ROMA: A Relational, Object-Model Learning Agent for Sample-Efficient Reinforcement Learning

Wilka Carvalho[1,*] **Anthony Liang**[1] **Kimin Lee**[3,†] **Sungryull Sohn**[1]
Honglak Lee[1,4] **Richard L. Lewis**[2] **Satinder Singh**[1,5]
[1]University of Michigan, Dept. of Computer Science
[2]University of Michigan, Dept. of Psychology
[3]UC Berkeley [4]Google Brain [5]DeepMind

Abstract

Sequential decision-making tasks that require relating and using multiple novel objects pose significant sample-efficiency challenges for agents learning from sparse task rewards. In this work, we begin to address these challenges by leveraging an agent's object-interactions to define an auxilliary task that enables sample-efficient reinforcement learning (RL) of such tasks. To accomplish this, we formulate ROMA: a relational reinforcement learning agent that learns an object-centric forward model during task learning. We find that this enables it to learn object-interaction tasks much faster than other relational RL agents with alternative auxiliary tasks for driving good object-representation learning. In order to evaluate the performance of our agent, we introduce a set of object-interaction tasks in the AI2Thor virtual home environment that require relating and interacting with multiple objects. By comparing against an agent equipped with ground-truth object-information, we find that learning an object-centric forward model best closes the performance gap, achieving $\geq 80\%$ of its sample-efficiency on 7 out of 8 tasks, with the next best method doing so on 2 out of 8 tasks. Additionally, we find that our object-model best captures interesting object information such as category, specific object state, and relationships among objects.

1 Introduction

Consider a robotic home-aid agent given the task of cooking a potato. In order to perform this task, it needs to transport both a potato and a pot to the stove and place them appropriately so that turning on the stove will heat the pot, which in turn will cook the potato. Learning to perform such object-interaction tasks are challenging for a number of reasons: (A) the agent needs to transport objects to each other so they can be used together; (B) it needs to learn a view-invariant representation to facilitate object-recognition and object-selection; (C) it needs to learn to reason over multiple objects; (D) it needs to recognize and use combined objects. In our example, the agent must recognize the pot and potato from different viewpoints, pick them up, transport them to the stove, place the potato in the pot and turn on the stove, recognizing that this will heat both objects. In learning and performing such skills, manipulating object relationships and appropriately responding to them is paramount.

The aim of our work is to develop an autonomous reinforcement learning (RL) agent that learns such object-interaction tasks from sparse task rewards in a sample-efficient manner. To create interesting learning challenges for the agent, we adopt the virtual home-environment AI2Thor [19] (or *Thor*). Thor is an open-source environment that is high-fidelity, 3D, partially observable, and enables explicit atomic object-interactions, i.e. an agent interacts with objects by selecting (*object, action*) tuples.

[*]Corresponding author: wcarvalh@umich.edu. [†]Work done while at the University of Michigan.

Preprint. Under review.

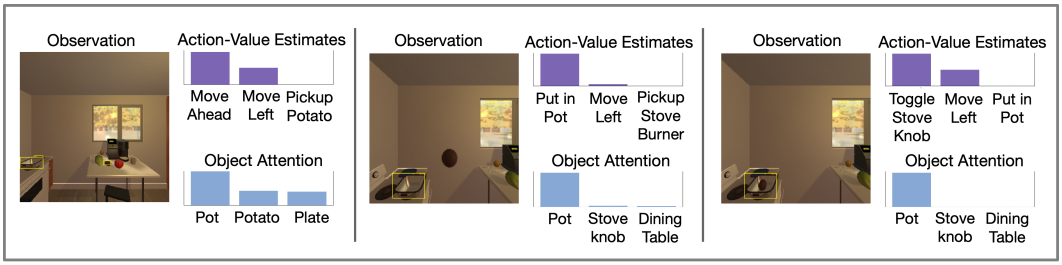


Figure 1: We present part of the trajectory when ROMA completes the "Cook Potato on Stove" task. We visualize the top-3 action-values and attention-weights at 3 steps along its trajectory. We see that it moves towards the table with the potato. Once its at the stove with the potato, we see that it continues to attend to the pot as it places the potato inside and decides to turn the stove knob on.

Thor poses significant learning challenges due to a large action space induced by many available object-interaction and navigation actions, and a relatively complex first-person visual input. No work has yet learned sparse-reward object-interaction tasks from scratch in this domain. Prior work has relied on imitation learning from expert demonstrations or leveraged ground-truth object-information [11, 16, 35, 51]. The approach we explore here assumes neither. Rather, we hypothesize that we can achieve sample-efficient learning by leveraging a method for unsupervised object-representation learning during decision-making. The specific goal we have is that the agent should achieve a 90%+ success rate after collecting $\leq$ 500K samples of experience with only a task-completion reward.

Towards this end, we present *ROMA*, a Relational, Object-Model Learning Agent. Drawing on literature in developmental psychology indicating that infants leverage object-detection [37] and object-interaction [30] to learn about the world, we equip our agent with an object-detector and have it predict the consequences of its interactions with an object-centric model. We hypothesize that this auxilliary task will provide a rich signal for learning about objects. To enable our agent to leverage object-relations into its decision-making, we equip it with an attention-based relational module [39, 50], which also enables its object-model to incorporate object-relations into its predictions. Without ground-truth information to identify objects, the agent must learn object-representations that are both *invariant* across object-views and *discriminative* across object-states so that correct actions are chosen as object-states change over a task. To address this, ROMA learns its object-model with a contrastive learning loss [1] that seeks to separate object-interactions based on their resultant transition. We present a partial trajectory of ROMA completing a task in figure 1, along with visualizations of ROMA's action-value estimates and how it attends to objects.

The work we present here makes three contributions. Our first and primary contribution is the ROMA agent, and the demonstration that it is possible to achieve sample-efficient learning in high-fidelity, 3D, object-interaction domains without access to expert demonstrations or ground-truth object-information by imbuing an RL agent with an object-centric inductive bias for model-learning and decision-making. Second, we present a new relational reinforcement learning agent, *Relational Object-DQN* (§4.1) that is a strong baseline for object-interaction tasks where agents interact with objects via (object-image-patch, action) tuples. This baseline enables us to remove the assumptions prior work have made of object-id knowledge when deciding object-interactions, and forms ROMA's foundation. Third, we present a simple and effective method for unsupervised object-representation learning via our object-centric model (§4.2).

By evaluating ROMA against Relational Object-DQN combined with alternative unsupervised object-representation learning methods, we show that ROMA best closes the performance gap to an agent with ground-truth object-information. We find that ROMA achieves $\geq 80\%$ of the sample-efficiency of a ground-truth-supplied agent on 7 out of 8 tasks, with the next best agent doing so on only 2 out of 8 tasks. Through a detailed analysis of the learned object-representations, we confirm quantitatively that the object-model best captures interesting object information such as category, specific object state, and relationships among objects.

2 Related work

Egocentric agents in simulated, 3D domains. Most prior work here has focused on navigation problems [3, 8, 24, 27, 34, 45, 48, 52]. For example, Mirowski et al. [24] use an LSTM [14] to learn to navigate complex enviroments in DeepMind Labs [3], and Wortsman et al. [45] learned to navigate to novel objects in Thor [19] with gradient-based meta-reinforcement learning [10]. In the Thor domain [19], Jain et al. [16] formulate a multiagent reinforcement learning problem [22] where two agents discover a basic communication protocol in order to coordinate picking up a large object, and Gordon et al. [11] developed a hierarchical reinforcement learning agent [2] with external memory [44] for visual question-answering. In the Kitchen 3D domain, Xu et al. [46] also tackle learning household tasks by learning to plan a series of high-level goals to accomplish basic cooking tasks. In contrast to our work, all three provide strong supervision from expert trajectories.

The work most closely related to ours is Oh et al. [28] (in Minecraft) and Zhu et al. [51] (in Thor). Both develop a hierarchical reinforcement learning agent where a meta-controller provides goal object-interactions for a low-level controller to complete. Oh et al. [28] focuses on executing instructions and generalizing to novel instructions in a multitask setting whereas Zhu et al. [51] focuses on learning in a single-task setting. Zhu et al. [51] assumes an oracle low-level policy that has access to ground-truth object-information (such as whether objects are open/closed, on/off, etc.). Both provide agents with knowledge of all objects in the state and both assume lower-level policies pretrained to navigate to and select objects. In contrast, we do not provide the agent with any ground-truth object information; nor do we pretrain navigation to or selection of objects.

Improving sample efficiency via an object-centric forward model. An increasingly popular direction in reinforcement learning is to learn an object-centric forward model [18, 40, 43, 49]. Watters et al. [43], Ye et al. [49] showed this enabled more sample-efficient learning via model-predictive control [5] and Veerapaneni et al. [40] showed that this enabled generalization to unseen object combinations. However, the aforementioned work focused on relatively simple scenes with basic visual shapes such as rectangles and circles. In this work, we focus on a large class of diverse 3D object categories that change in state and can effect eachother in interesting ways–a sink can be used to fill a cup with water, a stove can heat a pan, etc. Most similar to our object-model is the Contrastive Structured World Model (CSWM) [18], which employs contrastive learning to learn an object-model built upon a graph neural network, whereas ours is built upon self-attention. Despite similar mechanics, the goals were quite different as they applied their model towards video prediction while we do so towards reinforcement learning.

3 Egocentric Reinforcement Learning for Object-Interaction Tasks

Observations. We focus on an embodied agent that has a 2D camera for experiencing *egocentric* observations $s^{\texttt{ego}}$ of the environment. Our agent also has a pretrained vision system that enables it to extract image-patches corresponding to the objects in its observation $s^o = \{o_i\}$ (but not object labels or identifiers). We also assume that the agent has access to its (x, y, z) location and body rotation $(\varphi_1, \varphi_2, \varphi_3)$ in a global coordinate frame, $s^{\texttt{loc}} = (x, y, z, \varphi_1, \varphi_2, \varphi_3)$ (though this assumption could be relaxed in future work). We treat the egocentric observation, location, and body rotation information as the *context* of the objects and collectively refer to them as $s^\kappa = \{s^{\texttt{ego}}, s^{\texttt{loc}}\}$.

