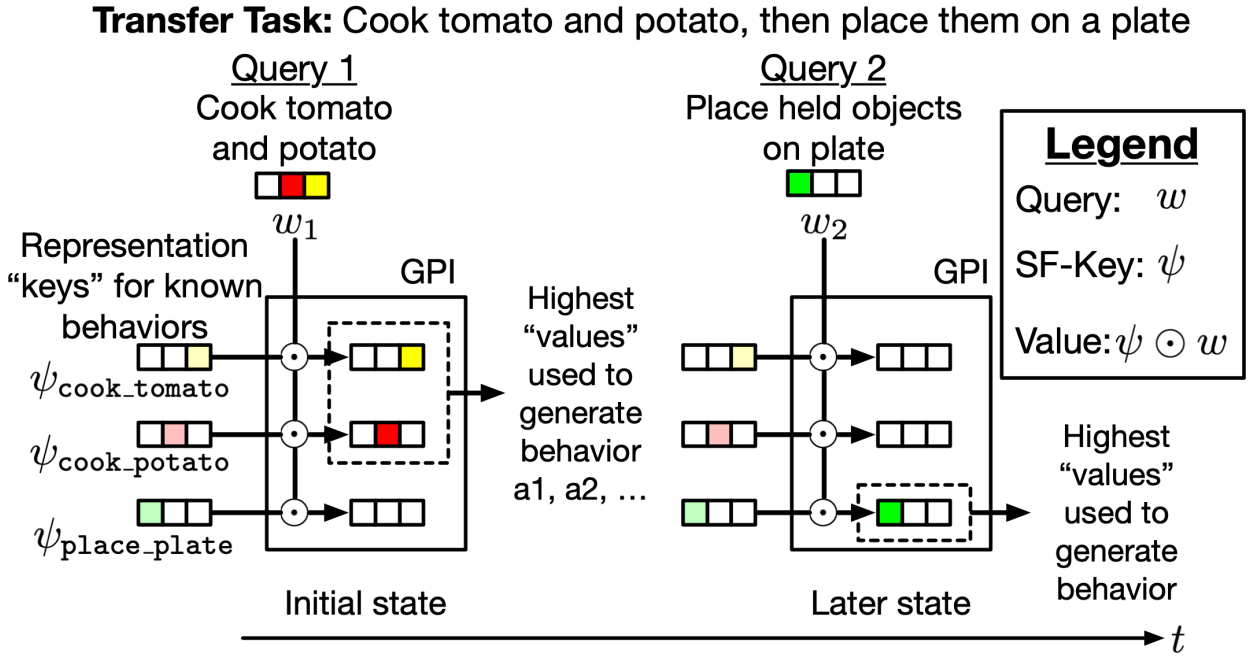


Figure 5.1: **Diagram of transfer with the Option Keyboard (OK)**. OK uses SF-based "keys" to represent behaviors by how much of some feature (known as a "cumulant") they obtain. OK *dynamically* selects from behaviors with GPI by generating preference "queries" over these features. The "value" of each behavior is then computed with a dot-product and the highest value is used to generate behavior. One limitation of OK is that it **hand-designs** the features that define queries and SFs. Here, we study the problem of transferring with OK while discovering all necessary representations.

They represent behaviors with estimates of how much state-features (known as "cumulants") will be experienced given that behavior. GPI can be thought of as a query-key-value system that, given feature preferences, can be used to select from behaviors that obtain those features.

While OK is a promising transfer method, it relies on hand-designed representations for the "cumulants" and task feature-preferences (i.e. task encodings). There is work that has discovered either cumulants or task encodings (Barreto et al., 2017, 2018; Filos et al., 2021; Carvalho et al., 2023a). However, either (a) they only showed transfer with SF&GPI using a *fixed* (not dynamic) transfer query, (b) they only demonstrated results in simple grid-worlds where an agent combines short-horizon "goto" tasks (e.g. goto A and B), or (c) they leveraged *separate* networks for the SFs of each task-policy and for each task-encoding. This means parameter count scales with the number of tasks and limits representation re-use across tasks.

Ideally, we can transfer with a dynamic query while leveraging *discovered* representations for cumulants and feature preferences. Further, a transfer method should scale to sparse-reward long-horizon tasks such as those found in complex 3D environments. However, jointly discovering cumulants and feature preferences while estimating SFs is challenging in this setting. First, jointly learning cumulants and SFs involves estimating boot-strapped returns over a non-stationary target with high-variance and a shifting magnitude (Barreto et al., 2018). To maximize knowledge sharing across tasks, we can *share* our SF-estimator and task encoder across tasks. However, no work has yet achieved transfer results doing so.

To transfer with a dynamic query that leverages discovered representations while sharing functions across tasks, we propose two novel methods, the *Deep Option Keyboard (Deep OK)* and the *Categorical Successor Feature Approximator (CSFA)*. Deep OK leverages CSFA to learn SF-estimates over discovered cumulants and task-preferences in a pretraining phase. Afterwards, in a finetuning phase, Deep OK learns to generate dynamic queries which are linear combinations of CSFA-discovered task-preferences. CSFA addresses challenges with estimating a non-stationary return by estimating SFs with a variant of the categorical two-hot representation introduced by MuZero (Schrittwieser et al., 2020). We discretize the space of cumulant-return values into bins and learns a probability mass function (pmf) over them. Modelling cumulant-returns with a pmf is more robust to outliers and can better accomodate a shifting magnitude. In contrast, standard methods estimate SFs with regression of a single point-estimate (Barreto et al., 2017; Borsa et al., 2019) which is succeptible to outliers and varying scales (Raffin et al., 2021).

We study Deep OK and CSFA in Playroom (Abramson et al., 2020), a challenging 3D environment with high-dimensional pixel observations and long-horizon tasks defined by sparse rewards. Prior work on transfer with SF&GPI has mainly focused on transfer in simpler 2D environments (Abdolshah et al., 2021; Barreto et al., 2017, 2018; Filos et al., 2021; Carvalho et al., 2023a). While Borsa et al. (2019) studied transfer in a 3D environment, they relied on hand-designed cumu-

lants and task encodings (Borsa et al., 2019) and only transferred to combinations of "Goto" tasks. We discover cumulants and task encodings while studying transfer to combinations of long-horizon, sparse-reward "Place near" tasks.

**Contributions**. (1) We propose the Deep Option Keyboard, a novel method that transfers to novel taks with SF&GPI using a dynamic query, discovered representations, and a task encoder and SF-approximator that are shared across tasks. (2) To enable discovery when sharing a task encoder and SF-approximator cross tasks, we propose a novel learning algorithm, the Categorical Successor Feature Approximator. (3) We present the first demonstration of transfer with successor features in a complex 3D environment where all the necessary representations are discovered.

## 5.2 Related work on Transfer in Deep RL

Several avenues exist to transfer knowledge in Deep RL. We can transfer an agent's representations (how they represent situations), their control policy (how they act in situations), or their value function (how they evaluate situations). To transfer representations, one can learn a mapping from source domains to target domains (Taylor et al., 2007), learn disentangled representations (Higgins et al., 2017; Watters et al., 2019), or learn a modular architecture (Rusu et al., 2016; Fernando et al., 2017). To transfer a control policy, some methods *distill* knowledge from a source policy to a target policy (Rusu et al., 2015), others exploit *policy improvement* (Bellman, 2010), and a third set transfer low-level policies by learning a meta-controller (Sutton et al., 1999b). Finally, to transfer value functions, some approaches learn universal value functions (Schaul et al., 2015) and others exploit SFs (Barreto et al., 2020). Below we review approaches most closely related to ours.

**Transferring policies**. One strategy to transfer a policy is to leverage multi-task RL (MTRL) training where you learn and transfer a goal-conditioned policy (Oh et al., 2017a). Another stratgy is to distill knowledge from one policy to another, as Distral does (Teh et al., 2017). Distral works by learning two policies: one goal-conditioned policy and another goal-agnostic "centroid policy". The action-likelihoods of each policy are then distilled into the other by minimizing KL-divergences. Distral has strong performance in multi-task settings but it relies on the utility of a "centroid" policy for sharing knowledge across tasks. When we transfer to *longer* horizon tasks with sparse rewards, neither MTRL nor Distral may provide a policy with good jumpstart performance (Zhu et al., 2020). In this work, we study jumpstart performance and exploit successor features (SFs) with generalized policy improvement (GPI) (Barreto et al., 2017).

**SFs** are useful because they enable computing of action-values for new task encodings (Barreto et al., 2017). When combined with GPI, prior work has shown strong zero-shot or few-shot transfer to combinations of tasks. To accomplish this, GPI can evaluate known SFs with a "query" transfer task encoding. SF&GPI relies on "cumulants" (which SFs predict returns over) and task encodings

| Method | disc. $\phi$ | disc. $w$ | query | share $w_\theta$ | share $\psi_\theta$ | 3D | transfer |
|---|---|---|---|---|---|---|---|
| Barreto et al. (2017) | ✓ | ✓ | static | ✗ | ✗ | ✗ | goto-n |
| Zhu et al. (2017)$^\beta$ | ✓ | ✓ | static | ✗ | ✗ | ✓ | goto-n |
| Barreto et al. (2018) | ✓ | ✗ | static | ✗ | ✗ | ✓ | goto-c |
| Filos et al. (2021)$^\beta$ | ✓ | ✓ | static | ✗ | ✗ | ✗ | goto-c |
| Borsa et al. (2019) | ✗ | ✗ | static | ✗ | ✓ | ✓ | goto-c |
| Carvalho et al. (2023a) | ✓ | ✗ | static | ✗ | ✓ | ✗ | goto-c |
| Barreto et al. (2019) | ✗ | ✗ | **dynamic** | ✗ | ✗ | ✗ | goto-c |
| Deep OK (ours) | ✓ | ✓ | **dynamic** | ✓ | ✓ | ✓ | **place-c** |

Table 5.1: **Related methods for transfer with SF&GPI**. Deep OK is the first method to transfer with a dynamic query, discover cumulants $\phi$ and task encodings $w$, while sharing a task encoder $w_\theta$ and SF approximator $\psi_\theta$ across tasks. Each of these is important to transfer with SF&GPI in a large-scale multi-task setting. Together, this enables the first SF method to transfer to combinations of long-horizon place tasks in a 3D environment with discovered $\phi$ and $w$. Note: $\beta$ refers to methods which learn from demonstration data, which we do not. goto-n is "goto new goal state", goto-c is "goto object combo", and place-c is "place object combo".

that respect a dot-product relationship. Prior work has had one of three limitations. Some work has discovered representations but only shown transfer with a *static* transfer query and did not share SF-estimators or task-encoders across tasks (Barreto et al., 2017; Zhu et al., 2017; Barreto et al., 2017; Filos et al., 2021). Other work has shared SF-estimators across tasks but exploited hand-designed task encodings with static GPI queries (Borsa et al., 2019; Carvalho et al., 2023a). The Option Keyboard (Barreto et al., 2020) transferred using a *dynamic* query; however, they hand-designed cumulants and task encodings and didn't share functions across tasks. In this work, we present the Deep Option Keyboard, where we address all three limitations. We transfer with a dynamic query, discover cumulants and task encodings, and learn both a task-encoder and SF-estimator that are shared across tasks. Additionally, prior work has only studied transfer to combinations of short-horizon "go to" tasks whereas we include longer horizon "place" tasks and do so in a 3D environment. We summarize these differences in Table 5.1.

