LEARNING AND PLANNING WITH A SEMANTIC MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Building deep reinforcement learning agents that can generalize and adapt to unseen environments remains a fundamental challenge for AI. This paper describes progresses on this challenge in the context of man-made environments, which are visually diverse but contain intrinsic semantic regularities. We propose a hybrid model-based and model-free approach, *LEArning and Planning with Semantics (LEAPS)*, consisting of a multi-target sub-policy that acts on visual inputs, and a Bayesian model over semantic structures. When placed in an unseen environment, the agent plans with the semantic model to make high-level decisions, proposes the next sub-target for the sub-policy to execute, and updates the semantic model based on new observations. We perform experiments in visual navigation tasks using House3D, a 3D environment that contains diverse human-designed indoor scenes with real-world objects. LEAPS outperforms strong baselines that do not explicitly plan using the semantic content.

1 INTRODUCTION

Deep reinforcement learning (DRL) has undoubtedly witnessed strong achievements in recent years (Silver et al., 2016; Mnih et al., 2015; Levine et al., 2016). However, training an agent to solve tasks in a new unseen scenario, usually referred to as its *generalization ability*, remains a challenging problem (Geffner, 2018; Lake et al., 2017). In model-free RL, the agent is trained to reactively make decisions from the observations, e.g., first-person view, via a black-box policy approximator. However the generalization ability of agents trained by model-free RL is limited, and is even more evident on tasks that require extensive planning (Tamar et al., 2016; Kansky et al., 2017). On the other hand, model-based RL learns a dynamics model, predicting the next observation when taking an action. With the model, sequential decisions can be made via planning. However, learning a model for complex tasks and with high dimensional observations, such as images, is challenging. Current approaches for learning action-conditional models from video are only accurate for very short horizons (Finn & Levine, 2017; Ebert et al., 2017; Oh et al., 2015). Moreover, it is not clear how to efficiently adapt such models to changes in the domain.

In this work, we aim to improve the generalization of RL agents in domains that involve high-dimensional observations. Our insight is that in many realistic settings, building a pixel-accurate model of the dynamics is not necessary for planning high-level decisions. There are semantic structures and properties that are shared in real-world man-made environments. For example, rooms in indoor scenes are often arranged by their mutual functionality (*e.g.* bathroom next to bedroom, dining room next to kitchen). Similarly, objects in rooms are placed at locations of practical significance (*e.g.* nightstand next to bed, chair next to table). Humans often make use of such structural priors when exploring a new scene, or when making a high-level plan of actions in the domain. However, pixel-level details are still necessary for carrying out the high-level plan. For example, we need high-fidelity observations to locate and interact with objects, open doors, etc.

Based on this observation, we propose a hybrid framework, *LEArning and Planning with Semantics (LEAPS)*, which consists of a model-based component that works on the semantic level to pursue a high-level target, and a model-free component that executes the target by acting on pixel-level inputs. Concretely, we (1) train model-free multi-target subpolicies in the form of neural networks that take the first-person views as input and sequentially execute sub-targets towards the final goal; (2) build a *semantic* model in the form of a latent variable model that only takes *semantic signals*, i.e., low-dimensional binary vectors, as input and is dynamically updated to plan the next sub-target. LEAPS has following advantages: (1) via model-based planning, generalization ability is improved; (2) by learning the *prior* distribution of the latent variable model, we capture the semantic consistency among the environments; (3) the semantic model can be efficiently updated by posterior inference

when the agent is exploring the unseen environment, which is effective even with very few exploration experiences thanks to the Bayes rule; and (4) the semantic model is lightweight and fully interpretable.

Our approach requires observations that are composed of both pixel-level data and a list of semantic properties of the scene. In general, automatically extracting high-level semantic structure from data is difficult. As a first step, in this work we focus on domains where obtaining semantics is easy. In particular, we consider real-world environments for which strong object detectors are available (He et al., 2017). An example of such environments is House3D which contains 45k real-world 3D scenes (Wu et al., 2018). House3D provides a diverse set of scene layouts, object types, sizes and connectivity, which all conform to a consistent "natural" semantics. Within these complex scenes, we tackle navigation tasks within novel indoor scenes. Note that this problem is extremely challenging as the agent needs to reach far-away targets which can only be completed effectively if it can successfully reason about the overall structure of the new scenario. Lastly, we emphasize that although we consider navigation as a concrete example in this work, our approach is general and can be applied to other tasks for which semantic structures and signals are available

Our extensive experiments show that our LEAPS framework outperforms strong model-free RL approaches, even when the semantic signals are given as input to the policy. Furthermore, the relative improvements of LEAPS over baselines become more significant when the targets are further away from the agent's birthplace, indicating the effectiveness of planning on the learned semantic model.

2 Related Work

Most deep RL agents are tested in the same training environments (Mirowski et al., 2016), disregarding generalization. While limited, robust training approaches have been proposed to enforce an agent's generalization ability, such as domain randomization (Tobin et al., 2017) and data augmentation by generating random mazes for training (Oh et al., 2017; Parisotto & Salakhutdinov, 2017). In our work, we use a test set of novel unseen environments, where an agent cannot resort to memorization or simple pattern matching to solve the task.

Meta-learning has shown promising results for fast adaptation to novel environments. Methods include learning a good initialization for gradient descent (Finn et al., 2017) or learning a neural network that can adapt its policy during exploration (Duan et al., 2016; Mishra et al., 2017). We propose to learn a Bayesian model over the semantic level and infer the posterior structure via the Bayes rule. Our approach (1) can work even without any exploration steps in a new environment and (2) is interpretable and can be potentially combined with any graph-based planning algorithm.