State. The agent treats observations as state $s = s^o \cup s^\kappa$. We acknowledge that since the environment is partially observable, some mechanism for maintaining history (e.g. recurrence) is required as tasks grow in complexity, but leave this for future work.

Actions. The agent interacts with objects by selecting (object-image-patch, interaction) pairs $a = (o_c, b)$, where o_c corresponds to the *chosen* image-patch and $b \in \mathcal{A}_I$ corresponds to an available object interaction that the agent can perform. For example, the agent can turn on the stove by selecting the image-patch containing the stove-knob and the *Turn on* interaction (see Figure 2 for a diagram). In addition to object-interactions, the agent can select navigation actions $b \in \mathcal{A}_N$ that do not require it to select an image-patch.

Reward. We consider a single-task setting where the agent receives a terminal reward of 1 if it completes the task. Due to the large action-space our agent acts in, this leads the agent to face a *sparse reward* problem.

Learning to act. During learning, the agent 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 π thereafter. It leverages $\widehat{Q}$ by behaving according to a policy that is ϵ-greedy w.r.t. $\widehat{Q}(s,a)$: i.e. $\pi(a|s) = \arg\max_a \widehat{Q}(s,a)$ with probability $1-\epsilon$ and is uniformly random otherwise. We estimate $\widehat{Q}(s,a)$ as a Deep Q-Network (DQN) by minimizing the following temporal difference objective:

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

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 θ. 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 [25]. To stabilize learning, we use Double-Q-learning [38] to choose the next action: $a_{t+1} = \arg\max_a \widehat{Q}(s_{t+1}, a; \theta)$.

4 ROMA: Relational, Object-Model Learning Agent

ROMA is composed of a base relational architecture that enables object-interaction with unlabeled object-image-patches (Relational Object-DQN) and a contrastive object-model learning method for unsupervised object-representation learning. We first present our Relational Object-DQN in §4.1, and then our contrastitve object-model learning method in §4.2. See Figure 2 for an overview of the full architecture.

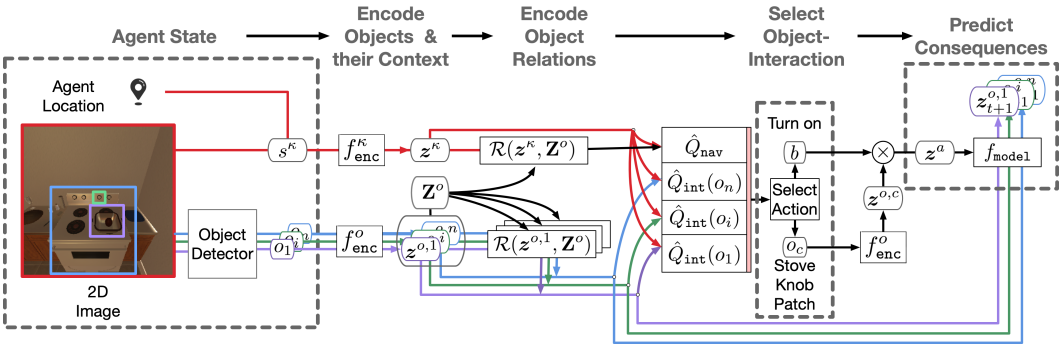


Figure 2: Full architecture and processing pipeline of ROMA. A scene is broken down into object-image-patches $\{o_j\}$ (e.g. of a pot, potato, and stove knob). The scene is combined with the agent's location to define the *context* of the objects, s^κ. The objects $\{o_j\}$ and their context s^κ are processed by different encoding branches and then recombined by a relational module $\mathcal{R}$ to produce inputs for computing Q-value estimates. Actions are selected as (object-image-patch, base action) pairs $a = (o_c, b)$. The agent then predicts the consequences of its interactions with an object-model $f_{\texttt{model}}$.

4.1 Relational Object-DQN

As we focus on object-interaction tasks that require relating multiple objects (e.g., the agent must place bread into a toaster to toast it), we hypothesize that an inductive bias for relational reasoning will facilitate task-learning. Recently, Zambaldi et al. [50] showed that self-attention [39] on entities in an observation can facilitate capturing predicate relationships, such as whether bread is in a toaster. We build on this line of research and formulate a Relational Object-DQN that uses self-attention over object-image-patches to capture and leverage object-relationships when deciding actions.

Computing object relations. There is contextual-state information encoding function $\boldsymbol{z}^\kappa = f^\kappa_{\texttt{enc}}(s^\kappa) \in \mathbb{R}^{d_\kappa}$ and an object-encoding function $\boldsymbol{z}^{o,i} = f^o_{\texttt{enc}}(o_i) \in \mathbb{R}^{d_o}$. Given n observed object-image-patches $s^o = \{o_i\}_{i=1}^n$, we encode each and concatenate them into a matrix, $\boldsymbol{Z}^o = \left[\boldsymbol{z}^{o,i}\right]_i \in \mathbb{R}^{n \times d_o}$. In order to model relations between objects, we apply a self-attention mechanism between encodings of object-image-patches. Specifically, given an object-image-patch encoding $\boldsymbol{z}^{o,i}$, we compute object-relations $\mathcal{R}(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^o)$ as follows:

$$\mathcal{R}(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^o) = \sum_j \alpha^{o,i}_j \boldsymbol{z}^{o,j} = \underbrace{\sigma\left(\frac{\left(\boldsymbol{Z}^o W^k\right)\left(W^q \boldsymbol{z}^{o,i}\right)}{\sqrt{d_k}}\right)^\top}_{\alpha^{o,i}} Z^o, \tag{2}$$

where $\sigma(\cdot)$ if the softmax function, $W^k \in \mathbb{R}^{d_o \times d_k}$, and $W^q \in \mathbb{R}^{d_k \times d_o}$. Intuitively, $\mathcal{R}(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^o)$ attends over object-encodings relevant when choosing object-interactions on o_i.

Computing Q-values. Our DQN formulation can be seen as a successor to Zhu et al. [51] as they also tackle object-interaction tasks with DQN. However, they assumed access to ground-truth object-information and computed Q-value-estimates for a fixed set of object-interactions based on a priori knowledge of object-ids. We remove this assumption, and instead condition the Relational Object-DQN on encodings of object-image-patches. We estimate Q-values as follows:

$$\widehat{Q}_{\texttt{int}}(o_i) = f_{\texttt{int}}([\boldsymbol{z}^{o,i}, \mathcal{R}(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^o), \boldsymbol{z}^\kappa]), \quad \widehat{Q}_{\texttt{nav}} = f_{\texttt{nav}}([\boldsymbol{z}^\kappa, \mathcal{R}(\boldsymbol{z}^\kappa, \boldsymbol{Z}^o)]),$$
$$\widehat{Q} = [\widehat{Q}_{\texttt{nav}}, \widehat{Q}_{\texttt{int}}(o_1), \ldots, \widehat{Q}_{\texttt{int}}(o_n)], \tag{3}$$

where $\widehat{Q}_{\texttt{int}}(o_i) \in \mathbb{R}^{|\mathcal{A}_I|}$ corresponds to Q-value estimates for object-interactions with object-image-patch o_i and $\widehat{Q}_{\texttt{nav}} \in \mathbb{R}^{|\mathcal{A}_N|}$ corresponds to Q-value estimates for navigation actions. This enables us to compute estimates for a variable number of *unlabeled* objects via their image patches.

4.2 Contrastive Object-Model Learning

To facilitate deriving our objective function, consider the global set of objects $\{o^g_{t,i}\}^m_i$, where m is the count of all present objects. 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 $\rho^g_{t,i}$ for short). The set of indices corresponding to visible objects at time t is $v_t = \{i : \rho^g_{t,i} \text{ is visible at time } t\}$, so the set of observed object-image-patches is $s^o_t = \{o_{t,j}\} = \{\rho^g_{t,i}\}_{i \in v_t}$. We will use this formulation to ease referencing image-patches of the same object across time-steps. Note that our method does not lose its generality as we encode object-image-patches and find their matches across time-steps using the learned encodings.

Objective function. Our agent assumes that the contextual state-information s^κ_t remains approximately constant when object-interactions are performed $a_t \in \mathcal{A}_I \times \mathcal{O}_t$. Thus, when modeling state-transition due to object-interactions, it only models $p(s^o_{t+1}|s_t, a_t)$. To encourage the agent's object-representations to capture useful information such as object-label or object-state, we have the agent model object dynamics strictly from object-image-patches. Assuming the probability of each object's next state is conditionally independent given the current set of objects and the action taken, we get

$$\begin{aligned} p(s^o_{t+1}|s_t, a_t) &= p(s^o_{t+1}|s^o_t, a_t) \\ &= p(\{\rho^g_{t+1,i}\}_{i \in v_{t+1}}|s^o_t, a_t) \\ &= \prod_{i \in v_{t+1}} p(\rho^g_{t+1,i}|s^o_t, a_t). \end{aligned} \tag{4}$$

This motivates the objective function for our object-centric forward model:

$$\begin{aligned} \mathcal{L}_{\texttt{model}} &= \mathbb{E}_{s_t, a_t, s_{t+1}} \left[-\log p(s^o_{t+1}|s^o_t, a_t) \right] \\ &= \mathbb{E}_{s_t, a_t, s_{t+1}} \left[-\sum_{i \in v_{t+1}} \log p(\rho^g_{t+1,i}|s^o_t, a_t). \right] \end{aligned} \tag{5}$$

Contrastive learning framework. We leverage contrastive learning by treating the learning of $p(\rho^g_{t+1,i}|s^o_t, a_t)$ as a classification problem. (See Appendix §A.1 for background on contrastive learning). Specifically, for an object-image-patch $\rho^g_{t,i}$, we define its *anchor* (or query) by an *object-centric forward model* $F(s^o_t, \rho^g_{t,i}, a_t)$ which takes in a set of object-image-patches s^o_t, the object-image-patch $\rho^g_{t,i}$, and an object-interaction a_t to produce the resultant encoding for $\rho^g_{t+1,i}$. We define the *positive* (or correct answer) as the encoding of a visible object-image-patch at the next time-step with the highest cosine similarity to the original encoding: $\boldsymbol{z}^{o,i}_+ = \arg\max_{\boldsymbol{z}^{o,j}_{t+1}} \cos(\boldsymbol{z}^{o,i}_t, \boldsymbol{z}^{o,j}_{t+1})$. We can then select K random object-encodings $\{\boldsymbol{z}^{o,i}_{k,-}\}^K_{k=1}$ as *negatives* (or incorrect answers). This leads to:

$$p(\rho^g_{t+1,i}|s^o_t, a_t) = \frac{\exp(F(s^o_t, \rho^g_{t,i}, a_t)^\top \boldsymbol{z}^{o,i}_+)}{\exp(F(s^o_t, \rho^g_{t,i}, a_t)^\top \boldsymbol{z}^{o,i}_+) + \sum_k \exp(F(s^o_t, \rho^g_{t,i}, a_t)^\top \boldsymbol{z}^o_{k,-})}. \tag{6}$$