## 5.3 Background

We study an RL agent's ability to transfer knowledge from a set $n_\kappa$ training tasks $\mathbb{T}_{\texttt{train}} = \{\kappa_1, \ldots, \kappa_{n_\kappa}\}$ to a set of $n'_\kappa$ transfer tasks $\mathbb{T}_{\texttt{new}} = \{\kappa^{\texttt{n}}_1, \ldots, \kappa^{\texttt{n}}_{n'_\kappa}\}$. During training, tasks are sampled from distribution $p_{\texttt{train}}(\mathbb{T}_{\texttt{train}})$. At transfer, tasks are sampled from distribution $p_{\texttt{transfer}}(\mathbb{T}_{\texttt{new}})$. Each task is specified as a Partially Observable Markov Decision Process (POMDP, (Kaelbling et al., 1998)), $\mathcal{M}_i = \langle \mathcal{S}^e, \mathcal{A}, \mathcal{X}, R, p, f_x \rangle$, where $\mathcal{S}^e$, $\mathcal{A}$ and $\mathcal{X}$ are the environment state, action, and observation spaces. Rewards are parameterized by a task description $\kappa$, i.e. $r^\kappa_{t+1} = R(s^e_t, a_t, s^e_{t+1}, \kappa)$

is the reward for transition $(s^e_t, a_t, s^e_{t+1})$. When the agent takes action $a_t$ in state $s^e_t$, $s^e_{t+1}$ is sampled according to $p(\cdot|s^e_t, a_t)$, an observation $x_{t+1}$ is generated via $f_x(s^e_{t+1})$, and the agent gets reward $r^\kappa_{t+1}$. We assume the agent learns a recurrent state function that maps histories to agent state representations, $s_t = s_\theta(x_t, s_{t-1}, a_{t-1})$. Given this learned state, we aim to obtain a behavior policy $\pi(s_t, \kappa)$ that maximises the expected reward when taking an action $a_t$ in state $s_t$: i.e. that maximizes $Q^{\pi,\kappa}_t = Q^{\pi,\kappa}(s_t, a_t) = \mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t r^\kappa_{t+1}\right]$. We study agents that continue learning during transfer and aim to maximize *jump-start performance* (Zhu et al., 2020).

**Transfer with SF&GPI** requires two things: (1) state-features known as "cumulants" $\phi_{t+1} = \phi_\theta(s_t, a_t, s_{t+1})$, which are useful "descriptions" of a state-transition, and (2) a task encoding $w = w_\theta(\kappa)$, which define "preferences" over said transitions. Reward is then defined as $r^\kappa_t = \phi_t^\top w$ (Barreto et al., 2017). Successor Features $\psi^\pi$ are then *value functions* that describe the discounted sum of future $\phi$ that will be experienced under policy $\pi$:

$$\psi^\pi_t = \psi^\pi(s_t, a_t) = \mathbb{E}_\pi\left[\sum_{i=0}^\infty \gamma^i \phi_{t+i+1}\right] \tag{5.1}$$

Given $\psi^\pi_t$, we can obtain action-values for $r^\kappa_t$ as $Q^{\pi,\kappa}_t = {\psi^\pi_t}^\top w$.

The linear decomposition of $Q^{\pi,\kappa}$ is interesting because it can be exploited to *re-evaluate* $\psi^\pi$ for new tasks with GPI. Assume we have learned SFs $\{\psi^{\pi_i}(s,a)\}_{i=1}^{n_\kappa}$ for tasks $\mathbb{T}_{\texttt{train}}$. Given a new task $\kappa'$, we can obtain a new policy $\pi(s_t; \kappa')$ with GPI in two steps: (1) re-evaluate each SF with the task's *query* encoding to obtain new Q-values (2) select an action using the highest Q-value. In summary,

$$\pi(s_t, \kappa') \in \arg\max_{a\in\mathcal{A}} \max_{i\in\{1,\dots,n_\kappa\}} \{{\psi^{\pi_i}_t}^\top w_\theta(\kappa')\} = \arg\max_{a\in\mathcal{A}} \max_{i\in\{1,\dots,n_\kappa\}} \{Q^{\pi_i,\kappa'}_t\} \tag{5.2}$$

**Option Keyboard**. One benefit of equation 5.2 is that it enables transfer to *linear combinations* of training task encodings. However, it has two limitations. First, the feature "preferences" $w_\theta(\kappa')$ are fixed across time. When $\kappa'$ is a complex task (e.g. avoiding an object at some time-points but going towards it at others), we may want something that is state-dependent. Second, if we learn $w_\theta$ with a nonlinear function approximator such as a neural network, there is no guarentee that $w_\theta(\kappa')$ is in the span of training task encodings. The "Option Keyboard" (Barreto et al., 2019, 2020) can circumvent these issues by learning a transfer *policy* that maps states and tasks to preferences $w = g_\theta(s, \kappa)$:

$$\pi(s_t, \kappa') \in \arg\max_{a\in\mathcal{A}} \max_{i\in\{1,\dots,n_\kappa\}} \{{\psi^{\pi_i}_t}^\top g_\theta(s_t, \kappa')\} \tag{5.3}$$

**Learning**. In the most general setting, we learn $\psi_\theta, \phi_\theta, w_\theta$ and $g_\theta$ from experience. Since rewards $r^\kappa$ and therefore their task encodings $w$ reference deterministic task policies $\pi_\kappa$, one can parameterize

the approximator for $\psi^{\pi_\kappa}$ with its task-encoding, i.e. $\tilde{\psi}^{\pi_\kappa} = \tilde{\psi}^{\pi_w} = \tilde{\psi}^{w} \approx \psi_\theta(s, a, w)$. This is known as a *Universal* Successor Feature Approximator (USFA) and can be learned with td-learning (Borsa et al., 2019) with cumulants as pseudo-rewards. To discover $\phi_\theta$ and $w_\theta$, we match their dot-product to experienced reward (Barreto et al., 2017). We summarize this below:

$$\mathcal{L}_\psi = ||\phi_\theta(s_t, a_t) + \gamma\psi_\theta(s_{t+1}, a_{t+1}, w) - \psi_\theta(s_t, a_t, w)|| \qquad \mathcal{L}_r = ||r^\tau - \phi_\theta(s_t, a_t)^\top w_\theta(\tau)|| \quad (5.4)$$

No prior work has *jointly* learned a task encoder $w_\theta$ and USFA $\psi_\theta(s, a, w)$ while discovering cumulants $\phi_\theta$. In this work, we introduce the Deep Option Keyboard to address these limitations to enable transfer with SF&GPI and discovered representations in a large-scale 3D environment.

## 5.4 The Deep Option Keyboard and Categorical Successor Feature Approximator

We propose a novel Deep RL method for transfer, the *Deep Option Keyboard (Deep OK)*, where all necessary representations are discovered. To discover representations, we propose a new method, the *Categorical Successor Feature Approximator* for jointly learning SFs $\psi_\theta$, cumulants $\phi_\theta$, and task encodings $w_\theta$. The rest of this section is structured as follows. In §5.4.1, we describe CSFA and how to leverage it for pretraining. Finally, in §5.4.2, we describe how to leverage Deep OK for transfer.

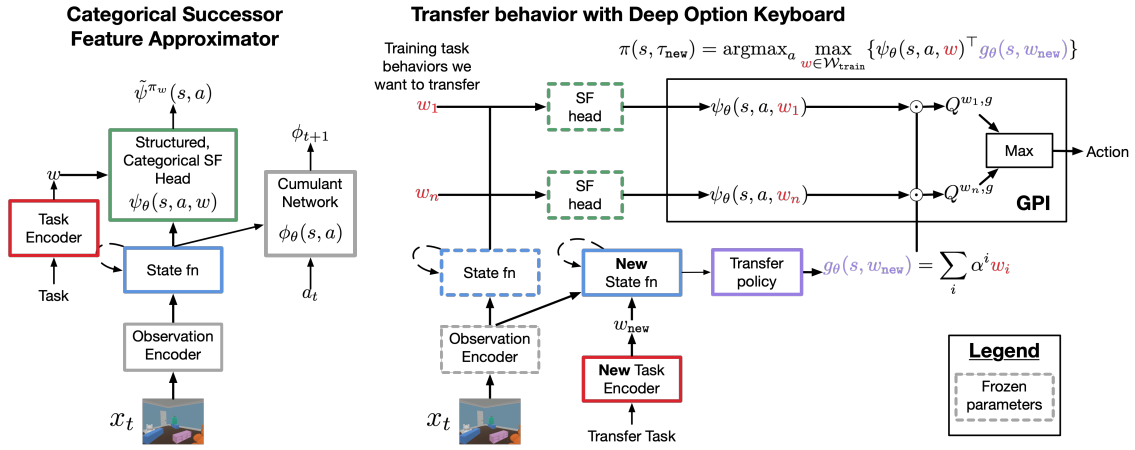


Figure 5.2: **Left: Categorical Successor Feature Approximator (CSFA)** estimates SFs for a task encoding $w$ with a structured, categorical network. **Right: Deep Option Keyboard (Deep OK)** transfers by dynamically selecting combinations of known task behaviors $\mathcal{W}_{\texttt{train}} = \{w_1, \ldots, w_n\}$. Deep OK accomplishes this by learning a policy $g_\theta(s, w_{\texttt{new}})$ that generates linear combinations of known task encodings $\mathcal{W}_{\texttt{train}}$. Deep OK then leverages GPI to compute $Q$-values for known behaviors, $Q^{w_i,g} = \psi(s, a, w_i)^\top g_\theta(s, w_{\texttt{new}})$ and acts using the highest $Q$-value.

