A. Experiment Details

A.1. Experiment Setup

We used the Adam optimizer from (Kingma and Ba) for learning our Successor Representation (SR) model with a learning rate of 1e-4 and a mini-batch size of 32. For the reinforcement learning experiments, we use the discounted factor $\gamma = 0.99$ and a replay buffer size of 100,000. The exploration term ϵ is annealed from 1.0 to 0.1 during the training process. We run an ϵ-greedy policy ($\epsilon = 0.1$) during evaluation. We use soft target updates ($\tau = 0.1$) after every episode. For the easy and medium tasks, we assign $+10.0$ immediate reward for task completion, -5.0 for invalid actions, and -1.0 for other actions (to encourage a shorter plan). For the hard task, we train our SR model to imitate a plan that searches all the *receptacles* for an object in a fixed order of visitation based on the spatial locations of the *receptacles*. We assign $+1.0$ immediate reward for task completion, and an episode terminates as failure if the agent does not follow the order of visitation in the plan.

A.2. Network Inputs

The input to the SR model consists of three components: action (action type and argument), agent's observation (image), and agent's internal state. The action type is encoded by a 7-dimensional one-hot vector, indicating one of the seven action types (Navigate, Open, Close, Pick Up, Put, Look Up, and Look Down). The action argument is encoded by a one-hot vector that has the same dimension as the number of interactable objects plus one. The first dimension denotes null argument used for Look Up and Look Down actions, and the other dimensions correspond to the index of each object. RGB images from the agent's first-person camera are preprocessed to 84×84 grayscale images. We stack four history frames to make an $84 \times 84 \times 4$ tensor as the image input to the convolutional networks. The agent's internal state is expressed by the agent's inventory, rotation, and viewpoint. The inventory is a one-hot vector that represents the index of the held *item*, with an extra dimension for null. The rotation is a 4-dimensional one-hot vector that represents the rotation of the agent (90 degree turns). The viewpoint is a 3-dimensional one-hot vector that represents the tiling angle of the agent's camera ($-30°$, $0°$, and $30°$).

A.3. Network Architecture

Here we describe the network architecture of our proposed SR model. The convolutional image encoder θ_{cnn} takes an $84 \times 84 \times 4$ image as input. The three convolutional layers are 32 filters of size 8×8 with stride 4, 64 filters of size 4×4 with stride 2, 64 filters of size 3×3 with stride 1. Finally a fully-connected layer maps the outputs from the convolutional encoder into a 512-d feature. The actions encoder θ_{mlp} and internal state encoder θ_{int} are both 2-layer MLPs with 512 hidden units. A concatenated vector of action, internal state, and image encodings is fed into two 2-layer MLPs θ_r and θ_q with 512 hidden units to produce the 512-d state-action feature $\phi_{s,a}$ and the successor feature $\psi_{s,a}$. We take the dot product of the 512-d reward predictor vector $\mathbf{w}$ and state-action features (successor features) to compute the immediate rewards (Q values). All the hidden layers use ReLU non-linearities. The final dot product layers of the immediate reward and the Q value produce raw values without any non-linearity.

B. Algorithm Details

We describe the reinforcement learning procedure of the SR model in Algorithm 1. This training method follows closely with previous work on deep Q-learning [35] and deep SR model [24]. Similar to these two works, replay buffer and target network are used to stabilize training.

C. Action Space

The set of plausible actions in a scene is determined by the variety of objects in the scene. On average each scene has 53 objects (a subset of them are interactable) and the agent is able to perform 80 actions. Here we provide an example scene to illustrate the interactable objects and the action space.

Scene #9: 16 *items*, 23 *receptacles* (at 11 unique locations), and 15 *containers* (a subset of *receptacles*)

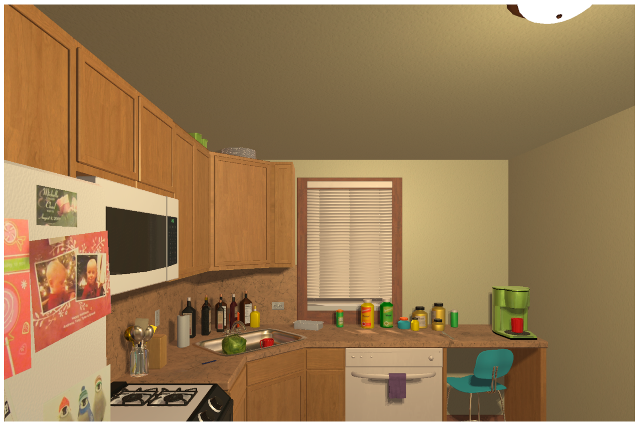

Figure 8. Screenshot of Scene #9

items: *apple*, *bowl*, *bread*, *butter knife*, *glass bottle*, *egg*, *fork*, *knife*, *lettuce*, *mug 1-3*, *plate*, *potato*, *spoon*, *tomato*

receptacles: *cabinet 1-13*, *coffee machine*, *fridge*, *garbage can*, *microwave*, *sink*, *stove burner 1-4*, *table top*

containers: *cabinet 1-13*, *fridge*, *microwave*

actions: 80 actions in total, including 11 Navigation actions, 15 Open actions, 15 Close actions, 14 Pick Up ac-