To learn an action encoding $\boldsymbol{z}^a_t$ for action $a_t = (b_t, o_{t,c})$, following Oh et al. [26], Reed et al. [33], 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

$$\boldsymbol{z}^a_t = W^o \boldsymbol{z}^{o,c}_t \odot W^b b_t, \tag{7}$$

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. Reusing $\mathcal{R}(\boldsymbol{z}_t^{o,i}, \boldsymbol{Z}_t^o)$ as input to our foward model, we arrive at

$$F(s_t^o, \rho_{t,i}^g, a_t) = f_{\texttt{model}}([\boldsymbol{z}_t^{o,i}, \mathcal{R}(\boldsymbol{z}_t^{o,i}, \boldsymbol{Z}_t^o), \boldsymbol{z}_t^a]) \tag{8}$$

In practice, $f_{\texttt{model}}$ is a small 1- or 2-layer neural network on top of the existing Relational Object-DQN, making this method compact and simple to implement. Our final objective is

$$\mathcal{L} = \mathcal{L}_{\texttt{DQN}} + \beta^{\texttt{model}} \mathcal{L}_{\texttt{model}}. \tag{9}$$

5 Experiments

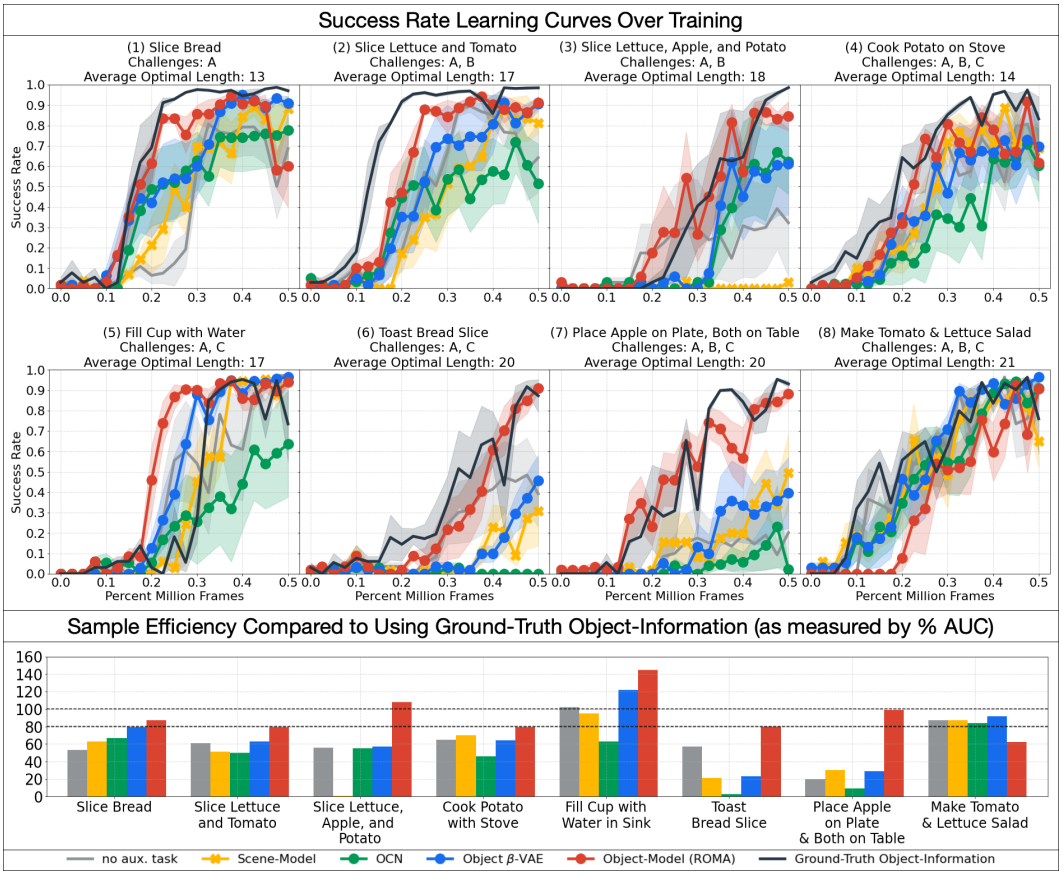


Figure 3: **Top-panel**: we present the success rate over learning for competing auxilliary tasks. These tasks exhibit varying combinations of challenges (e.g., A: view-invariance, B: reasoning over multiple objects, C: using combination of objects); see Section 5.2 for details. We seek a method that best enables an RL agent to obtain the sample-efficiency it would from using ground-truth object-information (black). **Bottom-panel**: by measuring the % AUC achieved by each agent w.r.t to the agent with ground-truth information, we can more precisely measure how close each method is to ground-truth performance. We find ROMA (red) best closes the performance gap, achieving $\geq 80\%$ on $7/8$ tasks. When tasks only require view-invariance (challenge A; e.g., task 1), ROMA only learns slighly faster than competing methods. However, when tasks require more objects (challenge B; e.g., tasks 3, 7) or involve placing objects in each other for further usage (challenge C; e.g., tasks 5-7), the gap grows. We hypothesize that this is due to ROMA's ability to better capture ground-truth information, which facilitates differentiating multiple objects and relating them pre/post object-placement. We quantitatively confirm these hypotheses in Table 1.

We evaluate ROMA on several object-interaction kitchen tasks in the Thor environment [19]. To gain understanding of the value of the specific object-model contrastive learning auxiliary task proposed above, we compare ROMA to four baselines that combine Relational-Object-DQN with alternative representation learning methods, and Relational-Object-DQN without an auxiliary task.

5.1 Representation Learning Baselines

Given that Zambaldi et al. [50] already showed the sample-efficiency utility of a relational inductive bias[2], we create two baselines to lower-bound and upper-bound performance: **"Relational Object-DQN"** and **"Relational Object-DQN + Ground-Truth Object-Information"**. For ground-truth object-information, we provide object-category (i.e. an object's type), object-id (i.e. an object's token), object-state (e.g. on, off, etc.), and object-containment (i.e. is the object in/on another object and if so, which object). This enables us to measure how much progress each method makes towards learning ground-truth object-information that is useful for an RL agent to incorporate into its decision-making.

Is it better to leverage object-interactions for a forward-model at the object-level or the scene-level? To answer this, we create a **Scene Model** baseline by adapting Oord et al. [29] to create a forward-model that predicts embeddings of entire scene-images.

How does our object-model compare to other strong unsupervised object-representation learning baselines? To answer this, we compare against an **Object β-VAE** baseline that that uses a β-Variational Autoencoder [4], and a **OCN** baseline that uses an Object-Contrastive Network [32]. We select a β-VAE because prior work has found it can capture ground-truth latent factors (e.g., object shape and color) [4, 6, 15, 17]. We select OCN due to its strong performance in learning view-invariant object-representations in real-world settings.

5.2 Experimental Settings

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 3 seeds. To quantitatively study sample-efficiency, we compare each model to "Relational Object-DQN + Ground-Truth Object-Information", by computing what % of its AUC each method achieved using the mean success rate of each method.

Tasks. We construct tasks that study the challenges of object-interaction tasks along the following three aspects (note: these are not mutually exclusive):

> **Challenge A:** the need for view-invariance (e.g. recognizing a knife across angles),
>
> **Challenge B:** the need to reason over ≥ 3 objects,
>
> **Challenge C:** the need to recognize and use *combined* objects (e.g. filling a cup with water in the sink or toasting bread in a toaster).

All tasks require the agent navigate to objects and transport them to each other. Across tasks, the agent's spawning location is randomized from 81 grid positions. The agent recieves reward 1 if a task is completed successfully and a time-step penaltiy of -0.04. We present the 8 most challenging tasks and results on easier tasks in Appendix E, along with descriptions of tasks in section D.

5.3 Results and Analysis

Performance Summary. We confirm that leveraging object-interactions for predictions at the object-level vs. scene-level better supports increased sample-efficiency. Looking at the bottom panel in Figure 3, we see that predicting at the scene-level was often comparable or worse to no auxiliary representation learning. This is evidence that making predictions about objects of the world might be key to improved sample-efficiency when learning representations with a forward-model. Looking again at the bottom panel in Figure 3, we see that the β-VAE and OCN performed comparably, though the β-VAE did slightly better. However, the β-VAE was only able to achieve $\geq 80\%$ ground-truth sample-efficiency on $2/8$ tasks, whereas ROMA was able to on $7/8$ tasks.

Performance Analysis. To understand the source of task-difficulty, we analyze each agent's object-encoding function $f^o_{\texttt{enc}}$. We freeze their parameters, and add a linear layer to predict some of the ground-truth object-information features of our upper-bound model using a dataset of collected object-interactions. This provides insight into whether this information is captured in the output of $f^o_{\texttt{enc}}$. The dataset contains (s, a, s') tuples from an oracle agent we create to complete various tasks. For example, we generated interactions from variants of "*Cook X with Stove*", where $X \in$ {Potato, Potato Slice, Cracked Egg}. We divide the object-features as follows. *Category* is a multi-class label indicating an object's category. The rest are binary labels. *Attribute* indicates whether

[2]See appendix A.2 for details on how our approach differs.

an object can be picked up, closed, etc.; *Object-state* indicates whether objects *are* closed, turned on, etc.; and *Containment Relationship* indicates if an object is inside another object or has another object inside of it. For each set, we present the mean average precision and standard error for each method across all 8 tasks in Table 1. For more details on training, see Appendix D.3.