### 5.4.1 Pretraining with a Categorical Successor Feature Approximator

We propose a novel learning algorithm, *Categorical Successor Feature Approximator (CSFA)*, composed of a novel architecture, shown in Figure 5.2, and a novel learning objective. **Challenge:** when jointly learning a Universal SF-approximator $\psi_\theta$ and a cumulant-network $\phi_\theta$ for long-horizon tasks, $\psi_\theta$ needs to fit $\phi_\theta$-generated returns that are potentially high-variance, non-stationary, and changing in magnitude (Barreto et al., 2018). CSFA addresses this challenge by modelling SFs with a probability mass function (pmf) over a discretized range of continous values. CSFA then fits this data by leveraging a categorical cross-entropy loss, enabling our estimator to give probability mass to different ranges of values. This is in contrast to prior work that models SFs with a point-estimate that is fit via regression (Barreto et al., 2017; Borsa et al., 2019). Leveraging a point-estimate can be unstable for modelling a non-stationary target with changing magnitude (Raffin et al., 2021) .

**Architecture**. At each time-step, we update a state function $s_\theta$ with an encoding of the current observation $z_t = f^{\texttt{enc}}(x_t)$, the previous action $a_{t-1}$, and the previous state representation $s_{t-1}$, i.e. $s_t = s_\theta(z_t, a_{t-1}, s_{t-1})$. CSFA estimates an SF $\psi^k_\theta(s, a, w) \in \mathbb{R}$ for each discovered cumulant $\phi^k_\theta(s_t, a_t)$ as the expectation of a learned probability mass function (pmf) $p_{\psi_k}(\cdot)$ over a predefined discrete domain $B = \{b_1, \ldots, b_m\}$, i.e. $\psi^k_\theta(s, a, w) = \mathbb{E}_{p_{\psi_k}}[B]$, where $b \in \mathbb{R}$ and $p_{\psi_k} \in \mathbb{R}^m$. We parameterize $p_{\psi_k}$ as a softmax with $m$ logits $l^\psi_k \in \mathbb{R}^m$ produced by a network that takes in the current state $s_t$, task encoding $w$, and an embedding for the current SF dimension $e_k$, i.e. $p_{\psi_k} \propto \exp(l^\psi_\theta(s_t, w, e_k))$. In summary,

$$\psi_\theta(s, a, w) = \{\psi^k_\theta(s, a, w)\}^n_{k=1} \qquad \psi^k_\theta(s, a, w) = \mathbb{E}_{p_{\psi_k}}[B] \qquad p_{\psi_k} \propto \exp(l^\psi_\theta(s_t, w, e_k)) \tag{5.5}$$

Using a pmf allows us to re-use the same network to estimate SFs across cumulants. We hypothesize that this provides a stronger learning signal to stabilize learning across more challenging return estimates. We show evidence in Figure 5.4.

**Learning objective**. We learn to generate behavior by employing a variant of Q-learning. In particular, we generate Q-values using learned SFs, and use these Q-values to create targets for both estimating Q-values and for estimating SFs. For both, targets correspond to the action which maximized future Q-estimates. We learn SFs with a categorical cross-entropy loss where we obtain targets from scalars with the $\text{twohot}(\cdot)$ operator. Intuitively, this represents a scalar with likelihoods across the two closest bins. For example, if bins increase by 1, 3.7 would be represented with likelihoods .3 and .7 across the two closest adjacent bins (see (Schrittwieser et al., 2020) for more).

In summary,

$$y_t^Q = r_{t+1} + \gamma \psi_{\theta^\circ}(s_{t+1}, a^*, w_{\theta^\circ}(\kappa))^\top w_{\theta^\circ}(\kappa) \qquad \mathcal{L}_Q = ||\psi_\theta(s_t, a_t, \underline{w_\theta(\kappa)})^\top \underline{w_\theta(\kappa)} - \underline{y_t^Q}||^2 \quad (5.6)$$

$$y_t^{\psi_k} = \phi_{\theta^\circ}^k(s_t, a_t) + \gamma \psi_{\theta^\circ}^k(s_{t+1}, a^*, w_{\theta^\circ}(\kappa)) \qquad \mathcal{L}_\psi^{\texttt{cat}} = \frac{1}{n} \sum_k \log p_{\psi_k}^\top \text{twohot}(\underline{y_t^{\psi_k}}) \quad (5.7)$$

$a^* = \arg\max_a \psi(s_{t+1}, a, w)^\top w$, where $p_{\psi_k} \propto \exp(l_\theta^\psi(s_t, \underline{w}, e_k))$, $\underline{w}$ is a stop-gradient operation on $w$. Like prior work (Mnih et al., 2015), we mitigate non-stationary in the return-targets by leveraging target networks with old parameters $\theta^\circ$ for computing $y_t^Q$ and $y_t^{\psi_k}$. The overall loss is $L = \beta_Q \mathcal{L}_Q + \beta_\psi \mathcal{L}_\psi^{\texttt{cat}} + \beta_\phi \mathcal{L}_r$.

**Important implementation details**. Estimating SFs while jointly learning a cumulant function and task encoder can be unstable in practice (Barreto et al., 2017). No work has jointly learned all three functions while sharing them across tasks. Here, we detail important implementation challenges that prohibited us from discovering representations that worked with SF&GPI in our large-scale setting. **D1**. We found that passing gradients to $w_\theta$ through $\psi_\theta(s, a, w)$ or through $\psi_\theta(s, a, w)^\top w$ during Q-learning lead to dimensional collapse (Jing et al., 2021) and induces a small angle between task encodings. We hypothesize that this makes $\psi_\theta(s, a, w)$ unstable and manifests as poor GPI performance. **D2**. When $||w||$ is large, it can make $\psi_\theta(s, a, w)$ unstable. To mitigate this, we bound $w$ by enforcing it lies on a unit sphere, i.e. $w = \frac{w_\theta(\kappa)}{||w_\theta(\kappa)||}$.

### 5.4.2 Transfer with the Deep Option Keyboard

The original Option Keyboard (equation 5.3) learned a policy that mapped states $s_t$ to queries $w$, $w' = g(s_t, \kappa^{\texttt{n}})$. However, they used a hand-designed $w_\theta$ and thus hand-designed space for $w$. In our setting, we learn $w_\theta$. However, GPI performance is bound by the distance of a transfer query $w'$ to known preference vectors $w$ (Barreto et al., 2017). Thus, we want to sample $w'$ that are not too "far" from known $w$. To accomplish this, we shift from learning policy that samples preference vectors to a policy that samples *coefficients* $\{\alpha_i\}_{i=1}^{n_\kappa}$ for known preference vectors $\mathcal{W}_{\texttt{train}} = \{w_1, \ldots, w_{n_\kappa}\}$. The GPI query is then computed as a weighted sum. Below we describe this policy in more detail along with how to learn it.

At each time-step, the agent uses a pretrained CSFA to compute SFs $\psi^{\pi_w}(s_t, a)$ for $w_i \in \mathcal{W}_{\texttt{train}}$:

$$z_t = f_\theta(x_t) \qquad s_t = s_\theta(z_t, a_{t-1}, s_{t-1}) \qquad \{\psi^{\pi_{w_i}} = \psi_\theta(s_t, a, w_i)\}_{w_i \in \mathcal{W}_{\texttt{train}}} \quad (5.8)$$

In our experiments, we freeze the observation encoder, state function, and task encoder and learn a new state function and task encoder at transfer time with parameters $\theta_{\texttt{n}}$. Given a new state

representation $s_t^{\mathtt{n}}$, we sample coefficients independently:

$$w'_t = g_{\theta_{\mathtt{n}}}(s_t^{\mathtt{n}}, w_{\theta_{\mathtt{n}}}(\kappa^{\mathtt{n}})) = \sum_{i=1}^{n_\kappa} \alpha_t^i w_i \qquad \text{where} \qquad \alpha_t^i \sim p_{\theta_{\mathtt{n}}}(\alpha^i | s_t^{\mathtt{n}}, w_{\theta_{\mathtt{n}}}(\kappa^{\mathtt{n}})) \tag{5.9}$$

We find that a Bernoulli distribution performs well. We learn this $\alpha$-coefficient policy with policy gradients (Sutton et al., 1999a), $\nabla \mathcal{L}_{\theta_{\mathtt{n}}} = \sum_i (R_t - V_{\theta_{\mathtt{n}}}(s_t^{\mathtt{n}})) \nabla \log p_{\theta_{\mathtt{n}}}(\alpha_i | s_t^{\mathtt{n}}, w^{\mathtt{n}})$ where $R_t = \sum_{i=0}^{\infty} r_{t+i+1}$ is the return experienced from that time-point onwards, and $V_{\theta_{\mathtt{n}}}(s_t^{\mathtt{n}})$ is an estimate of the return which, when subtracted, reduces variance in the gradients.

## 5.5 Experiments

We study transfer with sparse-reward long-horizon tasks in the complex 3D Playroom environment (Abramson et al., 2020). To transfer behavioral knowledge, we propose Deep OK for combining behaviors with SF&GPI, and CSFA for discovering the necessary representations. In §5.5.1, we study the utility of CSFA for discovering representations that are compatible with SF&GPI. In §5.5.2, we study the utility of Deep OK for transferring to sparse-reward long-horizon tasks.

**Environment setup**. We conduct our experiments in the 3D playroom environment. **Observations** are partial and egocentric pixel-based images. The agent gets no other information. **Actions**. The agent can rotate its body and look up or down. To pick up an object it must move its point of selection on the screen. When it picks up an object, it must continuously *hold it* in order to move it elsewhere. To accomplish this, the agent has 46 actions. **Training tasks**. The agent experiences $n_\kappa = 32$ training tasks $\mathbb{T}_{\mathtt{train}} = \{\kappa_1, \ldots, \kappa_{n_\kappa}\}$ composed of "Find A" and "Place A near B". $|A| = 8$ and $|B| = 3$. All tasks provide a reward of 1 upon-completion.