Our work can be viewed as a special case of hierarchical reinforcement learning (HRL). Unlike other approaches (Vezhnevets et al., 2017; Bacon et al., 2017), in our work high-level planning is performed based on the semantic signals. With orders of magnitudes fewer parameters, our approach is easier to learn compared to recurrent controllers.

LEAPS assumes a discrete semantic signal in addition to the continuous state. A similar assumption is also adopted in (Zhang et al., 2018), where the discrete signals are called "attributes" and used for planning to solve compositional tasks within the same fully observable environment. (Riedmiller et al., 2018) use additional discrete signals to tackle the sparse reward problem. The schema network (Kansky et al., 2017) further assumes that even the continuous visual signal can be completely represented in a binary form and therefore directly runs logical reasoning on the binary states.

For evaluating our approach, we focus on the problem of visual navigation, which has been studied extensively (Leonard & Durrant-Whyte, 1992). Classical approaches build a 3D map of the scene using SLAM, which is subsequently used for planning (Fox et al., 2005). More recently, end-to-end approaches have been applied to tackle various domains, such as maze (Mirowski et al., 2016), indoor scenes (Zhu et al., 2017) and Google street view (Mirowski et al., 2018). Evidently, navigation performance deteriorates as the agent's distance from the target increases (Zhu et al., 2017; Wu et al., 2018). To aid navigation and boost performance, auxiliary tasks (Mirowski et al., 2016; Jaderberg et al., 2016) are often introduced during training. Another direction for visual navigation is to use a recurrent neural network and represent the memory in the form of a 2D spatial map (Khan et al., 2018; Parisotto & Salakhutdinov, 2017; Tamar et al., 2016; Gupta et al., 2017) such that a differentiable planning computation can be performed on the spatial memory. Our approach considers more general graph structures beyond dense 2D grids and captures relationships between *semantic* signals, which we utilize as an informative latent structure in semantically rich environments like House3D.

Similar to our work, Savinov *et al*. (Savinov et al., 2018) constructs a graph of nodes corresponding to different locations of the environment. However, they rely on a pre-exploration step within the test scene and build the graph completely from the pixel space. In LEAPS, we use semantic knowledge and learn a prior over the semantic structures that are shared across real-world scenes. This allows us to directly start solving for the task at hand without any exploratory steps.

3 BACKGROUND

We assume familiarity with standard DRL notations. Complete definitions are in Appendix A.

Environment: We consider a *contextual Markov decision process* (Hallak et al., 2015) $E(c)$ defined by $E(c) = (\mathcal{S}, \mathcal{A}, P(s'|s, a; c), r(s, a; c))$. Here c represents the objects, layouts and any other *semantic* information describing the environment, and is sampled from $\mathcal{C}$, the distribution of possible semantic scenarios. For example, c can be intuitively understood as encoding the complete map for navigation, or the complete object and obstacle layouts in robotics manipulations, not known to the agent in advance, and we refer to them as the *context*.

Semantic Signal: At each time step, the agent observes from s a tuple (s_o, s_s), which consists of: (1) a high-dimensional observation s_o, *e.g.* the first person view image, and (2) a low-dimensional discrete *semantic signal* s_s, which encodes semantic information. Such signals are common in AI, *e.g.* in robotic manipulation tasks s_s indicates whether the robot is holding an object; for games it is the game status of a player; in visual navigation it indicates whether the agent reached a landmark; while in the AI planning literature, s_s is typically a list of *predicates* that describe binary properties of objects. We assume s_s is provided by an oracle function, which can either be directly provided by the environment or extracted by some semantic extractor.

Generalization: Let $\mu(a|\{s^{(t)}\}_t; \theta)$ denote the agent's policy parametrized by θ conditioned on the previous states $\{s^{(t)}\}_t$. The objective of *generalization* is to train a policy on training environments $\mathcal{E}_{\text{train}}$ such that the accumulative reward $R(\mu(\theta); c)$ on test set $\mathcal{E}_{\text{test}}$ is maximized.

4 LEARNING AND PLANNING WITH A SEMANTIC MODEL

The key motivation of LEAPS is the fact that while each environment can be different in visual appearances, there are structural similarities between environments that can be captured as a probabilistic graphical model over the semantic information. On a high level, we aim to learn a Bayesian model $\mathbf{M}^\star(\mathcal{D}, c)$ that captures the semantic properties of the context c, from the agent's exploration experiences $\mathcal{D}$. Given a new environment $E(c')$, the agent computes the posterior $P(c'|\mathcal{D}', \mathbf{M}^\star)$ for the unknown context c' via the learned model $\mathbf{M}^\star$ and its current experiences $\mathcal{D}'$. This allows the agent to plan according to its belief of c' to reach the goal more effectively. Thanks to the Bayes rule, this formulation allows probabilistic inference even with *limited* (or even no) exploration experiences.

Learning an accurate and complete Bayesian model $\mathbf{M}^\star(\mathcal{D}, c)$ can be challenging. We learn an *approximate* latent variable model $\mathbf{M}(y, z; \psi)$ parameterized by ψ with observation variable y and latent variable z that *only* depend on the *semantic signal* s_s. Suppose we have K different semantic signals $T_1, \ldots, T_K$ and $s_s \in \{0, 1\}^K$ where $s_s(T_k)$ denotes whether the kth signal T_k (e.g., landmarks in navigation) is reached or not. Assuming T_1 is the final goal of the task, from any state s, we want to reach some final state s' with $s'_s(T_1) = 1$. In this work, we consider navigation as a concrete example, which can be represented as reaching a state where a desired semantic signal becomes 'true'. We exploit the fact that navigation to a target can be decomposed into reaching several way points on way to the target, and therefore can be guided by planning on the semantic signals, i.e., arrival at particular way points.