We found that task-length, the need to relate ≥ 3 objects (challenge B), and the need to represent objects within objects (challenge C) were the best indicators of a task's difficulty. **Challenge B** was present in tasks like "Slice Lettuce, Apple, and Potato", and "Place Apple on Plate, Both on Table", where ROMA significantly outperformed other models. In Table 1, we see that ROMA best learns to represent categories, and we hypothesize that this enables it to differentiate a greater number of objects. **Challenge C** was present in tasks 4-8, (e.g. "Fill Cup with Water" and "Toast Bread Slice"), where ROMA achieved $\geq\approx 20\%$ *better* sample-efficiency than the next best method on $4/5$ of these tasks. Looking again at Table 1, we see that ROMA best captures the combination of object-state and containment relationships. When combined with its category representing abilities, we hypothesize that this enables it to best recognize when it has placed objects inside other objects, and when they experience subsequent changes in object-state (e.g. when bread is cooked inside a toaster). Interestingly, we find ROMA outperforms ground-truth information for "Fill Cup with Water" – we hypothesize that this might be due to its ability to capture temporal information that is hard to specify a priori. Together, these results indicate that ROMA best captures ground-truth object-information useful for decision-making.

Architecture Ablation. We do a brief comparison to Zhu et al. [51], which also tackled object-interaction tasks in Thor but did not rely on a relational architecture or auxilliary tasks for representation learning. Instead, they provided DQN with object-ids and relied on imitation learning. We present two tasks in figure 4 where we train their architecture, and a variant that replaces object-ids with object-image-patches. On the left panel, we show that without good representation learning, learning from object-image-patches is challenging. On the right panel, we show that simply providing ground-truth object-ids without other information (such as object-state) is insufficient to learn tasks. We note that ROMA was able to excel in both.

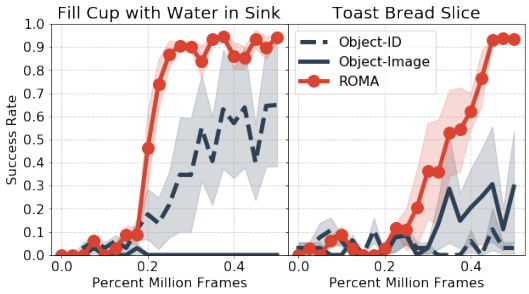


Figure 4: Comparison of conditioning DQN on object-ids [51] vs. object-image-patches. On the left panel, we see that learning from object-image-patches is challenging. On the right, we see that object-ids alone can be insufficient.

6 Conclusion

With ROMA, we have shown that learning an object-centric model in tandem with a relational inductive bias for decision-making 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 object-representation learning baselines, we have shown that our object-centric model is able to best capture ground-truth object information such as object categories, states of objects, and the presence of interesting object relationships. Given the results presented so far, there are a number of interesting future directions to take this work. For

Model	Category	Attribute	Object-State	Containment Relationship
Scene Model	74.5 ± 4.4	84.3 ± 2.4	78.2 ± 8.2	85.8 ± 4.3
OCN	39.2 ± 8.2	80.3 ± 3.4	66.5 ± 8.5	69.1 ± 9.0
Object β-VAE	60.8 ± 5.8	80.3 ± 2.3	60.7 ± 7.5	73.4 ± 8.2
Object-Model (ours)	$\mathbf{88.6 \pm 3.5}$	$\mathbf{92.0 \pm 0.7}$	$\mathbf{98.6 \pm 0.3}$	$\mathbf{94.3 \pm 0.6}$

Table 1: We study how well different unsupervised learning methods learn object-features (see text below for details). We find our Object-Model best differentiates object-categories (e.g. toaster, microwave, or pot), object-state (e.g. turned on, opened, etc.), and can recognize when objects are in each other. Together, these validate that learning with an object-model facilitates differentiating categories across states and interactions. These results, in tandem with our sample-efficiency results in Figure 3, are quantitative evidence that ROMA better captures ground-truth object-information useful for decision-making.

example, one could extend ROMA to a long-horizon setting, where, instead of toasting bread, the agent has to prepare breakfast. This would introduce the more challenging problem of exploration. Given that ROMA learns a forward model, it might be possible to leverage curiosity-driven learning [31] to address this challenge. We believe ROMA and the components that power it—object-attention and an object-centric model– are promising steps towards agents that can efficiently learn complex object-interaction tasks.

Broader Impact

ROMA and general autonomous robotic agents that can complete object-interaction tasks have the potential to revolutionize assisitive technology and in-home automation. For example, nursing robots can serve as healthcare workers in hospitals [7], distributing objects such as trays and medicine; or home-aid robots can help in elder care [20], helping with object-interaction household chores such as furniture rearrangement and food preparation. This kind of technology is especially helpful as elder care is challenging to provide and the elderly are likely to live far away from their families and from where they grew up most of their lives [9]. Autonomous robotic agents that can perform object-interaction tasks could potentially be provided by the government as an added resource for social welfare.

While autonomous robotic agents have great potential for good, agents such as ROMA that learn with reinforcement learning should be treated with caution. Object-interact tasks with sparse task rewards as we consider in this setting can potentially lead an agent to explore too many actions which may be harmful to those they care for. This is why it's important that agents have *some* built in knowledge that enables them to both learn tasks faster and to avoid taking negative actions. Additionally, it is important that we specify task rewards *correctly*, as prior research has shown that reward miss-specification can have negative consequences such as wire-heading [47]. One possible negative manifestation could arise if an autonomous agent that learns a model, such as ROMA, were designed to get positive intrinsic reward when it completed object-interactions that improved its model. ROMA's ability to learn ground-truth object-information from object-interactions is exciting; however, unfettered object-interaction in a real-world setting can be dangerous for those nearby. We wouldn't want an agent that tries to cook a metallic object such as a spoon in a microwave, or that tries to learn about the consequences of slicing arbitrary objects.

References

[1] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, 2019.

[2] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[3] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[4] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[5] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[6] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

[7] Eftychios G Christoforou, Andreas S Panayides, Sotiris Avgousti, Panicos Masouras, and Constantinos S Pattichis. An overview of assistive robotics and technologies for elderly care. In *Mediterranean Conference on Medical and Biological Engineering and Computing*, pages 971–976. Springer, 2019.

[8] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2054–2063, 2018.

[9] N Douthit, Sakal Kiv, Tzvi Dwolatzky, and Seema Biswas. Exposing some important barriers to health care access in the rural usa. *Public health*, 129(6):611–620, 2015.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. JMLR. org, 2017.

[11] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4089–4098, 2018.

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

[13] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2 (5):6, 2017.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[15] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pages 1878–1889, 2017.

[16] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6689–6699, 2019.

[17] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

[18] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[19] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[20] Minmin Leng, Peng Liu, Ping Zhang, Mingyue Hu, Haiyan Zhou, Guichen Li, Huiru Yin, and Li Chen. Pet robot intervention for people with dementia: A systematic review and meta-analysis of randomized controlled trials. *Psychiatry research*, 271:516–525, 2019.

[21] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[22] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[24] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[26] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

[27] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *arXiv preprint arXiv:1605.09128*, 2016.

[28] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2661–2670. JMLR. org, 2017.

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[30] Carolyn F Palmer. The discriminating nature of infants' exploratory actions. *Developmental Psychology*, 25(6):885, 1989.

[31] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

[32] Sören Pirk, Mohi Khansari, Yunfei Bai, Corey Lynch, and Pierre Sermanet. Online object representations with contrastive learning. *arXiv preprint arXiv:1906.04312*, 2019.

[33] Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*, pages 1431–1439, 2014.

[34] William B Shen, Danfei Xu, Yuke Zhu, Leonidas J Guibas, Li Fei-Fei, and Silvio Savarese. Situational fusion of visual representation for visual navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2881–2890, 2019.

[35] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *ArXiv*, abs/1912.01734, 2019.

[36] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016.

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

[38] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[40] Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pages 1439–1456, 2020.

[41] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[42] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[43] Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

[44] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[45] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[46] Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li F Fei-Fei. Regression planning networks. In *Advances in Neural Information Processing Systems*, pages 1317–1327, 2019.

[47] Roman V Yampolskiy. Taxonomy of pathways to dangerous artificial intelligence. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[48] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[49] Yufei Ye, Dhiraj Gandhi, Abhinav Gupta, and Shubham Tulsiani. Object-centric forward modeling for model predictive control. In *Conference on Robot Learning*, pages 100–109, 2020.

[50] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

[51] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 483–492, 2017.

[52] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *International conference on robotics and automation*, pages 3357–3364, 2017.

A Additional Background

A.1 Contrastive Learning

Contrastive learning refers to a class of methods that attempt to minimize the distance of data points that share some attribute or condition. Given some "anchor" datapoint, $\boldsymbol{x}$, one typically finds some "positive" datapoint $\boldsymbol{x}^+$ that meets a condition C and forms the pair $(\boldsymbol{x}, \boldsymbol{x}^+)$. "Negatives" $\{\boldsymbol{x}_k^-\}$ are then data points that do not meet condition C for $\boldsymbol{x}$. In order to learn an embedding function $\boldsymbol{z} = f_{\texttt{enc}}(\boldsymbol{x})$, a common objective used is of the form [1]

$$\mathcal{L}_{\texttt{contrastive}} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+,\boldsymbol{x}_1^-,\ldots,\boldsymbol{x}_N^-}\left[-\log\left(\frac{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+\right)}{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+\right)+\sum_k\exp\left(\boldsymbol{z}^\top\boldsymbol{z}_k^-\right)}\right)\right] \tag{10}$$

It is common to chose negatives randomly from one's available data. This objective then encourages the learner to learn f such that the inner product of $z^\top z^+$ is higher on average than for $z^\top z^-$.

In supervised learning, the condition C can be whether two data points share a label. In unsupervised learning, other ground-truth information may be used. This is often referred to as "self-supervised" learning. For example, Oord et al. [29] define the condition C as whether x^+ occured within K time-steps of x. When ground-truth information is not available, a heuristic is typically used. This is what we do in our work.