Figure 5.3: **Playroom**.

### 5.5.1 Evaluating the utility of discovered representations for SF&GPI

Our first experiments are a *sanity check* for the utility of discovered representations for SF&GPI. We train and evaluate agents on the same set of tasks. However, during evaluation the agent has access to all training tasks and must select the appropriate one with SF&GPI; given task encodings $\mathcal{W}_{\mathtt{train}} = \{w_1, \ldots, w_{n_\kappa}\}$, the agent acts according to policy $\pi(s, w_i) = \arg\max_a \max_{w_k} \{\psi(s, a, w_k)^\top w_i\}$. When $w_k = w_i$, $\pi(s, w_i) = \arg\max_a Q(s, a, w_i)$. This will fail if the agent hasn't learned representations that support GPI. If an agent cannot perform GPI on training tasks, then it will probably fail

with novel transfer tasks. **Metric**. We evaluate agents with average success rate. **Challenges**. Most prior work has leveraged SF&GPI for combining "Find tasks" where an agent simply navigates to objects (Barreto et al., 2018; Borsa et al., 2019; Carvalho et al., 2023a). We add a significantly *longer horizon* "Place Near" task where the agent must *select* an object and *hold* it as it moves it to another object. This tests the utility of discovered representations for learning SFs that enable SF&GPI over long horizons.

**Research questions. Q1**. How does CSFA compare against baseline methods that share their SF estimatar $\psi$ across tasks while discovering $\phi$ and $w$? **Q2**. Is each piece of CSFA necessary?

**Baselines**. (1) **Universal Successor Feature Approximators (USFA)** (Borsa et al., 2019) is the only method that shares an SF estimator across tasks and has shown results in a 3D environment. (2) **Modular Successor Feature Approximators (MSFA)** (Carvalho et al., 2023a) showed that leveraging modules improved SF estimation and enabled cumulant discovery. However, they did not discover task encodings and only showed results in simple grid-worlds. **Both** baselines estimate SFs with point-estimates. Comparing to them tests (a) the utility of our categorical representation and (b) CSFA's ability to discover *both* cumulants and task encodings that enable GPI in a large-scale setting.

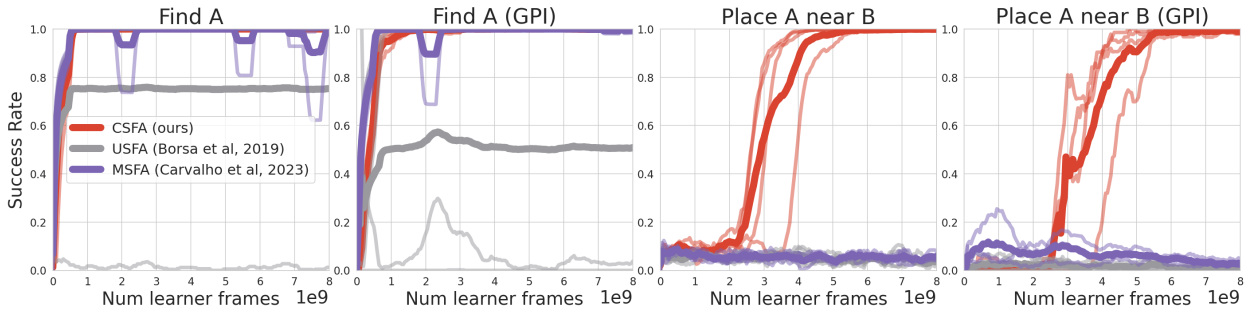


Figure 5.4: **CSFA discovers representations compatible with GPI across short and long task-horizons**. We see that while most USFA seeds learn Find tasks, a smaller subset can perform GPI. MSFA can do so consistently. Neither are able to learn our longer horizon task. We hypothesize that this is because they approximate SFs with a point-estimate. (4 seeds)

**CSFA discovers SFs compatible with SF&GPI while baseline methods cannot**. For fair comparison, we train each baseline with the same learning algorithm (except for SF-losses), enforce $w$ lie on a unit sphere, and stop gradients from Q-lerning. We found that large-capacity networks were needed for discovering $\phi$. In particular, we parameterized $\phi_\theta$ with a 8-layer residual network (ResNet) (He et al., 2016) for all methods. One important difference is that we leverage a set of ResNet modules for MSFA since Carvalho et al. (2023a) showed that modules facilitate cumulant discovery. Figure 5.4 shows that USFA and MSFA can perform GPI for Find tasks; however, neither learn place tasks in our computational budget. Given that we controlled for how $\phi$ and $w$ are learned,

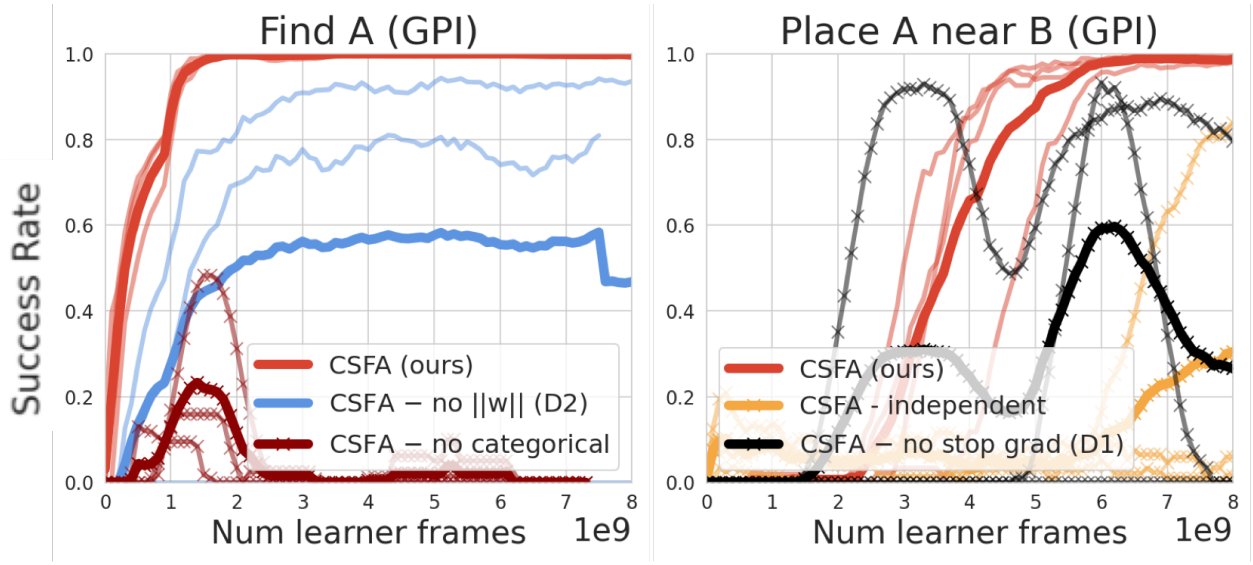


Figure 5.5: **CSFA Ablations. Left**. Not bounding $||w||$ (**D2**) degrades GPI performance. Interestingly, removing our categorical representation (CSFA - no categorical) gets perfect training performance but terrible GPI performance. **Right**. Passing gradients to the task-encoder from the SF-approximator leads to unstable GPI performance (**D1**) despite good training performance. If CSFA does not share its SF-estimator, it seems to learn more slowly. Thankfully, each addition is simple and together enables GPI with long-horizon place tasks. (3 seeds)

we hypothesize that the key limitation of USFA and MSFA is their reliance on scalar SF estimates.

**A categorical representation is necessary**. One difference between MSFA/USFA and CSFA is that CSFA shares an estimator parameters across individual SFs. Figure 5.5 shows that when CSFA shares an estimator but produces scalar estimates (CSFA - no categorical), GPI performance degrades *below* MSFA/USFA. We hypothesize that a network producing point-estimates has trouble estimating returns for cumulants of varying magnitude. **Sharing our estimator across cumulants is necessary**. If we keep our categorical representation but don't share it across cumulants (CSFA - independent), GPI performance degrades on our long-horizon place near task. **Stopping gradients from Q-learing is necessary**. Interestingly, when we pass gradients to the task-encoder from Q-learning (CSFA - no stop grad), we can get perfect train performance on place tasks but highly unstable GPI performance. We found that passing gradients leads to dimensional collapse (Jing et al., 2021). We hypothesize that this makes $\psi_\theta$ unstable. Likewise, if we don't bound tasks (CSFA - no $||w||$), we find degraded GPI performance compared to training but for even simpler tasks. While other methods may work, enforcing $w$ lie on a unit-sphere is a simple solution.

### 5.5.2 Transferring to combinations of long horizon tasks

Our second experiments test the utility of Deep OK for transferring to combinations of long horizon, sparse-reward tasks. **Transfer tasks** are conjunctions of known known tasks. Subtasks can be completed in any order but reward is only provided at task-completion. Find tasks contribute a reward of 1 and place tasks contribute a reward of 2.

**Research questions**. **Q3**. How do we compare against baseline transfer methods? **Q4**. How important is it to use CSFA and to sample coefficents over known task encodings?

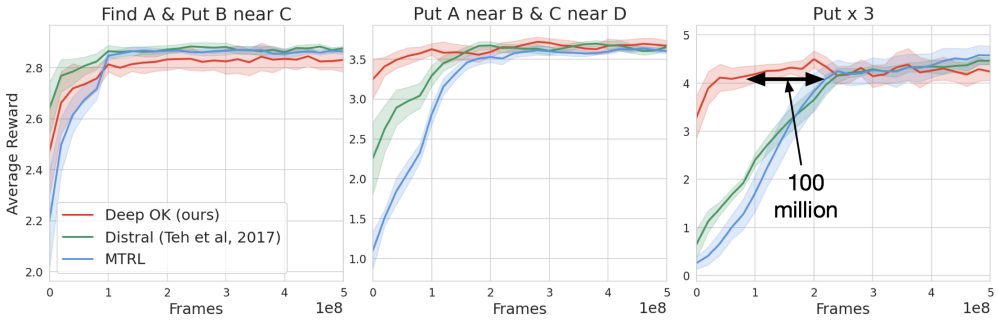


Figure 5.6: **Deep OK transfer most quickly to longer horizon tasks**. Distral and MTRL learn an about the same speed, though Distral is slightly faster. For put x 3, where the agent needs to do 3 place tasks (A near B, C near D, and E near F) Deep OK learns with 100+ fewer samples. (9 seeds)

**Baselines**. (1) **Multitask RL (MTRL)**. We train an Impala (Espeholt et al., 2018) agent on all training tasks and see how this enables faster learning for our transfer tasks. We select Impala because it is well studied in the Playroom environment (Chan et al., 2022; Hill et al., 2019; Abramson et al., 2020). (2) **Distral** (Teh et al., 2017) is a common transfer learning method which learns a centroid policy that is distilled to training task policies. Comparing against MTRL and Distral tests the utility of combining behaviors with SF&GPI.

