Visual Semantic Planning using Imitation Learning

Yuke Zhu*[3] Daniel Gordon*[4] Eric Kolve[1] Dieter Fox[4]
Li Fei-Fei[3] Abhinav Gupta[1,2] Roozbeh Mottaghi[1] Ali Farhadi[1,4]
[1]Allen Institute for Artificial Intelligence [2]Carnegie Mellon University
[3]Stanford University [4]University of Washington

I. Introduction

A crucial capability of autonomous robots is their ability to **plan** a sequence of actions to achieve their goals in the visual world. In this work, we address the problem of **visual semantic planning**: the task of predicting a sequence of actions from visual observations. Rather than using traditional reinforcement learning approaches, we propose using an optimal planner and imitation learning to train our agent. Our experiments show near optimal results across a wide range of tasks in the challenging THOR [6] environment. The supplementary video can be seen at https://goo.gl/vXsbQP.

A. Visual Semantic Planning

Visual semantic planning (VSP) involves addressing several challenging problems. For example, in the simple task of `putting a bowl in a microwave`, a successful plan involves finding the bowl, navigating to it, grabbing it, finding and navigating to the microwave, opening the microwave, and finally putting the bowl in the microwave. In this paper, we address VSP as a policy learning problem. We focus on high-level actions and do not take into account the low-level details of motor control and motion planning. To address the large state space and delayed rewards, we use imitation learning with an optimal planner rather than reinforcement learning.

B. Related Work

Task planning: Task-level planning [2, 3] addresses the problem of finding a high-level plan for performing a task. These methods typically work with high-level formal languages and low-dimensional state spaces. In contrast, visual semantic planning is particularly challenging due to the high dimensionality and partial observability of the visual inputs.
Learning From Demonstration: Expert demonstrations, often performed by humans, offer a source of supervision in tasks which must usually be learned with copious random exploration. [4] presents an algorithm for efficiently using human demonstration to learn complex policies. We differ by using an optimal planner allowing us to cheaply "label" infinitely many training examples.

II. Method

A. Interactive Framework

Our extended THOR [6] framework consists of 10 individual kitchen scenes. Each scene contains an average of 53 distinct objects. We interact with the environment with seven high-level actions (`navigate to, look up, look down, open, close, pick up, put`) which each take an argument (e.g. `open fridge`). The combination of actions and arguments results in an average 80 actions per scene.

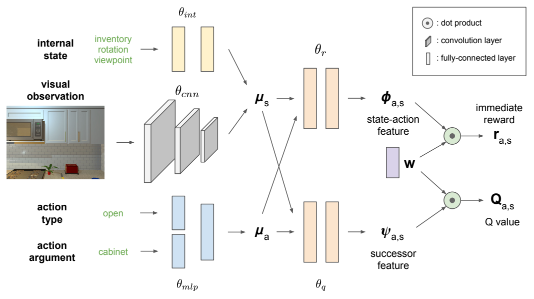

Fig. 1. An overview of the network architecture of our successor representation (SR) model. Our network takes in the current state as well as a specific action and predicts an immediate reward $r_{a,s}$ as well as a discounted future reward $Q_{a,s}$, performing this evaluation for each action. The learned policy π takes the argmax over all Q values as its chosen action.

B. Successor Representation

Successor representation (SR), proposed by Dayan [1], uses a value-based formulation for policy learning, differing from traditional Q learning by factoring the value function into a dot product of two components: a reward predictor vector $\mathbf{w}$ and a predictive successor feature $\psi(s,a)$. To derive the SR formulation, we start by factoring the immediate rewards:

$$r(s,a) = \phi(s,a)^T \mathbf{w} \tag{1}$$

where $\phi(s,a)$ is a *state-action feature*. We expand the standard Bellman equation using this reward factorization:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}^\pi[\sum_{i=0}^{\infty} \gamma^i r(s_i,a_i) | s_0 = s, a_0 = a] \\ &= \mathbb{E}^\pi[\sum_{i=0}^{\infty} \gamma^i \phi(s_i,a_i)^T \mathbf{w} | s_0 = s, a_0 = a] \\ &= \mathbb{E}^\pi[\sum_{i=0}^{\infty} \gamma^i \phi(s_i,a_i) | s_0 = s, a_0 = a]^T \mathbf{w} \\ &= \psi^\pi(s,a)^T \mathbf{w} \end{aligned} \tag{2}$$

Intuitively, the successor feature $\psi^\pi(s,a)$ summarizes the environment dynamics under a policy π. The reward predictor vector $\mathbf{w}$ induces structure similar to an embedding task into

* indicates equal contribution

	Easy		Medium		Hard	
	Success Rate	**Mean (σ) Episode Length**	**Success Rate**	**Mean (σ) Episode Length**	**Success Rate**	**Mean (σ) Episode Length**
Random Action	1.00	696.33 (744.71)	0.00	-	0.04	2827.08 (927.84)
A3C [5]	0.96	101.12 (151.04)	0.00	-	0.04	2674.29 (4370.40)
Basic IL	1.00	**2.42** (0.70)	0.65	256.32 (700.78)	**0.65**	475.86 (806.42)
SR IL (ours)	1.00	2.70 (1.06)	**0.80**	**32.32 (29.22)**	**0.65**	**34.25** (63.81)
Optimal planner	1.00	2.36 (1.04)	1.00	12.10 (6.16)	1.00	14.13 (9.09)