A.1.1 A more numerically stable version

To give insight into some of the numerical stability issues that arise with the contrastive learning objective, we can introduce temperature τ and re-write it as follows [36]

$$\begin{aligned}
\mathcal{L}_{\texttt{contrastive}} &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+,\boldsymbol{x}_1^-,\ldots,\boldsymbol{x}_N^-}\left[-\log\left(\frac{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+/\tau\right)}{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+/\tau\right)+\sum_k\exp\left(\boldsymbol{z}^\top\boldsymbol{z}_k^-/\tau\right)}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+,\boldsymbol{x}_1^-,\ldots,\boldsymbol{x}_N^-}\left[\log\left(\frac{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+/\tau\right)+\sum_k\exp\left(\boldsymbol{z}^\top\boldsymbol{z}_k^-/\tau\right)}{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+/\tau\right)}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+,\boldsymbol{x}_1^-,\ldots,\boldsymbol{x}_N^-}\left[\log\left(1+\sum_k\frac{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}_k^-/\tau\right)}{\exp\left(\boldsymbol{z}^\top\boldsymbol{z}^+/\tau\right)}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+,\boldsymbol{x}_1^-,\ldots,\boldsymbol{x}_N^-}\left[\log\left(1+\sum_k\exp\left(\frac{\boldsymbol{z}^\top\boldsymbol{z}_k^- - \boldsymbol{z}^\top\boldsymbol{z}^+}{\tau}\right)\right)\right].
\end{aligned} \tag{11}$$

This is lower-bounded by 0 when the inner term $\exp(\cdot) \to 0$, which occurs when $\boldsymbol{z}^\top\boldsymbol{z}^+ = -\boldsymbol{z}^\top\boldsymbol{z}_k^- = \infty$. In practice, τ helps ensure that the norm of the resultant encodings is not too large. Even when $||\boldsymbol{z}||$ is not large, the exponential term can still grow to ∞, causing numerical instability. In practice, we constrain it by upper-bounding the term inside the exponential by m (see table 3 for exact numbers):

$$\mathcal{L}_{\texttt{contrastive}} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}^+,\boldsymbol{x}_1^-,\ldots,\boldsymbol{x}_N^-}\left[\log\left(1+\sum_k\exp\left(\max\left(\frac{\boldsymbol{z}^\top\boldsymbol{z}_k^- - \boldsymbol{z}^\top\boldsymbol{z}^+}{\tau}, m\right)\right)\right)\right]. \tag{12}$$

This leads to the final contastive learning loss we used in practice in place of equation 5:

$$\mathcal{L}_{\texttt{model}} = \mathbb{E}_{s_t,a_t,s_{t+1}}\left[-\sum_{i\in v_{t+1}}\log p(\rho_{t+1,i}^g|s_t^o,a_t).\right] \tag{13}$$

$$p(\rho_{t+1,i}^g|s_t^o,a_t) = \left(1+\sum_k\exp\left(\max\left(\frac{F(s_t^o,\rho_{t,i}^g,a_t)^\top\boldsymbol{z}_{k,-}^o - F(s_t^o,\rho_{t,i}^g,a_t)^\top\boldsymbol{z}_+^{o,i}}{\tau}, m\right)\right)\right) \tag{14}$$

A.2 Deep Relational RL

We differ from Zambaldi et al. [50] as they obtained object representations as convolutional feature vectors on full scenes and chose objects by selecting (x, y) coordinates, whereas we assume access to

object images, directly encode them, and select between them. While their approach is more general, it would undoubtedly lead to far slower learning as the correspondence between feature vectors and objects of varying sizes would need to be learned. We note that their experiments on Starcraft [41] required $500+$ million frames of experience, whereas we seek learning in 500k frames and thus leave this extension for future work.

B Extended Related Work

Object-Centric Reinforcement Learning.

Object-Centric Forward Models.

C Agent Details

C.1 Training Algorithm

Algorithm 1 Training Algorithm

while observed $\leq K$ samples **do**
  Act for T timesteps and observe trajectories $\{\tau\}$
  Add $\{\tau\}$ to replay buffer $\mathcal{B}$
  for m random batches from $\mathcal{B}$ **do**
    Compute loss $\mathcal{L} = \mathcal{L}^{\texttt{DQN}}$ using equation 1
    if $a = (b, o_c) \in \mathcal{A}_I$ **then**
      Compute loss $\mathcal{L}_{aux}$ for corresponding auxilliary task
      $\mathcal{L} \leftarrow \mathcal{L} + \beta^{\texttt{aux}}\mathcal{L}_{aux}$
    Perform gradient descent on Q-network parameters θ
    $\theta \leftarrow \theta - \eta_1 \nabla \mathcal{L}$
    Perform soft-update on target Q-network parameters $\theta_{\texttt{old}}$
    $\theta_{\texttt{old}} \leftarrow (1 - \eta_2)\theta_{\texttt{old}} + \eta_2 \theta$

We present the training algorithm as algorithm 1. We ran 10 parallel agents for $K = .5$ million joint time-steps. Between each set of batch updates, the 10 parallel agents collected $T = 5000$ samples of experiences. The environment automatically reset when episodes terminated. All agents relied on the Relational Object-DQN base architecture. Following Zhu et al. [51], they were trained with ϵ-greedy exploration, annealing ϵ from 1 to .1 over .5 million environment step. Agents were evaluated with an ϵ-greedy policy every 25000 steps, where ϵ was set to .1. We used the AdamW optimizer [23] with $w^{\texttt{AdamW}} = 1e-2, \beta_1^{\texttt{AdamW}} = 0.9, \beta_2^{\texttt{AdamW}} = 0.999, \alpha^{\texttt{AdamW}} = 0.001$. We used "soft target updates" [21] for $\theta_{\texttt{old}}$ with a smoothing coefficient η_2. You can find more details on hyperparameters (such as batchsize) in table 3.

You can find videos of ROMA and the next best-performing baseline performing our more challenging tasks in this anonymous youtube channel.

C.2 Architectures and objective functions

We present the details of the architecture used for all models in table 2. All models shared the Relational Object-DQN as their base. We built the Relational Object-DQN using the rlkit open-source reinforcement-learning library.

Relational Object DQN. This is our base architecture. Aside from the details in the main text, we note that $f^{\kappa}_{\texttt{enc}}$ is the concatenation of one function which encodes image information and another that encodes location information: $f^{\kappa}_{\texttt{enc}}(s^{\kappa}) = f^{\kappa}_{\texttt{enc}}(s^{\texttt{ego}}, s^{\texttt{loc}}) = [f^{\texttt{ego}}_{\texttt{enc}}(s^{\texttt{ego}}), f^{\texttt{loc}}_{\texttt{enc}}(s^{\texttt{loc}})]$.

Relational Object DQN + Scene Model: each agent predicts the encoding of the next scene. This is a simplified variant of Contrastive Predictive Coding (CPC) [29] that we augment to leverage object-interaction encodings $\boldsymbol{z}^a_t$ in its forward prediction. For a scene encoding $\boldsymbol{z}^{\texttt{ego}}_t = f^{\texttt{ego}}_{\texttt{enc}}(s^{\texttt{ego}})$, the query is a *scene-centric* forward model $F(s^{\kappa}_t, a_t)$. The positive is the scene-encoding at the next

Networks	Parameters
	Relational Object-DQN
Activation fn. (AF)	Leaky ReLU (LR)
$f^{\texttt{ego}}_{\texttt{enc}}$	Conv(32-8-4)-AF-Conv(64-4-2)-AF-Conv(64-3-1)-AF-MLP(9216-512)-AF
$f^{o}_{\texttt{enc}}$	Conv(32-4-2)-AF-Conv(64-4-2)-AF-Conv(64-4-2)-AF-MLP(4096-512)-AF
$f^{\texttt{loc}}_{\texttt{enc}}$	MLP(6-256)-AF-MLP(256-256)-AF
$\widehat{Q}_{\texttt{int}}(o_i)$	MLP(1280-256)-AF-MLP(256-256)-AF-MLP(256-8)
$\widehat{Q}_{\texttt{nav}}$	MLP(768-256)-AF-MLP(256-256)-AF-MLP(256-8)
$\mathcal{R}(\boldsymbol{z}^{o,i}, \boldsymbol{Z}^{o}) : W^k_1, W^q_1$	MLP(512-64), MLP(512-64)
$\mathcal{R}(\boldsymbol{z}^{\kappa}, \boldsymbol{Z}^{o}) : W^k_2, W^q_2$	MLP(768-64), MLP(512-64)
	Object-centric model
$f_{\texttt{model}}$	MLP(1088-256)-AF-MLP(256-512)
$\boldsymbol{z}^a : W^o, W^b$	MLP(512-64), MLP(8-64)
	Scene-centric model
$f^{\kappa}_{\texttt{model}}$	MLP(832-512)
$\boldsymbol{z}^a : W^o, W^b$	MLP(512-64), MLP(8-64)
	VAE
$f_{\texttt{recon}}$	MLP(4096-512)-AF-Conv(64-4-2)-AF-Conv(64-4-2)-AF-Conv(32-3-2)

Table 2: Architectures used across all experiments.

time-step $\boldsymbol{z}^{\texttt{ego}}_{t+1}$. We then choose k negatives $\{\boldsymbol{z}^{\texttt{ego}}_{k,-}\}$. The objective function is:

$$\mathcal{L}_{\texttt{scene}} = \mathbb{E}_{s_t, a_t, s_{t+1}} \left[-\log p(s^{\texttt{ego}}_{t+1} | s^{\kappa}_t, a_t). \right] \tag{15}$$

$$p(s^{\texttt{ego}}_{t+1} | s^{\kappa}_t, a_t) = \frac{\exp(F(s^{\kappa}_t, a_t)^\top \boldsymbol{z}^{\texttt{ego}}_{t+1})}{\exp(F(s^{\kappa}_t, a_t)^\top \boldsymbol{z}^{\texttt{ego}}_{t+1}) + \sum_k \exp(F(s^{\kappa}_t, a_t)^\top \boldsymbol{z}^{\texttt{ego}}_{k,-})} \tag{16}$$

$$F(s^{\kappa}_t, a_t) = f^{\kappa}_{\texttt{model}}([\boldsymbol{z}^{\kappa}_t, \boldsymbol{z}^{a}_t]) \tag{17}$$

Relational Object DQN + Object β-VAE: each agent predicts the latent factors that have generated each individual object-image-patch. This requires an additional reconstruction network for the object-encoder, $f_{\texttt{recon}}(\boldsymbol{z}^{o,i}_t)$, which produces an object-image-patch back from an encoding. The objective function is:

$$\mathcal{L}_{\texttt{vae}} = \mathbb{E}_{s_t} \left[\sum_{i \in v_t} \left(||f_{\texttt{recon}}(\boldsymbol{z}^{o,i}_t) - o_{t,i}||^2_2 - \beta^{\texttt{kl}} \, \mathrm{KL}(p(\boldsymbol{z}^{o,i}_t | o_{t,i}) || p(\boldsymbol{z}^{o,i}_t)) \right) \right] \tag{18}$$

where KL is the Kullback-Leibler Divergence and $p(\boldsymbol{z}^{o,i}_t)$ is an isotropic, unit gaussian. We also model $p(\boldsymbol{z}^{o,i}_t | o_{t,i})$ as a gaussian. We augment the Relational Object-DQN so that $\boldsymbol{z}^{o,i}_t$ is the mean of the gaussian and so that a standard deviation is also computed. Please see Higgins et al. [13] for more.