Algorithm 1 Reinforcement Learning for Successor Representation Model

```
1:  procedure RL-TRAINING
2:      Initialize replay buffer $\mathcal{D}$ to size $N$
3:      Initialize an SR network $\theta$ with random weights $\theta = [\theta_{int}, \theta_{cnn}, \theta_{mlp}, \theta_r, \theta_q, \mathbf{w}]$
4:      Make a clone of $\theta$ as the target network $\tilde{\theta}$
5:      for $i = 1$ : #episodes do:
6:          Initialize an environment with random configuration
7:          Reset exploration term $\epsilon = 1.0$
8:          while not terminal do
9:              Get agent's observation and internal state $s_t$ from the environment
10:             Compute $Q_{s_t,a} = f(s_t, a; \theta)$ for every action $a$ in action space
11:             With probability $\epsilon$ select a random action $a_t$; otherwise, select $a_t = \arg\max_a Q_{s_t,a}$
12:             Execute action $a_t$ to obtain the immediate reward $r_t$ and the next state $s_{t+1}$
13:             Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
14:             Sample a random mini-batch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$
15:             Compute $\tilde{r}_j$, $\phi_{s_j,a_j}$, and $\psi_{s_j,a_j}$ using $\theta$ for every transition $j$
16:             Compute gradients that minimize the mean squared error between $r_j$ and $\tilde{r}_j$
17:             Compute $\phi_{s_{j+1},a}$, $\psi_{s_{j+1},a}$, and $\tilde{Q}_{s_{j+1},a}$ using $\tilde{\theta}$ for every transition $j$ and every action $a$
18:             if $s_{j+1}$ is a terminal state then:
19:                 Compute gradients that minimize the mean squared error between $\psi_{s_j,a_j}$ and $\phi_{s_j,a_j}$
20:             else:
21:                 Compute gradients that minimize the mean squared error between $\psi_{s_j,a_j}$ and $\phi_{s_j,a_j} + \gamma\psi_{s_{j+1},a'}$
22:                 where $a' = \arg\max_a \tilde{Q}_{s_{j+1},a}$
23:             end if
24:             Perform a gradient descend step to update $\theta$
25:         end while
26:         Anneal exploration term $\epsilon$
27:         Soft-update target network $\tilde{\theta}$ using $\theta$
28:     end for
29: end procedure
```

tions, 23 Put actions, Look Up and Look Down.

We have fewer Navigation and Pick Up actions than the number of receptacles and items respectively, as we merge some adjacent receptacles to one location (navigation destination). We also merge picking up items from the same object category into one action. This reduces the size of the action space and speeds up learning. An important simplification that we made is to treat the Navigation actions as "teleports", which abstracts away from visual navigation of the agent. The actual visual navigation problem can be solved as an independent subroutine from previous work [54]. As discussed in Sec. 3.2, not all actions in the set can be issued given a certain circumstance based on affordance. We use the PDDL language to check if the preconditions of an action are satisfied before the action is sent to THOR for execution.

D. Tasks

We list all the tasks that we have evaluated in the experiments in Table 2. In summary, we evaluated tasks from three levels of difficulty, with 10 easy tasks, 8 medium tasks, and 7 hard tasks.

References

[1] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Table 2. List of Tasks from Three Levels of Difficulty

Scene	**Easy**	**Medium**	**Hard**
1	open / close *fridge*	put *lettuce, tomato* and *glass bottle* to the *sink*	find *bowl* and put in *sink*
2	open / close *cabinet*	put *apple, egg* and *glass bottle* to the *table top*	find *plate* and put in *cabinet*
3	open / close *microwave*	put *glass bottle, lettuce* and *apple* to the *table top*	find *lettuce* and put in *fridge*
4	open / close *cabinet*	put three *mugs* to the *fridge*	find *glass bottle* and put in *microwave*
5	open / close *fridge*	-	-
6	open / close *fridge*	-	-
7	open / close *cabinet*	put three *mugs* to the *table top*	-
8	open / close *fridge*	put *potato, tomato* and *apple* to the *sink*	find *lettuce* and put on *table top*
9	open / close *microwave*	put three *mugs* to the *table top*	find *glass bottle* and put in *fridge*
10	open / close *cabinet*	put *glass bottle, bread* and *lettuce* to *fridge*	find *bowl* and put in *sink*