**Q3: Deep OK has better jumpstart performance**. Figure 5.6 shows that all methods get similar performance by 500 million frames. However, for longer task combinations (Put 2 times or 3 times), Deep OK gets to similar performance with far fewer frames. When we remove our curriculum (Figure 5.7) this gap further increases. No method does well for our longest task (Put x 4) which involves 8 objects. We conjecture that one challenge that methods face is holding an object over prolonged periods of time. If the agent selects the wrong action, it will drop the object it's holding. This may make it challenging when there's some noise from either (a) centroid task, as with Distral, or (b) SF&GPI as with Deep OK. Despite not reaching optimal performance, Deep OK provides a good starting point for long-horizon tasks.

**Q4: Leveraging CSFA and sampling coefficents over known task encodings is critical to jumpstart performance**. Figure 5.8 shows us that if we don't leverage CSFA to estimate SFs and

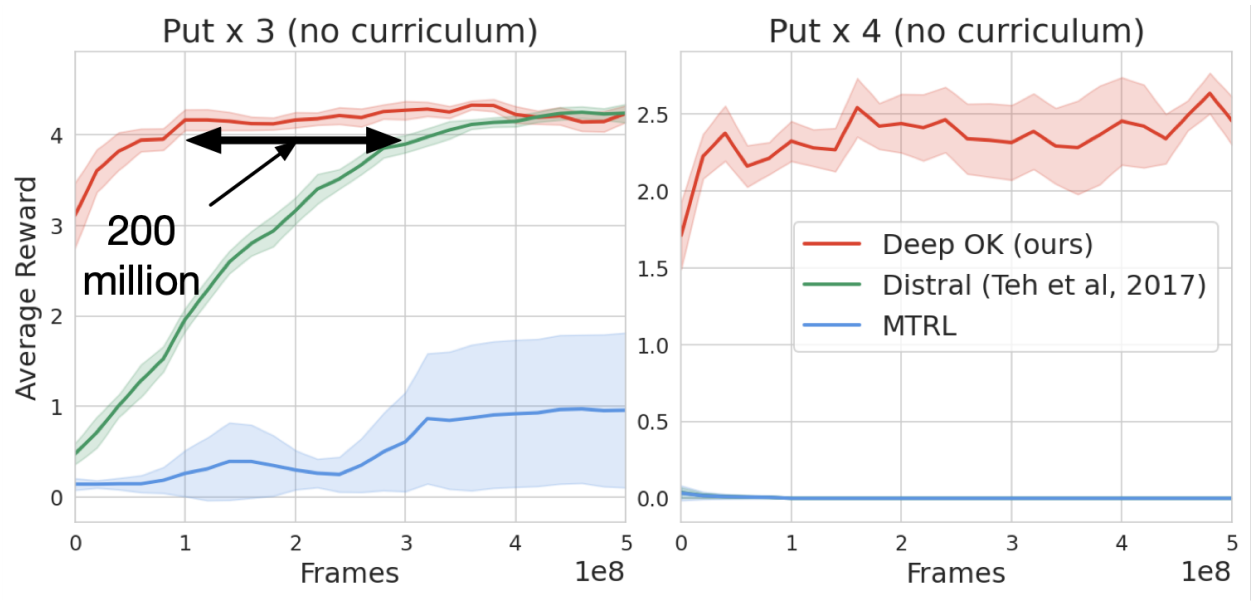


Figure 5.7: **Deep OK transfers most quickly with no curriculum of shorter tasks**. Deep OK and Distral reach the same performance but Deep OK is 200+ million frames faster. MTRL fails to transfer here. For "Put x 4", Deep OK is suboptimal but achieves some success. (8 seeds)

instead use USFA, Deep OK fails to transfer. We hypothesize that this is because of the USFA uses point-estimates for SFs, which shows poor GPI on training tasks (see Figure 5.4) so its not surprising it fails on novel transfer tasks. We find that directly sampling from the encoding space does not transfer nearly as quickly. Empirically, we find that learned task encodings tend to have a high cosine similarity, indicating that they occupy a relatively small portion of the possible encoding space. We hypothesize that this makes it challenging to directly sample in this embedding space to produce meaningful behavior.

## 5.6 Discussion and conclusion

We have presented Deep OK, a novel method for transfer that adaptively combines known behaviors using SF&GPI. To discover representations that are compatible with SF&GPI, Deep OK estimates SFs with a novel algorithm, CSFA. CSFA constitutes both a novel architecture (of the same name) which approximates SFs with a pmf over a discretized continous values and a novel learning objective which estimates SFs for discovered cumulants with a categorical cross-entropy loss.

We first showed that CSFA is able to discover SF&GPI-compatible cumulants and task encodings for long-horizon sparse-reward tasks in the 3D Playroom environment (§5.5.1). We compared CSFA to other methods which share their approximator across tasks: (1) USFA, which showed results in a 3D environment but hand-designed representations, and (2) MSFA, which discovered cumulants

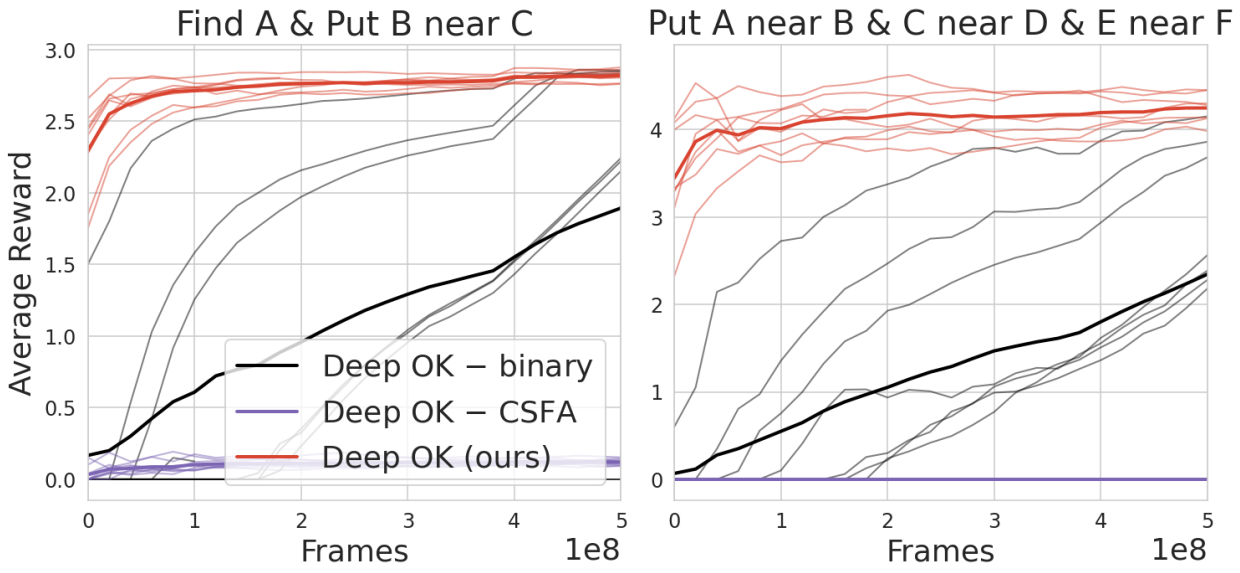


Figure 5.8: **Deep OK ablation**. Deep OK fails to transfer if we remove CSFA. We find that sampling directly in task-encoding space instead of sampling binary coefficents is both slower to learn and has higher performance variance. We hypothesize that this is due to concentation in the task encoding space. (6 seeds)

but did so in gridworlds with hand-designed task encodings. We additionally compared against an ablation which removed our categorical representation and or did not share it across cumulants. Our results show that a categorical representation over discretized values can better handle estimating SF-returns when discovering cumulants and task encodings for long-horizon sparse-reward tasks.

We built on these results for our second set of experiments and showed that Deep OK provides strong jumpstart performance for transfer to combinations of training tasks (§5.5.2). We compared Deep OK to (1) Mulitask RL (MTRL) pretraining and finetuning, and (2) Distral, which distills knowledge back and forth between a task-specific policy and a "centroid" policy. Our results showed that, for long-horizon tasks, Deep OK could transfer with 100 million+ fewer samples when a curriculum was present, and 300 million+ fewer samples when no curriculum was present.

**Limitations**. While we demonstrated discovery of cumulants and task encodings that enabled transfer with a dynamic SF&GPI query, we relied on generating queries as weighted sums of known task encodings. A more general method would directly sample queries from the task encoding space. We found that the task encodings we discovered were fairly concentrated. Future work may mitigate this with contrastive learning (Schroff et al., 2015). This respects the dot-product relationship of cumulants and task encodings while enforcing they be spread in latent space. Finally, while we demonstrated good jumpstart performance for long-horizon tasks, we did not reach optimal performance in our sample budget. Despite building on the Option Keyboard, we did not employ options as we sampled a new query at each time-step. Future work may improve performance by

sampling queries more sparsely and by implementing a latent "initiation set" (Veeriah et al., 2022).

These results present the first demontration of transfer with the Option Keyboard (SF&GPI with a dynamic query) when all representations are discovered and when the task-encoder and SF-approximator are *shared* across tasks. This may enable other methods that leverage SFs in a multi-task settings (e.g. for exploration (Janz et al., 2019) or for multi-agent RL (Filos et al., 2021; Gupta et al., 2021)) to leverage discovered representations. More broadly, this work may also empower neuroscience theories that leverage SFs as cognitive theories (e.g. for multitask transfer (Tomov et al., 2021) or for learning cognitive maps (De Cothi et al., 2022)) to better incorporate discovered representations. We are hopeful that Deep OK and CSFA will enable SFs to be adopted more broadly with less hand-engineering.