4.1 THE SEMANTIC MODEL

Note that there can be 2^K different values for s_s. For efficient computation, we assume *independence* between different semantic signals T_k: we use a binary variable $z_{i,j}$ to denote whether some state s' with $s'_s(T_j) = 1$ can be "*directly reached*", i.e., by a few exploration steps, from some state s with $s_s(T_i) = 1$, *regardless* of other signals $T_k \notin \{T_i, T_j\}$. In addition, we also assume *reversibility*, i.e., $z_{i,j} = z_{j,i}$, so only $K(K-1)/2$ latent variables are needed. Before entering the unknown environment, the agent does not know the true value of $z_{i,j}$, but holds some prior belief $P(z_{i,j})$, defined by $z_{i,j} \sim \text{Bernoulli}(\psi^{\text{prior}}_{i,j})$, where $\psi^{\text{prior}}_{i,j}$ is some parameter to be learned. After some exploration steps, the agent receives a noisy observation $y_{i,j}$ of $z_{i,j}$, i.e., whether a state s' with

$s'_s(T_j) = 1$ is reached. We define the observation model $P(y_{i,j}|z_{i,j})$ as follows:

$$y_{i,j} \sim \begin{cases} \text{Bernoulli}(\psi^{\text{obs}}_{i,j,0}) & \text{if } z_{i,j} = 0 \\ \text{Bernoulli}(1 - \psi^{\text{obs}}_{i,j,1}) & \text{if } z_{i,j} = 1 \end{cases} \tag{1}$$

At any time step, the agent hold an overall belief $P(z|\mathcal{Y})$ of the semantic structure of the unknown environment, based on its experiences $\mathcal{Y}$, namely the samples of y.

4.2 Combining the Semantic Model with Multi-Target Sub-policies

Multi-target sub-policies: With our semantic model, we correspondingly learn multi-target sub-policies $\mu(a|\{s_o^{(t)}\}_t; T_i, \theta)$ taking s_o as input such that $\mu(T_i, \theta)$ is particularly trained for sub-target T_i, i.e., reaching a state s' with $s'_s(T_i) = 1$. Hence the semantic model can be treated as a model-based *planning module* that picks an intermediate sub-target for the sub-policies to execute so that the final target T_1 can be reached with the highest probability. Learning the multi-target sub-policies can be accomplished by any standard deep RL method on $\mathcal{E}_{\text{train}}$.

Inference and planning on $\mathbf{M}$: We assume the agent explores the current environment for a *short* horizon of N steps and receives semantic signals ${s_s}^{(1)}, \ldots, {s_s}^{(N)}$. Then we compute the bit-OR operation over these binary vectors $B = {s_s}^{(1)}$ `OR` $\ldots$ `OR` ${s_s}^{(N)}$. By the reversibility assumption, for T_i and T_j with $B(T_i) = B(T_j) = 1$, we know that T_i and T_j are "directly reachable" for each other, namely a sample of $y_{i,j} = 1$, and otherwise $y_{i,j} = 0$. Combining all the history samples of y and the current batch from B as $\mathcal{Y}$, we can perform posterior inference $P(z|\mathcal{Y})$ by the Bayes rule. By the independence assumption, we can individually compute the belief of each latent variable $z_{i,j}$, denoted by $\hat{z}_{i,j} = P(z_{i,j}|\mathcal{Y}_{i,j})$. Given the current beliefs $\hat{z}_{i,j}$, the current semantic signals s_s and the goal T_1, we search for an optimal plan $\tau^* = \{\tau_0, \tau_1, \ldots, \tau_{m-1}, \tau_m\}$, where $\tau_m = 1$, so that the joint belief along the path from some current signal to the goal is maximized:

$$\tau^\star = \arg\max_{\tau} s_s(T_{\tau_0}) \prod_{t=1}^{m} \hat{z}_{\tau_{t-1},\tau_t}. \tag{2}$$

After obtaining $\tau^\star$, we execute the sub-policy for the next sub-target $T_{\tau_1^\star}$, and then repeatedly update the model and replan every N steps.

4.3 Learning the Semantic Model

The model parameters ψ have two parts: ψ^{prior} for the prior of z and ψ^{obs} for the noisy observation y. Note that ψ^{obs} is related to the performance of the sub-policies $\mu(\theta)$: if $\mu(\theta)$ has a high success rate for reaching sub-targets, ψ^{obs} should be low; when $\mu(\theta)$ is poor, ψ^{obs} should be higher (cf. Eq. (1)).

Learning ψ^{prior}: We learn ψ^{prior} from $\mathcal{E}_{\text{train}}$. During training, for each pair of semantic signals T_i and T_j, we run *random explorations* from some state s with $s(T_i) = 1$. If eventually we reach some state s' with $s'(T_j) = 1$, we consider T_i and T_j are reachable and therefore a *positive* sample $z_{i,j} = 1$; otherwise a *negative* sample $z_{i,j} = 0$. Suppose $\mathcal{Z}$ denotes the samples we obtained for z from $\mathcal{E}_{\text{train}}$. We run maximum likelihood estimate for ψ^{prior} by maximizing $L_{\text{MLE}}(\psi^{\text{prior}}) = P(\mathcal{Z}|\psi^{\text{prior}})$.