Relational Object DQN + Scene β-VAE: each agent predicts the latent factors that have generated each scene $s^{\texttt{ego}}$. This requires an additional reconstruction network for the scene, $f_{\texttt{recon}}(\boldsymbol{z}^{\texttt{ego}}_t)$. The objective function is:

$$\mathcal{L}_{\texttt{vae}} = \mathbb{E}_{s_t} \left[\left(||f_{\texttt{recon}}(\boldsymbol{z}^{\texttt{ego}}_t) - s^{\texttt{ego}}_t||^2_2 - \beta^{\texttt{kl}} \, \mathrm{KL}(p(\boldsymbol{z}^{\texttt{ego}}_t | s^{\texttt{ego}}_t) || p(\boldsymbol{z}^{\texttt{ego}}_t)) \right) \right] \tag{19}$$

All other settings except for hyperparameters were the same as the object β-VAE.

Relational Object DQN + OCN: the agent tries to learn encodings of object-image-patches such that patches across time-steps corresponding to the same object are grouped nearby in latent space, and patches corresponding to different objects are pushed apart. This also relied on contrastive learning, except that it uses it on image-pairs across time-steps. Following Pirk et al. [32], the anchor

is defined as the object-encoding $\boldsymbol{z}_t^{o,i} = f(o_{t,i})$, which we will refer to as f. The positive is defined as the object-image-patch encoding at the next time-step with lowest $L2$ distance in latent space, $f^+ = \arg\min_{z_{t+1}^{o,j}} ||\boldsymbol{z}_t^{o,i} - \boldsymbol{z}_{t+1}^{o,j}||^2$. We then set negatives $\{f_k^-\}$ as the object-image-patches that did not correspond to the match. We note that augmenting Pirk et al. [32] so that their objective function had temperature τ was required for good performance. For a unified perspective with our own objective function, we write their n-tuplet-loss with a softmax (see Sohn [36] for more details on their equivalence). The objective function is:

$$\mathcal{L}_{\texttt{ocn}} = \mathbb{E}_{s_t, s_{t+1}} \left[-\log \left(\frac{\exp\left(f^\top f^+ / \tau\right)}{\exp\left(f^\top f^+ / \tau\right) + \sum_k \exp\left(f^\top f_k^- / \tau\right)} \right) \right] \tag{20}$$

Relational Object DQN + Ground Truth Object Info: the agent doesn't have an auxilliary task and doesn't encode object-images. Instead it encodes object-information. For each object, we replace object-image-patches with the following information available in Thor:

1. the object's category (i.e. its type index)
2. the object's unique instance index (i.e. its token index)
3. the object this object was in (e.g. if this object is a cup in the sink, this would correspond to the sink index)
4. distance to object (in meters)
5. whether object is visible (boolean)
6. whether object is toggled (boolean)
7. whether object is broken (boolean)
8. whether object is filledWithLiquid (boolean)
9. whether object is dirty (boolean)
10. whether object is cooked (boolean)
11. whether object is sliced (boolean)
12. whether object is open (boolean)
13. whether object is pickedUp (boolean)
14. object temperature (cold, room-temperature, hot)

C.3 Hyperparameter Search

Relational Object DQN. All models are based on the same Relational Object DQN agent and thus use the same hyperparameters. We searched over these parameters using "Relational Object DQN + Ground-Truth Object-Information". We searched over tuples of the parameters in the "DQN" portion of table 3. In addition to searching over those parameters, we searched over "depths" and hidden layer size of the multi-layer perceptrons $f_{\texttt{enc}}^{\texttt{loc}}$, $\widehat{Q}_{\texttt{int}}(o_i)$, and $\widehat{Q}_{\texttt{nav}}$. For depths, we searched uniformly over $[0, 1, 2]$ and for hidden later sizes we searched uniformly over $[128, 256, 512]$. We searched over 12 tuples on the "Fill Cup with Water" task and 20 tuples on the "Place Apple on Plate & Both on Table" task. We found that task-performance was sensitive to hyperparameters and choose hyperparameters that achieved a $90\%+$ success rate on both tasks. We fixed these settings and searched over the remaining values for each auxilliary task.

Object-centric forward model. In addition to experimenting with using a dot-product as our kernel $\boldsymbol{u}^\top \boldsymbol{w}$, due to the inherent numerical instability of equation 10, we also experimented with using cosine similarity and applying an L2 norm on the vectors. While the latter has been recommended by prior work [36], we found that applying temperature τ as recent work has [12, 42] was superior. We experimented with the number of negative examples used for the contrastive loss and found no change in performance. We performed a search over 4 tuples from the values in table 3. We chose the loss-coefficient as the the coefficient which put the object-centric model loss at the same order of magnitude as the DQN loss.

Scene-centric forward model, Object β-VAE, OCN. For each auxilliary task, we performed a search over 6 tuples from the values in table 3. For each loss, we chose loss coefficients that scaled the loss so they were between an order of magnitude above and below the DQN loss.

Hyperparameter	Final Value	Values Considered
Max gradient norm	0.076	log-uniform$(10^{-4}, 10^{-1})$
DQN		
Learning rate η_1	1.8×10^{-5}	log-uniform$(10^{-6}, 10^{-2})$
Target Smoothing Coeffecient η_2	0.00067	log-uniform$(10^{-6}, 10^{-3})$
Discount γ	0.99	
Training ϵ annealing	$[1, .1]$	
Evaluation ϵ	.1	
Replay Buffer Size	200000	
Batchsize	50	
Object-centric model		
upper-bound m	85	-
Number of Negative Examples	20	-
temperature τ	8.75×10^{-5}	log-uniform$(10^{-6}, 10^{-3})$
Loss Coeffecient $\beta^{\texttt{model}}$	10^{-3}	-
Scene-centric model		
upper-bound m	85	-
Number of Negative Examples	20	-
temperature τ	0.0034	log-uniform$(10^{-6}, 10^{-3})$
Loss Coeffecient $\beta^{\texttt{model}}$	0.0013	log-uniform$(10^{-4}, 10^{-2})$
VAE		
KL Coeffecient $\beta^{\texttt{kl}}$	29	log-uniform$(10^{-1}, 10^{2})$
Loss Coeffecient $\beta^{\texttt{vae}}$	0.0026	log-uniform$(10^{-4}, 1)$
OCN		
temperature τ	5×10^{-5}	log-uniform$(10^{-6}, 10^{-3})$
Loss Coeffecient $\beta^{\texttt{ocn}}$	0.0047	log-uniform$(10^{-4}, 10^{-2})$

Table 3: Hyperparameters shared across all experiments.

D Thor Implementation Details

D.1 Thor Settings

Environment. While AI2Thor has multiple maps to choose from, we chose "Floorplan 24". To reduce the action-space, we restricted the number of object-types an agent could interact with so that there were 10 distractor types beyond task relevant object-types. We defined task-relevant object-types as objects needed to complete the task or objects they were on/inside. For example, in "Place Apple on Plate & Both on Table", since the plate is on a counter, counters are task object-types. We provide a list of the object-types present in each task with the task descriptions below.

Observation. Each agent observes an 84×84 grayscale image of the environment, downsampled from a 300×300 RGB image. They can detect up to 20 obects per time-step within its line of sight, if they exceed 50 pixels in area, regardless of distance. Each object in the original 300×300 scene image is cropped and resized to a 32×32 grayscale image[3]. Each agent observes its (x, y, z) location, and its pitch, yaw, and roll body rotation $(\varphi_1, \varphi_2, \varphi_3)$ in a global coordinate frame.

Object-Interactions. Each agent has 8 base object-interactions: $\mathcal{A}_I = \{$*Pickup, Put, Open, Close, Toggle on, Toggle off, Slice, Fill*$\}$. When an interaction is chosen, one of the ≤ 20 detected obects must be chosen as an argument. No masking is done on object-interactions. Thor's physics engine

[3]For "Slice" tasks and "Make Tomato & Lettuce Salad", we used an object image size of 64×64 to facilitate recognition of smaller objects. We decreased the replay buffer to have 120000 samples.

resolved whether interactions succeeded. We selected the maximum interaction distance as $1.5m$, as some tasks (such as cook potato on stove required it).

Navigation Actions. Each agent has 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 agents head could rotate its head up or down between angles $\{0°, \pm 30°, \pm 60°\}$ in increments of $30°$, where $0°$ represents looking straight ahead. With $\{$*Rotate Left, Rotate Right*$\}$, each agent could also rotate it's body by $\{\pm 90^o\}$.

Episodes. The episode terminates either after 500 steps or when a task is complete. The agent's spawning location is randomly sampled from the 81 grid positions facing North with a body angle $(0°, 0°, 0°)$. Each agent recieves reward 1 if a task is completed successfully and a time-step penalty of -0.04.

Setting	Values
Observation Size	300×300
Downsampled Observation Size	84×84
Object Image Size	32×32
Min Bounding Box Proportion	$\frac{50}{300 \times 300}$
Max Interaction Distance	1.5m

Table 4: Settings used in Thor across experiments.