# CHAPTER 6

# Conclusion

The thesis of this disseration is that leveraging sparsely connected and structured neural networks in the core components of an RL learner can enable the discovery of representations which improve sample-efficiency and generalization. In Chapter 1, I motivated this by recent advancements in AI, which have shown that such structured neural networks can improve sample efficiency and generalization in computer vision and natural language processing. However, this research was also motivated by literature in cognitive neuroscience which suggests that structured, predictive representations are key to human learning speed and generalization. In this final chapter, I will connect the work in each chapter to literature in cognitive neuroscience, describe what artificial intelligence can still glean from these findings, and postulate how this work may contribute to future research in cognitive neuroscience.

## 6.1 Summary of contributions in context of cognitive neuroscience

**In Chapter 2**, I drew from evidence suggesting that humans learn predictions about objects in the environment. In particular, cognitive science has found evidence that infants expect objects to have continuity (i.e. they can't pass through each other), cohesiveness (they can't spontaneously break apart), and solidity (they can't occupy the same space) (Spelke, 1990). Within neuroscience, prominent theoretical accounts of human vision assert that representations in the inferior temporal cortex and in the occipital and temporal lobes are explicitly about object categories (Kriegeskorte et al., 2008; DiCarlo et al., 2012). Given this evidence, I hypothesized that learning object-oriented predictions would enable the discovery of object-centric representation that reduced the nuber of samples an agent needed for sparse-reward object-interactiont tasks in a photorealistic 3D environment.

To test this hypothesis, I developed an agent which learned an object-centric transition model where predicting how an object evolved was formalized as a classification problem. In this setting,

the agent predicted an object's next state and them compared its prediction to "incorrect" object-states randomly sampled from the agent's experience. This simple solution enabled unsupervised discovery of representations which captured an object's category (is it a toaster?), its properties (is it on?), and object-relations (is something inside of it?). From a sample-efficiency perspective, our method learned as quickly as an agent equipped with hand-designed object-representations. However, one limitation of this work was that I relied on an agent having access to an object-detector to segment its perception into useful pieces.

**In Chapter 3**, I sought to mitigate this limitation by exploiting a mixture of modularity and visual attention. Neuroscience researchers have argued that brains have sets of "expert" modules that specialize on different aspects of sensory data. The argument is that leveraging modules enables generalization to novel combinations of sensory inputs and to system perturbations (Graybiel et al., 1994; Ghahramani and Wolpert, 1997; Gershman et al., 2009). For example, Ghahramani and Wolpert (1997) presented a neuroscience model for visuomotor learning where different modules were responsible for representing different parts of a state space. They showed that this parameterization enabled a visual system to generalize to new starting states and to environment perturbations where the agent's actual location different from its perceived location.

Brains seem to exploit modules to represent subsets of perception. However, most models assume fixed, hand-designed functions for how module select their inputs (Graybiel et al., 1994; Ghahramani and Wolpert, 1997; Gershman et al., 2009). In neuroscience, researchers have suggested attention models to explain how processes in the visual system select inputs from each other (Sperling and Weichselgartner, 1995; Sperling, 2018).

My hypothesis was that the combination of modularity and attention would enable discovering useful object representations with fewer assumptions and improve generalization across object-centric environments. To test this hypothesis, I developed state modules that discovered generically useful state factors by using attention to select their own inputs. We found that this enabled the discovery of useful object primitives that were distributed across subsets of modules. For example, we found evidence that combinations of modules learned to represent primitive state events such as moving around an obstacle and picking up goal objects. Additionally, this also enabled generalization to longer tasks with more distractor objects across a diverse set of object-centric environments. Despite these promising results, one limitation of this work was that it only enabled generalization to variations of a task—a more general method would allow for generalization to *combinations* of known tasks.

**In Chapter 4**, I sought to enable generalization to combinations of tasks by drawing on a popular neuroscience representation scheme known as "successor representations" (Dayan, 1993) and its deep learning generalization, "successor features" (Barreto et al., 2016). As a reminder, successor features are predictive representations that represent states with predictions of what features will

follow. Interestingly, there is significant evidence that brains learn successor features. For example, neuroscience researchers have found that successor features to explain how place fields represent cognitive maps (Stachenfeld et al., 2017), why transferring to new rewards is easier than transferring to new transition functions (Momennejad et al., 2017), and how humans re-apply behaviors to new tasks (Tomov et al., 2021). One limitation of both neuroscience and AI research leveraging successor features is that they commonly rely on hand-designed representations. To both develop scalable AI algorithms and to model human learning, we need algorithms that can work with sensory data such as visual observations.

In the previous chapter, I showed that learning modules enabled the discovery of useful state primitives. In this chapter, I hypothesized that modules could be used to discover the features that successor features would form predictions over. To accomplish this, I developed an agent that learned successor feature modules where individual modules both discovered what is useful to predict and learned successor features over over these discovered representations. By learning successor feature modules, we not only get generalization to task variations, we also get generalization to *combinations* of tasks. For example, once an agent has learned to pick up an apple and a potato, it can now pick up combinations of apples and potatoes for free. By leveraging modules, we were able to discover the features that were used to represent these objects. Additionally, this solution enabled zero-shot transfer that was competitve to methods that used hand-designed features. Despite these benefits, we faced two major obstaces. The first is that successor features rely on hand designed task representations—and we were not able to circumvent this requirement. The second is that one transfers to a task task with a fixed transfer task representation. This prevented us from transferring to tasks that involved both conjunctions and sequences of known tasks.

**In Chapter 5**, I sought to address these challenges by drawing on inspiration from literature on hierarchical reinforcement learning. Significant evidence suggests that humans and animals are able to chain behaviors sequentially in-time by leveraging a meta-controller which adaptively selects subgoals for an agent to follow (Botvinick, 2012; Botvinick and Weinstein, 2014; Merel et al., 2019). Brains seem to leverage hierarchical RL to chain behaviors. Brains also seem to use successor features to transfer behavioral knowledge. Is there a way to integrate the two of these?

One limitation of the algorithm we used in Chapter 4 was that we transferred to new tasks with a fixed representation for transfer tasks. In this chapter, we mitigated this challenge by having the transfer representation be time-dependent and addressed how to leverage *learned* task representations. The lesson of this work is much the lesson of contemporary deep learning and deep reinforcement learning: the pieces to solve the problem already exist in the literature; however, we need careful engineering to figure out how to leverage them in large-scale settings. Discovering all representations needed for learning successor features leads to a non-stationary prediction problem that is challenging to learn with standard methods. However, if we learn successor feature modules

that represent successor features with a categorical representation and leverage sparse, structured neural networks for discovering features, we can discover all representations for transfer with successor features and transfer to both conjunctions and sequences of tasks.

## 6.2 Towards more human-like human-level AI

Currently, human intelligence in its broad flexibility — its ability to learn diverse and dispirate tasks, its ability to learn and generalize in the real world — is unrivaled by artificial intelligence. What are some of the key ingredients to human-like human-level intelligence?

**Discovering objects**. Humans have an extraordinary ability to uncover structure that is present in the environment. I began this chapter by describing one source of structure that humans are well-equipped to discover: objects. In Chapter 2, I imbued an AI agent with the ability to detect objects by providing it an object detector. While I didn't assume knowledge of object information (e.g. categories or properties), I found that I was still limited by the detector as this placed a ceiling on what our agent could learn about.

While humans learn object-oriented representations, the objects which we hold in our minds can be quite diverse. We can reference a cup on a table, the moon in the sky, the forest in front of us, and the particles in our cup. Humans seem to capable of delienating arbitrary aspects of their perception into useful "objects" to use for reasoning. If the sky seems dark, I may reason that it will rain and choose to stay indoors. If our office building's lights are off, I may reason that it is closed and that I cannot enter it. If we are to imbue machines with object-oriented representations, we need algorithms that make minimal assumptions about the format of those objects. Recent research has shown that simply learning a *structured* transition model over a *structured* state space can enable the discovery of objects (at least in rudimentary settings) (Kipf et al., 2019). This suggests that the key to object discovery may lie in combining that right *prediction problem* with the right set of (minimal) assumptions.

Returning to cognitive neuroscience, these methods may provide theoretical models for how brains discovers object representations. Recently, Konkle and Alvarez (2022) showed that contrastive learning that simply contrasts pairs of images can explain representations in lower layers of the ventral stream but fails to capture more object-oriented features in the inferior temporal cortex. The method we introduced in Chapter 2 is a strict generalization of their method where we leverage contrastive learning with a structured world model. Thus, given the ability of these methods to enable object-discovery, perhaps our algorithm can inform future theories for how object-representations are learned brain regions such as the inferior temporal cortex.

**Discovering agents**. In addition to object-oriented representations, evidence suggests that humans leverage agent-oriented representations. Cognitive science research has shown that early in

infancy humans have expectations that agents will have (a) beliefs about the world, (b) goals they pursue subject to constraints, and (c) pursue goals in an efficient manner (Spelke and Kinzler, 2007; Jara-Ettinger et al., 2020). This brings to question: is there a basic prediction problem and set of minimal assumptions to enable agent discovery in artificial systems? Keeping to the theme of this thesis—consider enforcing a structured state function as we did in chapters 3 and 4—perhaps we can discover agent-oriented primitives by learning to predict a state factor's (a) policy (b) reward function and (c) value function? This might enable AI agents to discover representations of state factors as agents that efficiently pursue goals as "optimal" RL agents.

**Generally discovering primitives in a large and complex world**. In both Chapters 3 and 4, I found that simply leveraging modules enabled the discover of useful primitives which promoted generalization. This is consistent with other work in which structured, modular neural networks facilitate feature discovery. For example, Transformers have multiple attention "heads" where different heads will capture different syntactic structures in language. When I did secondary analysis, I found a lot of redundancy among the modules. People have found similar with Transformers. Many modules can be effectively "zeroed-out" and the agent would still generalize. Reflecting on this work and on the state of current in-simulation AI research, I conjecture that we don't currently have rich and sophisticated enough environments to benefit from modularity. For these experiments, I was able to recover our results with just 2 or 4 modules. Presumably, the benefit of having dozens of even hundreds of modules will come when we have such rich and varied environments and an agent **needs** to reserve sparse subsets of modules to represent radically different aspects of its experience. Rather than learning in a single house, I suspect we will need environments where an agent must learn within a neighborhood or city filled with diverse houses and rich terrains that the agent must traverse over.