TABLE I
RESULTS OF EVALUATING THE MODEL ON THE EASY, MEDIUM, AND HARD TASKS. EACH TASK HAS A MAXIMUM OF 5,000 STEPS BEFORE FAILURE. FOR EACH TASK, WE EVALUATE HOW MANY OUT OF THE 100 EPISODES WERE COMPLETED (SUCCESS RATE) AND THE MEAN AND STANDARD DEVIATION FOR SUCCESSFUL EPISODE LENGTHS. FAILURES ARE NOT INCLUDED IN THE EPISODE LENGTH CALCULATIONS.

the reward functions. To utilize the successor decomposition, we develop a neural network architecture to learn ϕ, ψ and $\mathbf{w}$, illustrated in Fig. 1.

C. *Imitation Learning*

Given a task, we generate a state-action trajectory:

$$\mathcal{T} = \{(s_0, a_0), \{(s_1, a_1), \dots, (s_{T-1}, a_{T-1}), (s_T, \emptyset)\} \quad (3)$$

using the planner starting from an initial state-action pair (s_0, a_0). Each state-action pair is associated with a true immediate reward $r^*_{s,a}$ and a true Q value $Q^*_{s,a}$, computed using the optimal plan. We use the mean squared loss function to minimize the error of reward predictions:

$$\mathcal{L} = \frac{1}{T}\sum_{i=0}^{T-1}(r^*_{s,a} - \phi^T_{s,a}\mathbf{w})^2 + (Q^*_{s,a} - \psi^T_{s,a}\mathbf{w})^2 \quad (4)$$

To fully explore the state space, we deviate from the optimal plan with probability ϵ. After every action, we recompute the new optimal trajectory. Using this loss signal, we train the whole SR network end-to-end on a large collection of state-action pairs.

III. EXPERIMENTS

A. *Quantitative Evaluation*

We examine the effectiveness of our model and baseline methods on a set of tasks that require three levels of planning complexity. **Easy** tasks consist of a single `navigate` and either an `open` or a `close` action. **Medium** tasks require multiple objects to be collected from known locations. **Hard** tasks require the agent to first find an object from an unknown location, and move it to a target receptacle.

We compare our SR model with the state-of-the-art deep RL model, A3C [5] to establish a strong baseline for reinforcement learning. We use the same architecture to obtain an imitation learning baseline, trained with a softmax classifier to predict the planner action given an input (Basic IL).

Table I summarizes the results of these experiments. Pure RL-based methods struggle with the medium and hard tasks because the action space is so large that naïve exploration rarely, if ever, succeeds. Overall, the SR method outperforms (or ties) the baselines across all three task difficulties.

B. *Task Transfer*

One benefit of the SR decomposition is its ability to transfer to new tasks by retraining $\mathbf{w}$ while freezing the rest of the network. We examine the sample efficiency of adapting a trained SR model to multiple novel tasks within the same scene. We take a policy for searching for a bowl and train a new $\mathbf{w}$ for four additional items. Results are shown in Fig. 2.

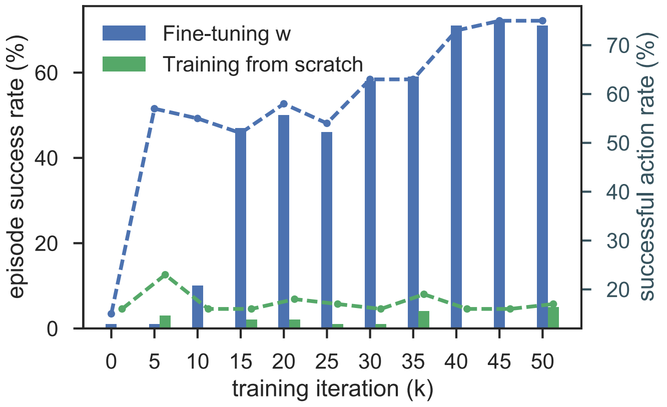

Fig. 2. We compare updating $\mathbf{w}$ with retraining the whole network for new hard tasks in the same scene. By using successor features, we can quickly learn an accurate policy for the new item. Bar charts correspond to the episode success rates, and line graphs correspond to successful action rate.

By fine-tuning $\mathbf{w}$, the model quickly adapts to new tasks. In contrast, the model trained from scratch takes substantially longer to converge. We also experiment with fine-tuning the entire model, and it suffers from similar slow convergence.

IV. CONCLUSION

In this work, we presented a novel architecture using the successor representation to train a visual semantic planner. Using imitation learning on an optimal planner, we achieved near-optimal results on complex planning procedures.

REFERENCES

[1] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 1993.
[2] Christian Dornhege et al. Integrating symbolic and geometric planning for mobile manipulation. In *SSRR*, 2009.
[3] Siddharth Srivastava et al. Combined task and motion planning through an extensible planner-independent interface layer. In *ICRA*, 2014.
[4] Stéphane Ross et al. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, page 6, 2011.
[5] Volodymyr Mnih et al. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.
[6] Yuke Zhu et al. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *ICRA*, 2017.