D.2 Task Details

For each task, we describe which challenges were present, what object types were interactable, and the total Key Semantic Actions available. We chose objects that were evenly spaced around the environment. As a reminder, the challenges were:

Challenge A: the need for view-invariance (e.g. recognizing a knife across angles),

Challenge B: the need to reason over ≥ 3 objects,

Challenge C: the need to recognize and use *combined* objects (e.g. filling a cup with water in the sink or toasting bread in a toaster).

Make Coffee.
Challenges:
A: recognizing the coffee machine across angles.
C: recognizing the mug in the coffee machine.
Interactable Object Types: 13
- DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Knife: 1, Mug: 1

Key Semantic Actions:
1. Go to Mug
2. Pickup Mug
3. Place Mug in Coffee Machine
4. Turn on Coffee Machine

Clean Bowl in Sink.
Challenges:
A: recognizing the sink across angles.
C: recognizing the bowl in the sink.
Interactable Object Types: 19
1. CounterTop: 3, Faucet: 2, Sink: 1, DiningTable: 1, Microwave: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Egg: 1, Cup: 1, SinkBasin: 1, Pot: 1, Tomato: 1, DishSponge: 1, Knife: 1, Bowl: 1

Key Semantic Actions:
1. Go to Bowl
2. Pickup Bowl
3. Place Bowl in Sink

4. Turn on Faucet

Tomato in Garbage.

Challenges:

A: trecognizing the garbage across occlusions.

Interactable Object Types: 13

- DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Garbage-Can: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Knife: 1

Key Semantic Actions:

1. Go to Tomato
2. Pickup Tomato
3. Go to Garbage
4. Put Tomato in Garbage

Lettuce in Fridge.

Challenges:

A: recognizing the fridge across angles.

Interactable Object Types: 13

- DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Lettuce: 1, Knife: 1

Key Semantic Actions:

1. Go to Lettuce
2. Pickup Lettuce
3. Go to Fridge
4. Put Lettuce in Fridge

Slice Bread.

Challenges:

A: recognizing the knife across angles.

Interactable Object Types: 15

- CounterTop: 3, DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Knife: 1

Key Semantic Actions:

1. Go to Knife
2. Pickup Knife
3. Go to Bread
4. Slice Bread

Slice Lettuce and Tomato. (order doesn't matter)

Challenges:

A: recognizing the knife across angles.

B: ecognizing and differentiate 3 task objects: the knife, lettuce, and tomato. As each object is cut, the agent needs to choose from more objects as it can select from the object-slices.

Interactable Object Types: 17

- CounterTop: 3, DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Spatula: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Lettuce: 1, Knife: 1

Key Semantic Actions:

1. Go to Knife
2. Pickup Knife
3. Go to Table
4. Slice Lettuce
5. Slice Tomato

Slice Lettuce and Apple, and Potato. (order doesn't matter)

Challenges:

A: recognizing the knife across angles.

B: recognizing and differentiate 4 task objects: the knife, lettuce, and apple, and potato. As each object is cut, the agent needs to choose from more objects as it can select from the object-slices.

Interactable Object Types: 18

- CounterTop: 3, DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Potato: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Lettuce: 1, Apple: 1, Knife: 1

Key Semantic Actions:

1. Go to Knife
2. Pickup Knife
3. Go to Table
4. Slice Lettuce
5. Slice Apple
6. Slice Potato

Cook Potato on Stove.
Challenges:
A: recognizing the stove across angles.
B: needs to differentiate 3 objects: the stove knob, pot, and potato.
C: recognizing the potato in the pot.
Interactable Object Types: 21
1. StoveBurner: 4, StoveKnob: 4, DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Potato: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Knife: 1

Key Semantic Actions:
1. Go to Potato
2. Pickup Potato
3. Go to Stove
4. Put Potato in Pot
5. Turn on Stove Knob

Fill Cup with Water.
Challenges:
A: recognizing the cup across angles and backgrounds.
B: recognizing the cup in the sink.
C: the need to recognize and use *combined* objects (e.g. filling a cup with water in the sink or toasting bread in a toaster).
Interactable Object Types: 18
- CounterTop: 3, Faucet: 2, Sink: 1, DiningTable: 1, Microwave: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Egg: 1, Cup: 1, SinkBasin: 1, Pot: 1, Pan: 1, Tomato: 1, Knife: 1

Key Semantic Actions:
1. Go to Cup
2. Pickup Cup
3. Go to Sink
4. Put Cup in Sink
5. Fill Cup

Toast Bread Slice.
Challenges:
A: recognizing the toaster across angles.
C: recognizing the bread slice in the toaster.
Interactable Object Types: 21
- BreadSliced: 5, CounterTop: 3, Bread: 2, DiningTable: 1, Microwave: 1, CoffeeMachine: 1, Fridge: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Knife: 1, Toaster: 1

Key Semantic Actions:
1. Go to Bread Slice
2. Pickup Bread Slice
3. Go to Toaster
4. Put Breadslice in Toaster
5. Turn on Toaster

Place Apple on Plate & Both on table.
Challenges:
A: recognizing the plate across occlusions.
B: needs to differentiate 3 objects: the apple, plate, and table.
C: recognizing the apple on the plate.
Interactable Object Types: 16
- CounterTop: 3, DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Bread: 1, Fridge: 1, Spatula: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Apple: 1, Knife: 1

Key Semantic Actions:
1. Go to Apple

2. Pickup Apple
3. Put Apple on Plate
4. Pickup Plate
5. Go to Table
6. Put Plate on Table

Make Tomato & Lettuce Salad.
Challenges:

A: recognizing the plate across occlusions.
B: needs to differentiate 3 objects: the tomato slice, lettuce slice, and plate.
C: recognizing the tomato slice or lettuce slice on the plate.

Interactable Object Types: 32

- TomatoSliced: 7, LettuceSliced: 7, CounterTop: 3, Bread: 2, DiningTable: 1, Microwave: 1, Plate: 1, CoffeeMachine: 1, Fridge: 1, Spatula: 1, Egg: 1, Cup: 1, Pot: 1, Pan: 1, Tomato: 1, Lettuce: 1, Knife: 1

Key Semantic Actions:

1. Go to table
2. Pickup tomato or lettuce slice
3. Put slice on Plate
4. Pickup other slice
5. Put slice on Plate

D.3 Interaction Dataset

In order to measure and analyze the quality of the object representations learned via each auxilliary task, we created a dataset with programmatically generated object-interactions and with random object-interactions. This enabled us to have a diverse range of object-interactions and ensured the dataset had many object-states present.

Programatically Generated object-interactions. This dataset contains programmatically generated sequences of interactions for various tasks. The tasks currently supported by the dataset include: *pickup X, turnon X, open X, fill X with Y, place X in Y, slice X with Y, Cook X in Y on Z.* For each abstract task type, we first enumerate all possible manifestations based on the action and object properties. For example, manifestations of open X include all objects that are openable. We exhaustively test each manifestation and identify the ones that are possible under the physics of the environment. We explicitly build the action sequence required to complete each task. Because we only want to collect object-interactions, we use the high level "TeleportFull" command for navigation to task objects. The TeleportFull command allows each agent to conveniently navigate to desired task objects at a particular location and viewing angle. For example, the sequence for place X in Y is: TeleportFull to X, Pickup X, TeleportFull to Y, and Put X in Y. An agent will execute each action until termination. We collect both successful and unsuccessful task sequences. There is a total of 156 unique tasks in the dataset and 1196 individual task sequences amounting to 2353 $(\text{state}, \text{action}, \text{next state})$ tuples.

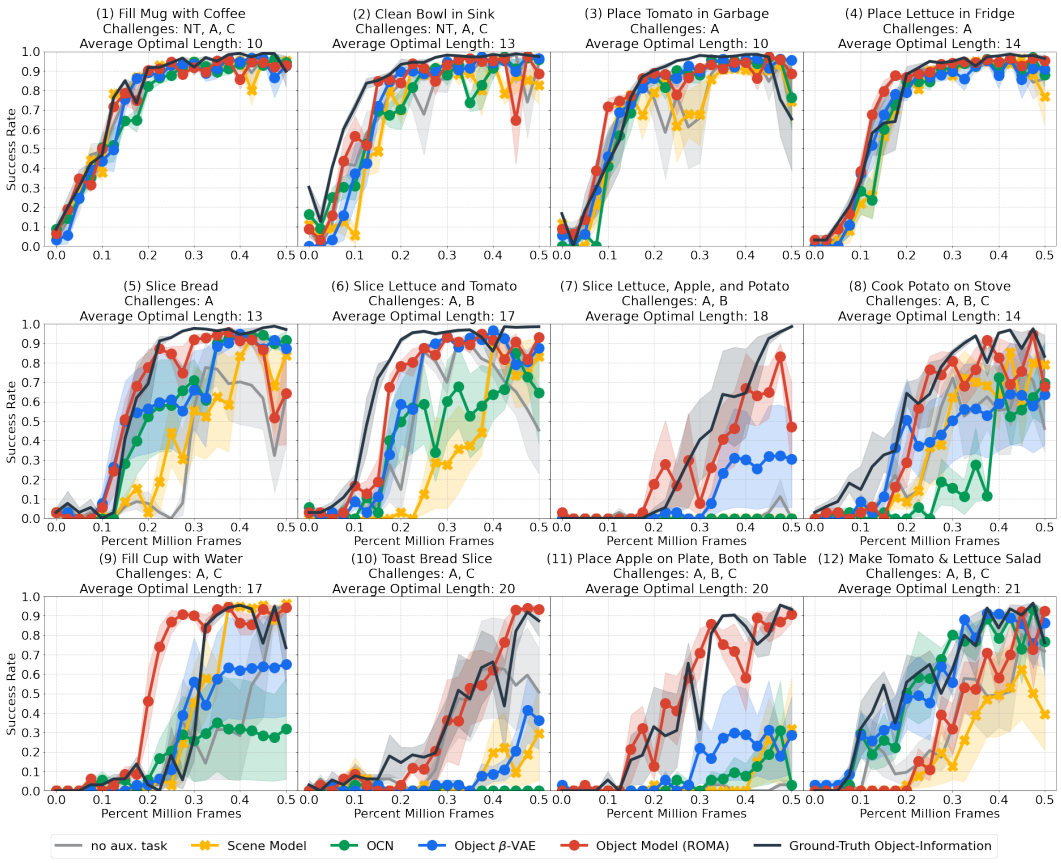
Random object-interactions. The random interaction dataset consists of $(\text{state}, \text{action}, \text{next state})$ tuples of random interactions with the environment. An agent equipped with a random action policy interacts with the environment for episodes of 500 steps until it collects a total of 4000 interaction samples.

training. We divided the data into an 80/20 training/evaluation split and trained for 750 epochs. We reported the test data results.