**Planning with a causal world model**. One key finding of this thesis is that simply adding structure and enforcing sparsity within a state function can facilitate the discovery of useful primitives for representing the environment. Having explicit state factors can enable relational reasoning over these factors. One key human ability is our ability to identify causal variables and infer a causal structure for our experience (Goddu and Gopnik, 2020; Gerstenberg et al., 2021; Wente et al., 2019). Recently, cognitive scientists have tried to imbue this into RL agents and dubbed it "Theory-based RL" (Tsividis et al., 2021), in reference to the theory-theory (Gopnik and Wellman, 1994). Here, RL agents infer "theories" for their environments as causal models which they then use for planning. While this is an exciting direction for RL, it thus far has relied on hand-designed representations of possible objects and interactions. Given the findings of this thesis, I posit that adding structure and enforcing sparsity may enable "Theory-based **Deep** RL", where the primitives that are the basis for causal inference can be discovered rather than hand-engineered. If we can develop Theory-based Deep RL agents, perhaps we can begin to form theoretical models for how

humans are able to plan with causal models in rich, high-dimensional, pixel-based environments that are filled with a multitude of diverse and unfamiliar objects and interactions.

**Transferring object-oriented behavioral skills**. Chapters 4 and 5 focused on discovering representations that enabled transfer with successor features. To discover these features, I setup a prediction problem where the features had to predict the environment reward signal. While general, this meant that the resultant features were limited by the reward signal and could only represent the aspects of the environment that defined reward. A more general method would learn to represent and form predictions over arbitrary aspects of the environment. But how do we enable an agent to discover features (and thus predictions) over broad and general aspects of the environment?

Cognitive science tells us that humans learn predictions over representations of objects. This thesis and concurrent work (Kipf et al., 2019) has shown that learning a structured world model can enable the discovery of object-oriented representations. If features that defined rewards are covered by the features that describe objects, then the features discovered by a structured world model might be good candidates to learn successor features over. If we can learn successor features over discovered object representations, this may enable an agent to mix-and-match object-oriented behavioral predictions with object-oriented subgoals for flexible recombination of its behavioral skills. Consider an object-oriented representation of the environment which indicates that the refridgerator has vegetables while cabinets have dishware. Behaviors oriented around objects then become gateway predictions to the other objects present: opening the refridgerator leads to vegetables and opening the cabinet leads to dishware. Such "object-oriented" successor features may be key to flexible, human-like behavioral transfer for artificial agents. From a cognitive neuroscience perspective, we know that humans leverage both object-oriented representations and successor feature like representations. This research may thus inform theoretical models for how humans compose object-oriented behavioral skills.

**What role do large language models play in this picture?** Large language models (LLMs) are a class of unsupervised learningLarge language models (LLMs) are a class of unsupervised learning algorithms that, given some context of words (e.g., sentences), learn to predict the next word. One of the most surprising developments of the 21st century is that if you train these LLMs on petabytes of human data, they begin to produce both human-like and human-level behavior for text-based reasoning. This has led some to coin LLMs as effectively "amortized humans". LLMs are a powerful and interesting ingredient in producing both human-like and human-level intelligence due to their ability to mimic human-like behavior in text and reproduce it at what seems like a human-level.

LLMs can effectively capture any knowledge that humans know how to put down into text. This can be either in the form of explicit rules (e.g., the rules of chess) or exemplar-based definitions that define abstract patterns governing rules. However, humans possess a lot of tacit knowledge that we

do not know how to write down. Perhaps the most notable is the social domain. When we interact with each other, we leverage tacit knowledge to map people's tones, inflections, talking speed, and body language to their emotional and mental states. Speaking quickly, for instance, might indicate nervousness, excitement, or joy, depending on the situation and individual personality, which is often implicitly defined within our minds. Body language also plays a role in this.

LLMs, in their current incarnation, rely on the premise that significant amounts of knowledge can be discovered by learning to predict the next pieces of data based on a history of data. This approach has worked remarkably well in language domains where we have petabytes of data to learn from. In fields like mathematics, computer science, and law—due to copious amounts of high-quality, human-generated data—this enables the development of human-level AI assistants. However, in other domains, such as social reasoning, we lack copious amounts of high-quality data.

Additionally, in domains like video, it is unclear whether the strategy of simply predicting the next data point is a feasible learning problem for capturing the structure inherent in the data. This is because videos consist of sequences of high-dimensional images, whereas language consists of highly compressed, well-defined tokens. Whether the training regime for LLMs is sufficient for these challenges remains uncertain. While I do not claim to have the answer, I merely highlight this challenge and emphasize that achieving human-like, human-level intelligence will require the ability to acquire tacit knowledge in areas such as social reasoning and with complex data modalities such as video.

## 6.3 Concluding Remarks

We have shown that leveraging sparsely connected and structured neural networks in the core components of an RL learner can enable the discovery of representations which improve sample-efficiency and generalization. This hypothesis was motivated by cognitive neuroscience research that has argued that structured functions over objects and agents is key to human-like human-level intelligence. While human intelligence may not be the pinnacle of all possible intelligences, it is a fascinating exemplar. I hope this thesis provides inspiration to future work in AI that aims to draw inspiration from cognitive neuroscience and to future work in cognitive neuroscience that aims to draw inspiration from AI. While this thesis may only be a few bricks of the bridge between the two fields, I hope that it contributes to a new era of understanding for how intelligence may flourish in an environment as complex as the real world.

# BIBLIOGRAPHY


Abdolshah, M., Le, H., George, T. K., Gupta, S., Rana, S., and Venkatesh, S. (2021). A new representation of successor features for transfer across dissimilar environments. In *International Conference on Machine Learning*, pages 1–9. PMLR.

Abramson, J., Ahuja, A., Barr, I., Brussee, A., Carnevale, F., Cassin, M., Chhaparia, R., Clark, S., Damoc, B., Dudzik, A., et al. (2020). Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672*.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48.

Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Mourad, S., Silver, D., Precup, D., et al. (2019). The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. (2018). Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *ICML*, pages 501–510. PMLR.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Van Hasselt, H., and Silver, D. (2016). Successor features for transfer in reinforcement learning. *NIPS*.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. *NIPS*, 30.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. (2020). Fast reinforcement learning with generalized policy updates. *PNAS*, 117(48):30079–30087.

Bellman, R. E. (2010). *Dynamic programming*. Princeton university press.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D., Munos, R., Van Hasselt, H., Silver, D., and Schaul, T. (2019). Universal successor features approximators. *ICLR*.

Botvinick, M. and Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655):20130480.

Botvinick, M. M. (2012). Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, 22(6):956–62.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brooks, E., Rajendran, J., Lewis, R. L., and Singh, S. (2021). Reinforcement learning of implicit and explicit control flow instructions. In *International Conference on Machine Learning*, pages 1082–1091. PMLR.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*.

Carvalho, W., Filos, A., Lewis, R. L., Singh, S., et al. (2023a). Composing task knowledge with modular successor feature approximators. *arXiv preprint arXiv:2301.12305*.

Carvalho, W., Lampinen, A., Nikiforou, K., Hill, F., and Shanahan, M. (2021a). Feature-attending recurrent modules for generalization in reinforcement learning. *ICML Unsupervised RL Workshop*.

Carvalho, W., Liang, A., Lee, K., Sohn, S., Lee, H., Lewis, R. L., and Singh, S. (2021b). Reinforcement learning for sparse-reward object-interaction tasks in first-person simulated 3d environments. *IJCAI*.

Carvalho, W., Saraiva, A., Filos, A., Lewis, R. L., Lee, H., Singh, S., Zoran, D., and Rezende, D. (2023b). Discovering representations for transfer with successor features and the deep option keyboard. *In Preparation*.

Chan, S. C., Lampinen, A. K., Richemond, P. H., and Hill, F. (2022). Zipfian environments for reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 406–429. PMLR.

Chaplot, D. S., Sathyendra, K. M., Pasumarthi, R. K., Rajagopal, D., and Salakhutdinov, R. (2018). Gated-attention architectures for task-oriented language grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2019). BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020). Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR.

Corona, R., Fried, D., Devin, C., Klein, D., and Darrell, T. (2020). Modular networks for compositional instruction following. *arXiv preprint arXiv:2010.12764*.

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

De Cothi, W., Nyberg, N., Griesbauer, E.-M., Ghanamé, C., Zisch, F., Lefort, J. M., Fletcher, L., Newton, C., Renaudineau, S., Bendor, D., et al. (2022). Predictive maps in rats and humans for spatial navigation. *Current Biology*, 32(17):3676–3689.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. (2017). Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*.

DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, 73(3):415–434.

Diuk, C., Cohen, A., and Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th ICML*, pages 240–247.

Džeroski, S., De Raedt, L., and Driessens, K. (2001). Relational reinforcement learning. *Machine learning*, 43(1):7–52.

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

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. (2017). Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Filos, A., Lyle, C., Gal, Y., Levine, S., Jaques, N., and Farquhar, G. (2021). Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. In *International Conference on Machine Learning*, pages 3305–3317. PMLR.

Fujimoto, S., Meger, D., and Precup, D. (2021). A deep reinforcement learning approach to marginalized importance sampling with the successor representation. In *International Conference on Machine Learning*, pages 3518–3529. PMLR.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192.

Gershman, S. J., Pesaran, B., and Daw, N. D. (2009). Human reinforcement learning subdivides structured action spaces by learning effector-specific values. *The Journal of Neuroscience*, 29(43):13524–13531.

Gerstenberg, T., Goodman, N. D., Lagnado, D. A., and Tenenbaum, J. B. (2021). A counterfactual simulation model of causal judgments for physical events. *Psychological review*, 128(5):936.

Ghahramani, Z. and Wolpert, D. M. (1997). Modular decomposition in visuomotor learning. *Nature*, 386(6623):392–5.

Goddu, M. K. and Gopnik, A. (2020). Learning what to change: Young children use "difference-making" to identify causally relevant variables. *Developmental psychology*, 56(2):275.

Gopnik, A. and Wellman, H. M. (1994). The theory theory. In *An earlier version of this chapter was presented at the Society for Research in Child Development Meeting, 1991.* Cambridge University Press.

Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., and Farhadi, A. (2018). Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Goyal, A., Lamb, A., Gampa, P., Beaudoin, P., Levine, S., Blundell, C., Bengio, Y., and Mozer, M. (2020a). Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv*.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2020b). Recurrent independent mechanisms. *ICLR*.