Learning ψ^{obs}: There is no direct supervision for ψ^{obs}. However, we can evaluate a particular value of ψ^{obs} by policy evaluation on the validation environments $\mathcal{E}_{\text{valid}}$. We optimize the accumulative reward $L_{\text{valid}}(\psi^{\text{obs}}) = \mathbb{E}_{E(c)\in\mathcal{E}_{\text{valid}}}\left[R(\mu(\theta), M(\psi); c)\right]$, with the semantic model $M(\psi)$. Analytically optimizing L_{valid} is hard. Instead, we apply local search in practice to find the optimal ψ^{obs}.

5 RoomNav: A 3D Navigation Task for RL Generalization

RoomNav is a concept driven navigation task based on the House3D environment (Wu et al., 2018). In RoomNav, the agent is given a concept target, i.e., a room type, and needs to navigate to find the target room. RoomNav pre-selected a fixed set of target room types and provides a training set of 200 houses, a testing set of 50 houses and a small validation set of 20 houses.

Semantic signals: We choose the $K = 8$ most common room types as our semantic signals, such that $s_s(T_i)$ denotes whether the agent is currently in a room with type T_i[1]. When given a target T_i,

[1]We also treat $s_s = \mathbf{0}$ as a special semantic signal. So $\mathbf{M}$ actually contains $K + 1$ signals.

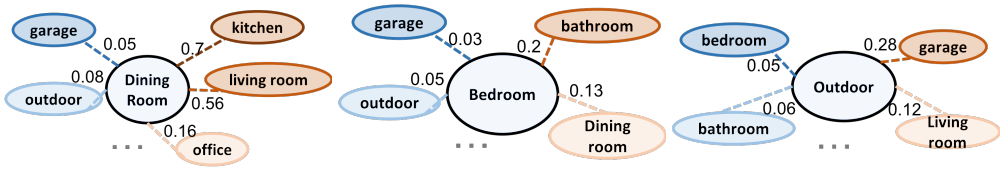

Figure 1: Visualization of learned semantic prior of $\mathbf{M}(\psi)$: the most and least likely nearby rooms for dining room (L), bedroom (M) and outdoor (R), with numbers denoting ψ^z, i.e., the probability of two rooms connecting to each other.

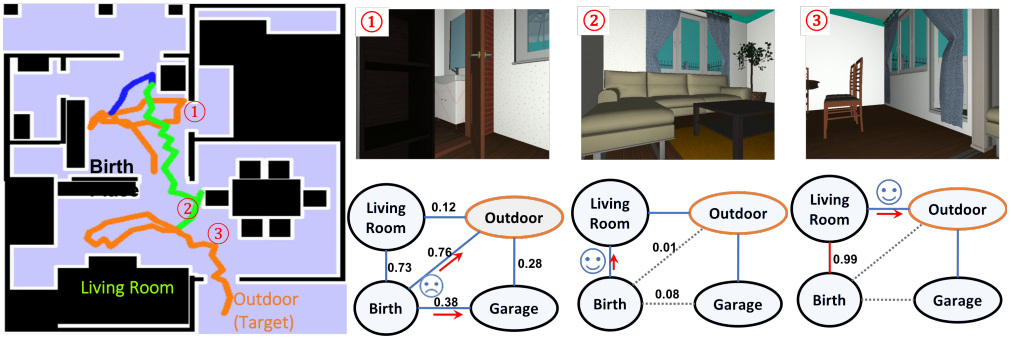

Figure 2: Example of a successful trajectory. The agent is spawned inside the house, targeting "outdoor". **Left**: the 2D top-down map with sub-target trajectories ("outdoor" – orange; "garage" – blue; "living room" – green); **Right, 1st row**: RGB visual image; **Right, 2nd row**: the posterior of the semantic graph and the proposed sub-targets (red arrow). Initially, the agent starts by executing the sub-policy "outdoor" and then "garage" according to the prior knowledge (**1st graph**), but both fail (top orange and blue trajectories in the map). After updating its belief that garage and outdoor are not nearby (grey edges in the **2nd graph**), it then executes the "living room" sub-policy with success (red arrow in the **2nd graph**, green trajectory). Finally, it executes "outdoor" sub-policy again, explores the living room and reaches the goal (**3rd graph**, bottom orange trajectory).

reaching a state s with $s_s(T_i) = 1$ becomes our final goal. House3D provides bounding boxes for rooms, which can be directly used as the oracle for semantic signals. We can also train a room type classifier to extract the semantics signals, which is evaluated in Sec. 6.4.

The semantic model and sub-policies: In navigation, the reachability variable $z_{i,j}$ can naturally represent the connectivity between room type T_i and room type T_j[2]. We run random explorations in training houses between rooms to collect samples for learning ψ^{prior}. For learning ψ^{obs}, we perform a grid search and evaluate on the validation set. For sub-policies, we learn target driven LSTM policies by A3C (Mnih et al., 2016) with shaped reward on $\mathcal{E}_{\text{train}}$. More details are in Appendix. E.

6 Experiments

In this section, we experiment on RoomNav and try to answer the following questions: **(1)** Does the learned prior distribution capture meaningful semantic consistencies? **(2)** Does our LEAPS agent generalize better than the model-freel RL agent that *only* takes image input? **(3)** Our LEAPS agent takes additional semantic signals as input. How does LEAPS compare to other model-free RL approaches that *also* take the semantic signals as part of the inputs but in a different way from our semantic model? For example, what about replacing our semantic model with a complicated RNN controller? **(4)** House3D provides labels for rooms (although noisy). Can our approach still work if we extract the semantic signals from a trained neural classifier for room types?