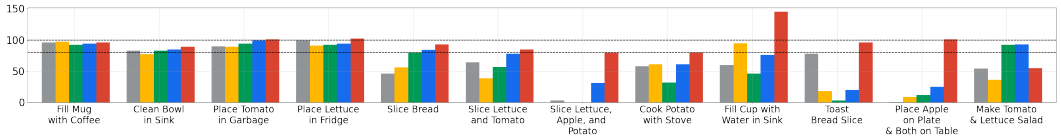
E Additional Results

E.1 Results on easier tasks

In addition to the main results presented in 5.3, we also present results on simpler tasks in the first row of Figure 5. For these tasks we add the challenge label **NT**, which represents No-Transportation. This means that all task-objects are at the same location so the agent doesn't need to transport them to each other. For example, the Mug in "Fill Mug with Coffee" is at the same location as the coffee machine. We find that all models were able to perform reasonably well for these simpler tasks. The benefits of unsupervised object-representation learning seem to manifest when the agent needs to interact with more objects or needs to model *combinations* of objects.



(a) Success rate over learning for competing auxilliary tasks. We see that all models perform well, with ROMA slighly outperforming other models.



(b) Sample efficiecy of each auxilliary task w.r.t. the agent with ground-truth object-information.

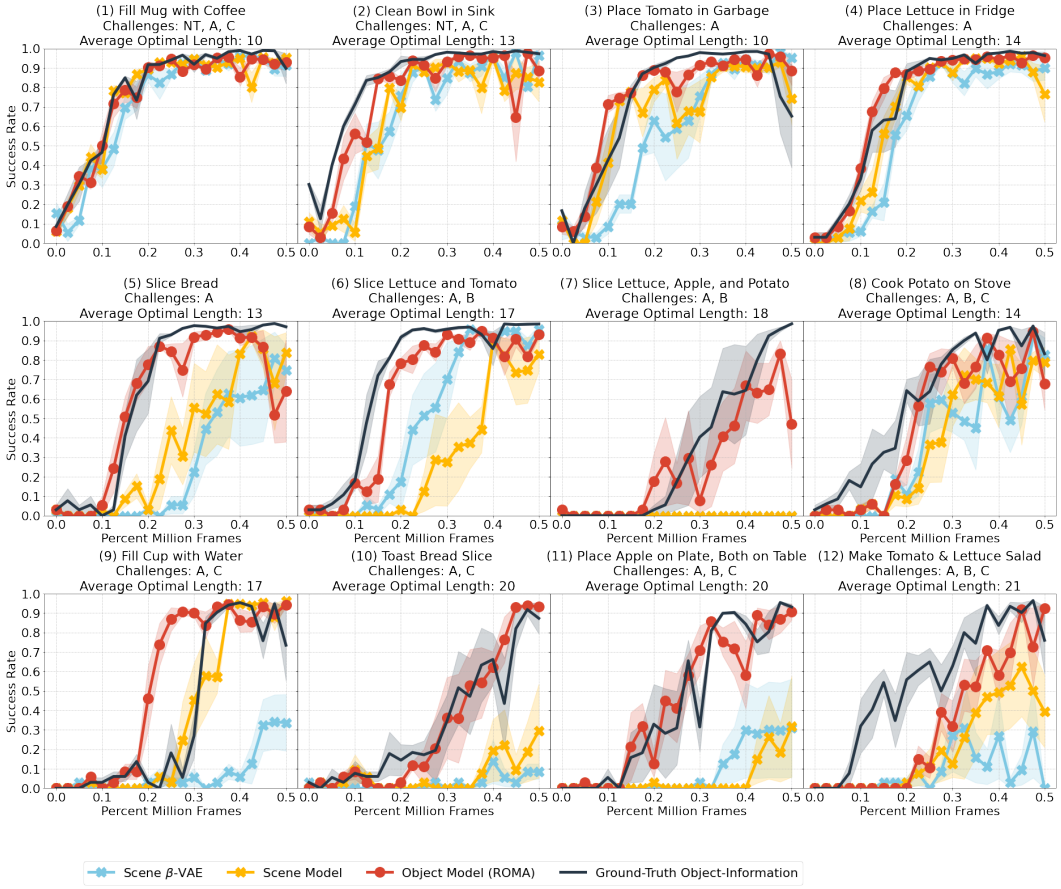
Figure 5: We see that using an object-model for object-representation learning has the strongest benefits when (i) more objects and (ii) combinations of objects need to be modelled.

E.2 Additional scene-centric representation learning results

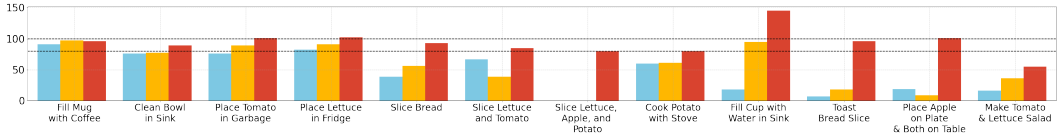
In section 5.3, we compared methods for learning object-representations to learning scene-representations with a scene-centric forward model. Here, we present further results comparing against a **Scene β-VAE** baseline that uses a β-Variational Autoencoder to learn scene representations. As most prior work hasn't explicitly computed action-value estimates for object-image-patches, this falls more in line with a conventional representation learning solution for improved sample-efficiecy in reinforcement learning. We found this to be the worst performing auxilliary task. On $8/12$ tasks, we found that learning scene-representations with a scene-model outperformed learning scene-representations with the β-Variational Autoencoder. See figure 6 for more.

E.3 Trajectory analysis

When analyzing the penultimate state on the "Make Salad" task where the Object β-VAE outperformed our object-model (see Figure 7), we find that the Object β-VAE is able to find a perspective of the partially complete salad that doesn't require it learn to represent the full combination of objects; whereas our object-model doesn't. We hypothesize that this is due to it needs to unify representations across time, and thus leads ROMA to learn more slowly for this task.



(a) Success rate over learning for competing auxilliary tasks. We see that the Scene β-VAE



(b) Sample efficiecy of each auxilliary task w.r.t. the agent with ground-truth object-information.

Figure 6: We see that applying a beta-vae to learning scene-representations has marginal benefits. In general, learning a model does better.

E.4 View-invariance of learning representations

In Figure 8, we visualize the representations learned by our object-model and the β-VAE of 3 objects across state-states and we find that (a) our object-model better clusters differents views and states from the same object and (b) the object-model more often places pairs of object-states from the same-view nearby in latent space.

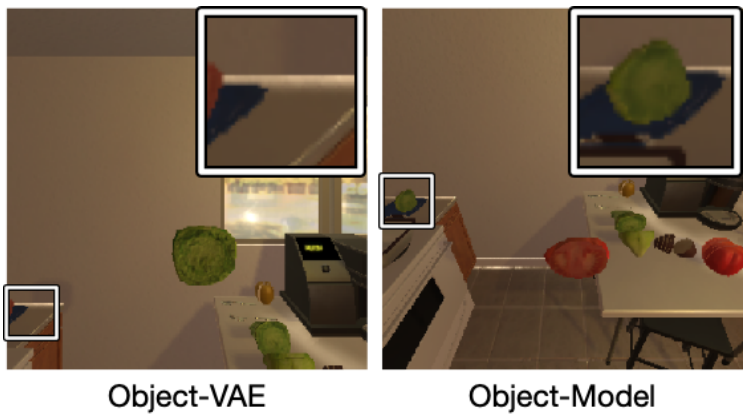


Figure 7: In both panels, we see common penultimate states of a relational agent trained with an object-vae and an object-model on the "Make Lettuce and Tomato Salad" task. On the left, we can see that the agent finds an object-view of the tomato in the plate that doesn't capture the full image, potentially enabling easier representation learning of the new *combined* object. On the right, we see the agent with an object-model finds a solution trajectory where it percieves a full image of the lettuce in the plate. This potentially requires the agent learn to represent a more complex image in order to represent the *combination* of the two objects. We suspect that this made representation learning more challenging for our object-model and led to slower learning on this task. Our results in table 1 indicate this might be true, as the object-model was able to better capture object-categories and the presence of containment object-relationships.

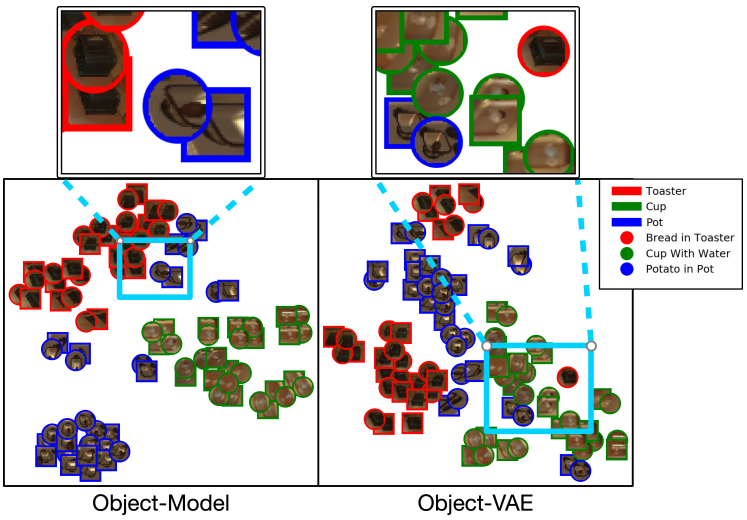


Figure 8: t-sne clustering visualizations of images by our Object-Model vs the Object VAE. We find that our object-model better creates distinct groups that cluster objects across both view and state. Additionally, we find that our object-model more often places images belonging to the same state-transition together.