Graybiel, A. M., Aosaki, T., Flaherty, A. W., and Kimura, M. (1994). The basal ganglia and adaptive motor control. *Science*, 265(5180):1826–1831.

Greff, K., van Steenkiste, S., and Schmidhuber, J. (2020). On the binding problem in artificial neural networks. *arXiv*.

Gupta, T., Mahajan, A., Peng, B., Böhmer, W., and Whiteson, S. (2021). Uneven: Universal value exploration for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 3930–3941. PMLR.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. (2018). Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6244–6251. IEEE.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Hansen, S., Dabney, W., Barreto, A., Van de Wiele, T., Warde-Farley, D., and Mnih, V. (2019). Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*.

He, F., Liu, T., and Tao, D. (2020). Why resnet works? residuals generalize. *IEEE transactions on neural networks and learning systems*, 31(12):5349–5362.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017). Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. (2019). Environmental drivers of systematicity and generalization in a situated agent. *arXiv preprint arXiv:1910.00571*.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. (2020). Environmental drivers of systematicity and generalization in a situated agent. *ICLR*.

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

Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K., et al. (2020). Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Huang, K., Wang, Y., Tao, M., and Zhao, T. (2020a). Why do deep residual networks generalize better than deep feedforward networks?—a neural tangent kernel perspective. *Advances in neural information processing systems*, 33:2698–2709.

Huang, W., Mordatch, I., and Pathak, D. (2020b). One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR.

Hunt, J., Barreto, A., Lillicrap, T., and Heess, N. (2019). Composing entropic policies using divergence correction. In *International Conference on Machine Learning*, pages 2911–2920. PMLR.

Jaakkola, T., Jordan, M., and Singh, S. (1993). Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems*, 6.

Jain, U., Weihs, L., Kolve, E., Rastegari, M., Lazebnik, S., Farhadi, A., Schwing, A. G., and Kembhavi, A. (2019). Two body problem: Collaborative visual task completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Janz, D., Hron, J., Mazur, P., Hofmann, K., Hernández-Lobato, J. M., and Tschiatschek, S. (2019). Successor uncertainties: exploration and uncertainty in temporal difference learning. *Advances in Neural Information Processing Systems*, 32.

Jara-Ettinger, J., Schulz, L. E., and Tenenbaum, J. B. (2020). The naive utility calculus as a unified, quantitative framework for action understanding. *Cognitive Psychology*, 123:101334.

Jing, L., Vincent, P., LeCun, Y., and Tian, Y. (2021). Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*.

Justesen, N., Torrado, R. R., Bontrager, P., Khalifa, A., Togelius, J., and Risi, S. (2019). Illuminating generalization in deep reinforcement learning through procedural level generation. *AAAI*.

Kabra, R., Zoran, D., Erdogan, G., Matthey, L., Creswell, A., Botvinick, M., Lerchner, A., and Burgess, C. P. (2021). Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *arXiv preprint arXiv:2106.03849*.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *ICML*, pages 1809–1818. PMLR.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T., van der Pol, E., and Welling, M. (2019). Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*.

Kolve, E., Mottaghi, R., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Konkle, T. and Alvarez, G. A. (2022). A self-supervised domain-general learning framework for human ventral stream representation. *Nature communications*, 13(1):491.

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., Tanaka, K., and Bandettini, P. A. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6):1126–1141.

Lampinen, A. K., Chan, S. C., Banino, A., and Hill, F. (2021). Towards mental time travel: a hierarchical memory for reinforcement learning agents. *arXiv*.

Lee, K., Lee, K., Shin, J., and Lee, H. (2020). Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*.

Li, Z., Zhang, Y., and Arora, S. (2020). Why are convolutional nets more sample-efficient than fully-connected nets? *arXiv preprint arXiv:2010.08515*.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*.

Logeswaran, L., Carvalho, W. T., and Lee, H. (2021). Learning compositional tasks from language instructions. In *Deep RL Workshop NeurIPS 2021*.

Lu, Y. and Lu, J. (2020). A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33:3094–3105.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2020). Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 04, pages 5125–5133.

Madan, K., Ke, N. R., Goyal, A., Schölkopf, B., and Bengio, Y. (2021). Fast and slow learning of recurrent independent mechanisms. *arXiv preprint arXiv:2105.08710*.

Marom, O. and Rosman, B. (2018). Zero-shot transfer with deictic object-oriented representation in reinforcement learning. In *NeurIPS*.

Merel, J., Botvinick, M., and Wayne, G. (2019). Hierarchical motor control in mammals and machines. *Nature communications*, 10(1):5489.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Momennejad, I., Russek, E. M., Cheong, J. H., Botvinick, M. M., Daw, N. D., and Gershman, S. J. (2017). The successor representation in human reinforcement learning. *Nature human behaviour*, 1(9):680–692.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., and Rezende, D. J. (2019). Towards interpretable reinforcement learning using attention augmented agents. *NeurIPS*.

Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*.

Oh, J., Guo, Y., Singh, S., and Lee, H. (2018). Self-imitation learning. *arXiv preprint arXiv:1806.05635*.

Oh, J., Singh, S., Lee, H., and Kohli, P. (2017a). Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR.

Oh, J., Singh, S., Lee, H., and Kohli, P. (2017b). Zero-shot task generalization with multi-task deep reinforcement learning. *ICML*, abs/1706.05064.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. (2018). Assessing generalization in deep reinforcement learning. *arXiv*.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., et al. (2020). Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Pirk, S., Khansari, M., Bai, Y., Lynch, C., and Sermanet, P. (2019). Online object representations with contrastive learning. *arXiv preprint arXiv:1906.04312*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355.

Reed, S., Sohn, K., Zhang, Y., and Lee, H. (2014). Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*.

Russell, S. J. and Zimdars, A. (2003). Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 656–663.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. (2015). Policy distillation. *arXiv preprint arXiv:1511.06295*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. (2018). Relational recurrent neural networks. *Advances in neural information processing systems*, 31.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2019). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *ArXiv*, abs/1912.01734.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308.

Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339.

Sohn, S., Oh, J., and Lee, H. (2018). Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. *NeurIPS*.

Sohn, S., Woo, H., Choi, J., and Lee, H. (2021). Meta reinforcement learning with autonomous inference of subtask dependencies. *ICLR*.

Sohn, S., Woo, H., Choi, J., Qiang, L., Gur, I., Faust, A., and Lee, H. (2022). Fast inference and transfer of compositional task structures for few-shot task generalization. In *Uncertainty in Artificial Intelligence*, pages 1857–1865. PMLR.

Spelke, E. S. (1990). Principles of object perception. *Cognitive science*, 14(1):29–56.

Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental science*, 10(1):89–96.

Sperling, G. (2018). A brief overview of computational models of spatial, temporal, and feature visual attention. *Invariances in human information processing*, pages 143–182.

Sperling, G. and Weichselgartner, E. (1995). Episodic theory of the dynamics of spatial attention. *Psychological review*, 102(3):503.

Stachenfeld, K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643–1653.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999a). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Sutton, R. S., Precup, D., and Singh, S. (1999b). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Taylor, M. E., Stone, P., and Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(9).

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30. IEEE.

Tomov, M. S., Schulz, E., and Gershman, S. J. (2021). Multi-task reinforcement learning in humans. *Nature Human Behaviour*, 5(6):764–773.

Tsividis, P. A., Loula, J., Burga, J., Foss, N., Campero, A., Pouncy, T., Gershman, S. J., and Tenenbaum, J. B. (2021). Human-level reinforcement learning through theory-based modeling, exploration, and planning. *arXiv preprint arXiv:2107.12544*.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *AAAI conference on artificial intelligence*.

Van Hoof, H., Hermans, T., Neumann, G., and Peters, J. (2015). Learning robot in-hand manipulation with tactile features. In *International Conference on Humanoid Robots*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv*.

Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J., and Levine, S. (2020). Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pages 1439–1456. PMLR.

Veeriah, V., Hessel, M., Xu, Z., Rajendran, J., Lewis, R. L., Oh, J., van Hasselt, H. P., Silver, D., and Singh, S. (2019). Discovery of useful questions as auxiliary tasks. *Advances in Neural Information Processing Systems*, 32.

Veeriah, V., Zheng, Z., Lewis, R., and Singh, S. (2022). Grasp: Gradient-based affordance selection for planning. *arXiv preprint arXiv:2202.04772*.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Watters, N., Matthey, L., Bosnjak, M., Burgess, C. P., and Lerchner, A. (2019). Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*.

Wente, A. O., Kimura, K., Walker, C. M., Banerjee, N., Fernández Flecha, M., MacDonald, B., Lucas, C., and Gopnik, A. (2019). Causal learning across culture and socioeconomic status. *Child development*, 90(3):859–875.

Xu, D., Martín-Martín, R., Huang, D.-A., Zhu, Y., Savarese, S., and Fei-Fei, L. F. (2019). Regression planning networks. In *Advances in Neural Information Processing Systems*.

Xu, T., Zhu, H., and Paschalidis, I. C. (2020). Learning parametric policies and transition probability models of markov decision processes from data. *European Journal of Control*.

Ye, Y., Gandhi, D., Gupta, A., and Tulsiani, S. (2020). Object-centric forward modeling for model predictive control. In *Conference on Robot Learning*.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., et al. (2018). Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*.

Zaragoza, J. H., Morales, E. F., et al. (2010). Relational reinforcement learning with continuous actions by combining behavioural cloning and locally weighted regression. *Journal of Intelligent Learning Systems and Applications*, 2(02):69.

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. (2021). Learning invariant representations for reinforcement learning without reconstruction. *ICLR*.

Zhang, A., Sukhbaatar, S., Lerer, A., Szlam, A., and Fergus, R. (2018). Composable planning with attributes. In *International Conference on Machine Learning*, pages 5842–5851. PMLR.

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. (2017). Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378. IEEE.

Zhu, Y., Gordon, D., Kolve, E., Fox, D., Fei-Fei, L., Gupta, A., Mottaghi, R., and Farhadi, A. (2017). Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision*, pages 483–492.

Zhu, Z., Lin, K., Jain, A. K., and Zhou, J. (2020). Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*.

Zoran, D., Kabra, R., Lerchner, A., and Rezende, D. J. (2021). Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10439–10447.