6.1 The Learned Prior of the Semantic Model

We visualize the learned prior $P(z|\psi^{\text{prior}})$ in Fig. 1 with 3 room types and their most and least likely connected rooms. The learned prior indeed captures reasonable relationships: bathroom is likely to connect to a bedroom; kitchen is often near a dining room while garage is typically outdoor.

[2] A house can have multiple rooms of the same type. But even this simplification improves generalization.

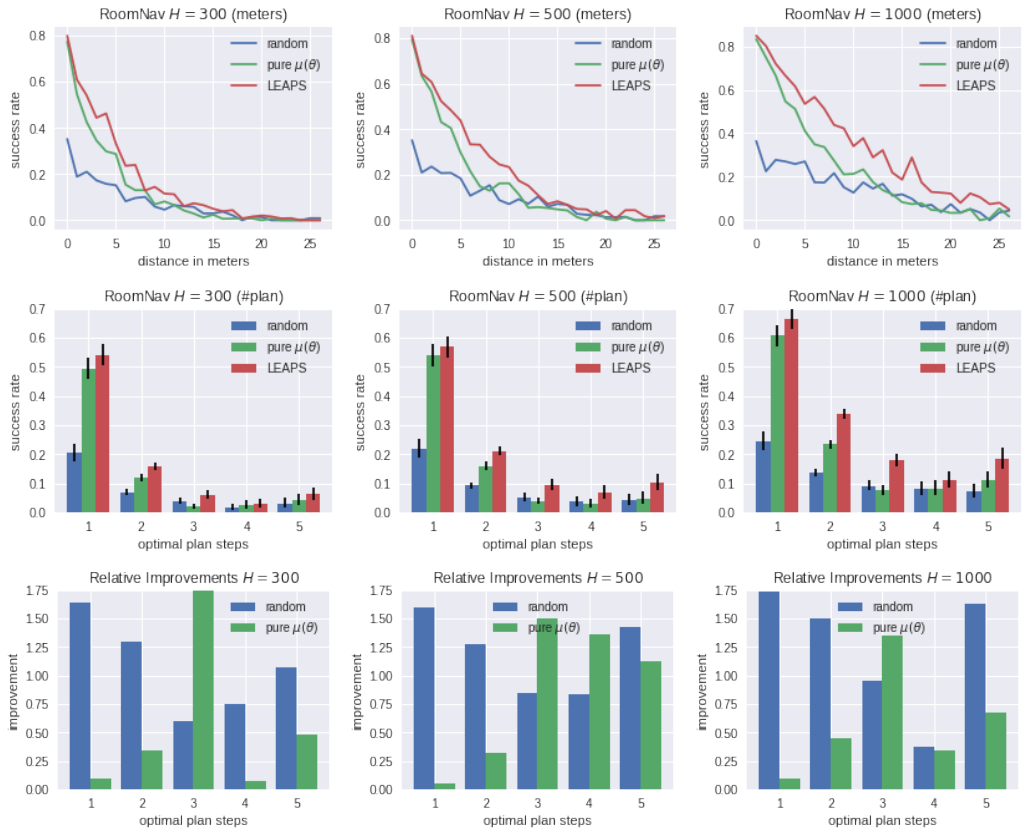

Figure 3: Comparison with model-free baselines (Sec. 6.2). We evaluate performance of random policy (blue), model-free RL baseline (pure $\mu(\theta)$, green) and our LEAPS agent (red), with increasing horizon H from left to right (**left**: $H = 300$; **middle**: $H = 500$; **right**: $H = 1000$). Each row shows a particular metric. **Top row**: success rate (y-axis) w.r.t. the distance in meters from the birthplace to target room (x-axis); **middle row**: success rate with confidence interval (y-axis) w.r.t. the shortest planning distance in the ground truth semantic model (x-axis); **bottom row**: relative improvement of LEAPS over the baseline (y-axis) w.r.t. the optimal plan distance (x-axis). As the number of planning computations, i.e., H/N, increases (from left to right), LEAPS agent outperforms baselines more. LEAPS also has higher relative improvements for faraway targets.

6.2 Comparison with Model-Free RL Baselines

We follow the evaluation process in (Wu et al., 2018) and measure the testing success rate on $\mathcal{E}_{\text{test}}$. More details are in Appendix C and D. We compare our LEAPS agent with two baselines (1) random policy (denoted by "random") and (2) model-free RL agent that only takes in image input s_o and executes $\mu(T_i, \theta)$ throughout the episode (denoted by "pure $\mu(\theta)$"). For LEAPS agent, we set $N = 30$, i.e., update the semantic model every 30 steps. We experiment on different horizons $H = 300, 500, 1000$ and evaluate the success rate and relative improvements of our LEAPS agent over the baselines in Fig. 3. As the number of planning computations, H/N, increases, our LEAPS agent outperforms the baselines more significantly in success rate. Note that the best relative improvements are achieved for targets neither too faraway nor too close, i.e., optimal plan steps equal to 3 or 4. Interestingly, we observe that there is a small success rate increase for targets that are 5 plan steps away. We suspect that this is because it is rare to see houses that has a diameter of 5 in the semantic model (imagine a house where you need to go through 5 rooms to reach a place). Such houses may have structural properties that makes navigation easier. Fig. 2 shows an example of a success trajectory of our LEAPS agent. We visualize the progression of the episode, describe the plans and show the updated graph after exploration.

6.3 Comparing to Semantic-Aware Agents without a Graph Representation

Here we consider two semantic-aware agents that also takes the semantic signals as input.

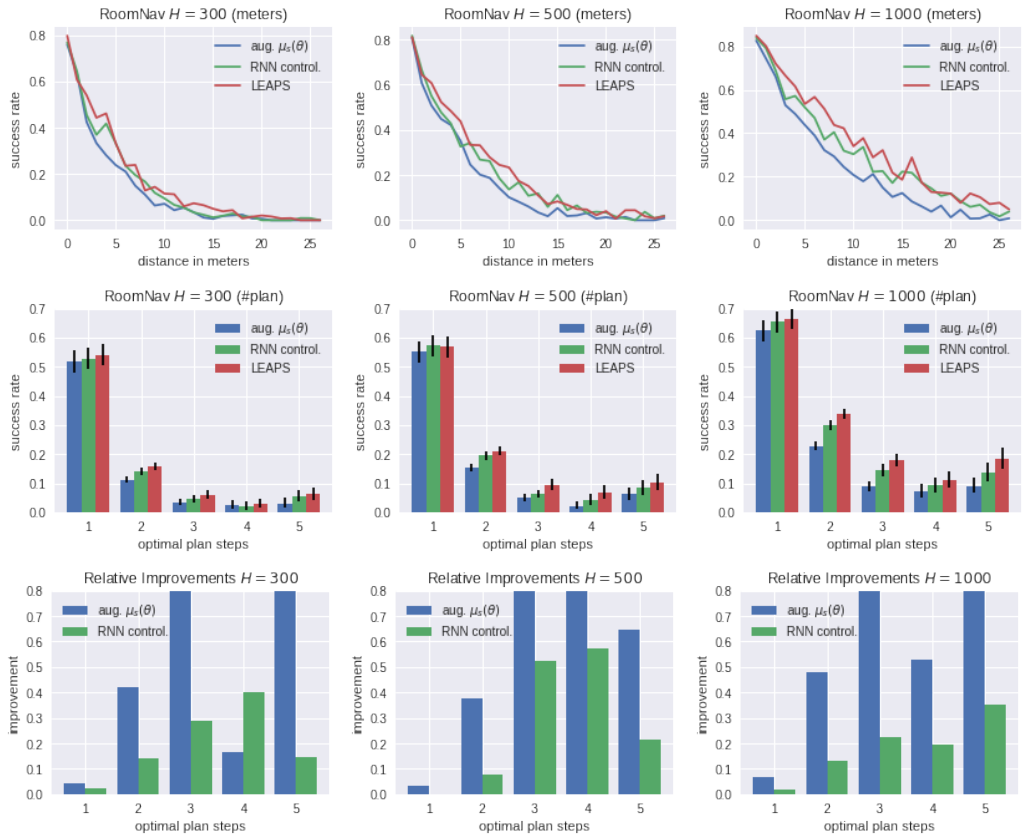

Figure 4: Comparison with semantic-aware policies (Sec. 6.3). We evaluate performance of the semantic augmented model-free agent ("aug. $\mu_s(\theta)$", blue), the HRL agent with the same sub-policies as LEAPS but with an LSTM controller ("RNN control.", green) and our LEAPS agent (red), with increasing horizon H from left to right (**left**: $H = 300$; **middle**: $H = 500$; **right**: $H = 1000$). **Top row**: success rate (y-axis) w.r.t. the distance in meters from birthplace to target (x-axis); **middle row**: success rate with confidence interval (y-axis) w.r.t. the shortest planning distance in the ground truth semantic model (x-axis); **bottom row**: relative improvements of LEAPS over the baselines (y-axis) w.r.t. the optimal plan distance (x-axis). Our LEAPS agent outperforms both of the baselines. Note that even though the LSTM controller has two orders of magnitudes more parameters than our semantic model $\mathbf{M}$, our LEAPS agent still performs better, especially for faraway targets.

Semantic augmented agents: We train new sub-policies $\mu_s(\theta_s)$ taking both s_o and s_s as input.

HRL agents with a RNN controller: Note that updating and planning on M (Eq. 2) only depend on (1) the current semantic signal s_s, (2) the target T_i, and (3) the accumulative bit-OR feature B. Hence, we fixed the same set of sub-policies $\mu(\theta)$ used by our LEAPS agent, and train an LSTM controller with 50 hidden units on $\mathcal{E}_{\text{train}}$ that takes all the necessary semantic information, and produce a sub-target every N steps. Training details are in Appendix G. Note that the only difference between our LEAPS agent and this HRL agent is the *representation* of the planning module. The LSTM controller has access to exactly the same semantic information as our model M and uses a much more complicated neural model. Thus we expect it to perform competitively to our LEAPS agent.

The results are shown in Fig. 4, where our LEAPS agent outperforms both baselines. The semantic augmented policy $\mu_s(\theta_s)$ does not improve much on the original $\mu(\theta)$. For the HRL agent with an LSTM controller, the LEAPS agent achieves higher relative improvements for faraway targets, and also has the following advantages: (1) M can be learned more efficiently with much fewer parameters: an LSTM with 50 hidden units has over 10^4 parameters while $M(\psi)$ only has 38 parameters[3]; (2) M can adapt to new sub-policies $\mu(\theta')$ with little finetuning (ψ^{prior} remains unchanged) while the LSTM controller needs to re-train; (3) the model M and the planning procedure are fully interpretable.

[3]We assign the same value to all $\psi^{\text{obs}}_{i,j,c}$ for each $c \in \{0, 1\}$. See more in Appendix F.

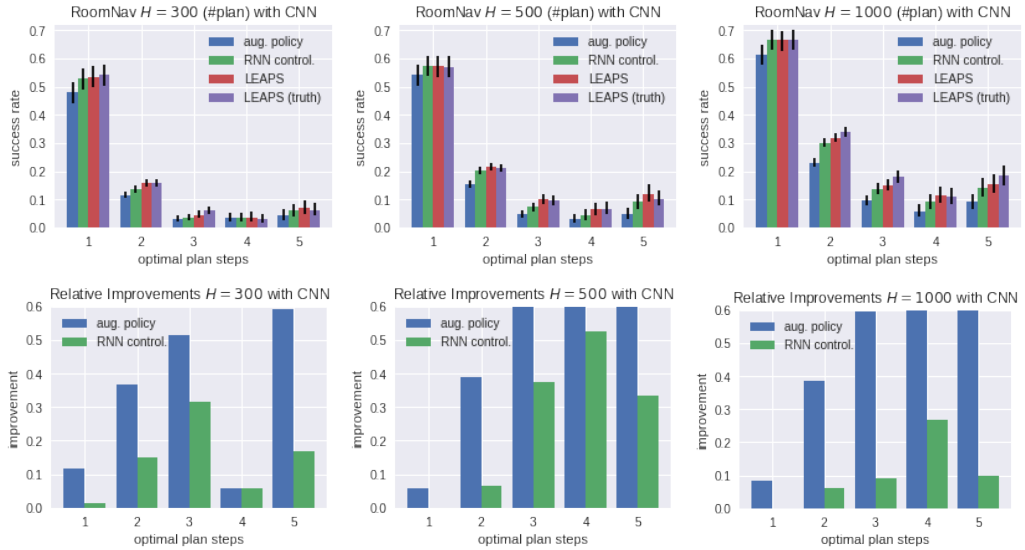

Figure 5: Performance of semantic-aware agents using a CNN semantic extractor (Sec. 6.4). We evaluate performance of the semantic augmented model-free agent ("aug. $\mu_s(\theta)$", blue), the HRL agent with an LSTM controller ("RNN control.", green), our LEAPS agent (red) as well as the reference LEAPS agent using ground truth semantic signals (purple), with increasing horizon H from left to right (**left**: $H = 300$; **middle**: $H = 500$; **right**: $H = 1000$). **Top row**: success rate with confidence interval (y-axis) w.r.t. the shortest planning distance in the ground truth semantic model (x-axis); **bottom row**: relative improvements of LEAPS over the baselines (y-axis) w.r.t. the optimal plan distance (x-axis). With a CNN extractor, our LEAPS agent still outperforms the baselines, despite a small performance drop compared to the reference LEAPS agent using ground truth signals.

6.4 Using a neural network to extract semantic information

In previous experiments, we use the bounding box labels from House3D as the semantic signals. Now we train a CNN room type classifier instead to extract semantic signals from visual input. The CNN classifier are trained on $\mathcal{E}_{\text{train}}$ and validated on $\mathcal{E}_{\text{valid}}$. More details are in Appendix H.

We compare our LEAPS agent with the semantic-aware agents using a CNN semantic extractor to produce s_s. The results are shown in Fig. 5, where we also include a reference LEAPS agent using the labels from House3D (purple). Note that all these agents are trained with ground truth semantic signals while only use the CNN extractor during test phase. Our LEAPS agent still outperforms all the baselines and has a comparable performance comparing to the reference LEAPS agent using the ground truth signals.

7 Conclusion and Future Work

In this work, we proposed LEAPS to improve generalization of RL agents in unseen environments with diverse room layouts and object arrangements, while the underlying semantic information is shared with the environments in which the agent is trained on. We adopt a graphical model over semantic signals, which are low-dimensional binary vectors. During evaluation, starting from a prior obtained from the training set, the agent plans on model, explores the unknown environment, and keeps updating the semantic model after new information arrives. For exploration, sub-policies that focus on multiple targets are pre-trained to execute primitive actions from visual input. The semantic model in LEAPS is lightweight, interpretable and can be updated dynamically with little explorations. As illustrated in the House3D environment, LEAPS works well for environments with semantic consistencies – typical of realistic domains. On random environments, e.g., random mazes, LEAPS degenerates to exhaustive search.

Our approach is general and can be applied to other tasks, such as robotics manipulations where semantic signals can be status of robot arms and object locations, or video games where we can plan on semantic signals such as the game status or current resources. In future work we will investigate models for more complex semantic structures.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *International Conference on on Robotics and Automation*, pp. 2786–2793. IEEE, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Dieter Fox, Sebastian Thrun, and Wolfram Burgard. *Probabilistic Robotics*. MIT press, 2005.

Hector Geffner. Model-free, model-based, and general intelligence. *arXiv preprint arXiv:1806.02308*, 2018.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 3, 2017.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2980–2988. IEEE, 2017.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv preprint arXiv:1706.04317*, 2017.

Arbaaz Khan, Clark Zhang, Nikolay Atanasov, Konstantinos Karydis, Vijay Kumar, and Daniel D. Lee. Memory augmented control networks. *ICLR*, 2018.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

John J. Leonard and Hugh F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Norwell, MA, USA, 1992. ISBN 0792392426.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy 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.

Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. Learning to navigate in cities without a map. *arXiv preprint arXiv:1804.00168*, 2018.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

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.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

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*, pp. 2863–2871, 2015.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *arXiv preprint arXiv:1706.05064*, 2017.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *ICLR*, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3D environment. *arXiv preprint arXiv:1801.02209*, 2018.

Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. Composable planning with attributes. 2018.

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 *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3357–3364. IEEE, 2017.

A COMPLETE NOTATIONS AND DEFINITIONS

Environment: We consider a *contextual Markov Decision Process* (Hallak et al., 2015) $E(c)$ defined by $E(c) = (\mathcal{S}, \mathcal{A}, P(s'|s, a; c), r(s, a; c))$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. c represents the objects, layouts and any other *semantic* information describing the environment, and is sampled from $\mathcal{C}$, the distribution of possible semantic scenarios. $r(s, a; c)$ denotes the reward function while $P(s'|s, a; c)$ describes transition probability conditioned on c. For example, c can be intuitively understood as encoding the complete map for navigation, or the complete object and obstacle layouts in robotics manipulations, not known to the agent in advance, and we refer to them as the *context*.

Semantic Signal: At each time step, the agent's observation is a tuple (s_o, s_s), which consists of: (a) a high-dimensional observation s_o, *e.g.* the first person view image, and (b) a low-dimensional *semantic signal* s_s, which encodes semantic information. Such low-dimensional discrete signals are commonly used in AI, *e.g.* in robotic manipulation tasks s_s indicates whether the robot is holding an object; for games it is the game status of a player; in visual navigation it indicates whether the agent reached a landmark; while in the AI planning literature, s_s is typically a list of *predicates* that describe binary properties of objects. We assume s_s is provided by an oracle function, which can either be directly provided by the environment or extracted by some semantic extractor.

Generalization: Let $\mu(a|\{s^{(t)}\}_t; \theta)$ denote the agent's policy parametrized by θ conditioned on the previous states $\{s^{(t)}\}_t$ and $R(\mu(\theta); c)$ denote the accumulative reward of $\mu(\theta)$ in $E(c)$. The objective is to find the best policy that maximizes the *expected* accumulative reward $\mathbb{E}_{c\sim\mathcal{C}}\left[R(\mu(\theta); c, T_i)\right]$. In practice, we sample a disjoint partition of a training set $\mathcal{E}_{\text{train}} = \{E(c_i)\}_i$ and a testing set $\mathcal{E}_{\text{test}} = \{E(c_j)\}_j$, where $\{c_i\}$ and $\{c_j\}$ are samples from $\mathcal{C}$. We train $\mu(\theta)$ with a *shaped* reward r_{train} only on $\mathcal{E}_{\text{train}}$, and measure the *empirical* generalization performance of the learned policy on $\mathcal{E}_{\text{test}}$ with the original unshaped reward (e.g., binary reward of success or not).

B ENVIRONMENT DETAILS

In RoomNav the 8 targets are: kitchen, living room, dining room, bedroom, bathroom, office, garage and outdoor. We inherit the success measure of "see" from (Wu et al., 2018): the agent needs to see some corresponding object for at least 450 pixels in the input frame and stay in the target area for at least 3 time steps.

For the binary signal s_s, we obtain from the bounding box information for each room provided from SUNCG dataset (Song et al., 2017), which is very noisy.

Originally the House3D environment supports 13 discrete actions. Here we reduce it to 9 actions: large forward, forward, left-forward, right-forward, large left rotate, large right rotate, left rotate, right rotate and stay still.

C EVALUATION DETAILS

We following the evaluation setup from (Wu et al., 2018) and measure the success rate on $\mathcal{E}_{\text{test}}$ over 5750 test episodes, which consists of 5000 random generated configurations and 750 specialized for faraway targets to increase the confidence of measured success rate. These 750 episodes are generated such that for each plan-distance, there are at least 500 evaluation episodes. Each test episode has a fixed configuration for a fair comparison between different approaches, i.e., the agent will always start from the same location with the same target in that episode. Note that we always ensure that (1) the target is connected to the birthplace of the agent, and (2) the the birthplace of the agent is never within the target room.

D VISUALIZATION DETAILS

For confidence interval of the measured success rate, we computed it by fitting a binomial distribution.

For optimal plan steps, we firstly extract all the room locations, and then construct a graph where a vertex is a room while an edge between two vertices is the shortest distance between these two rooms. After obtaining the graph and a birthplace of the agent, we compute shortest path from the birthplace to the target on this graph to derive the optimal plan steps.

E DETAILS FOR LEARNING NEURAL SUB-POLICIES

We utilize the same policy architecture as (Wu et al., 2018). It was mentioned in (Wu et al., 2018) that using segmentation mask + depth signals as input leads to relatively better performances for policy learning. So we inherit this setting here. We run A3C with $\gamma = 0.97$, batch size 64, learning rate 0.001 with Adam, weight decay 10^{-5}, entropy bonus 0.1. We backprop through at most 30 time steps. We also compute the squared l_2 norm of logits and added to the loss with a coefficient 0.01. We also normalize the advantage to mean 0 and standard deviation 1.

We run a curriculum learning by increasing the maximum of distance between agent's birth meters and target by 3 meters every 10000 iterations. We totally run 60000 training iterations and use the final model as our learned policy $\mu(\theta)$.

In the original House3D paper, a gated attention module is used to incorporate the target instruction. Here, since we only have $K = 8$ different sub-policies, we simply train an individual policy for each target and we empirically observe that this leads to better performances.

F DETAILS FOR LEARNING THE SEMANTIC MODEL

After evalution on the validation set, we choose to run random exploration for 300 steps to collect a sample of z. For a particular environment, we collect totally 50 samples for each $z_{i,j}$.

For all $i \neq j$, we set $\psi^{\text{obs}}_{i,j,0} = 0.001$ and $\psi^{\text{obs}}_{i,j,1} = 0.15$.

G ADDITIONAL DETAILS FOR TRAINING SEMANTIC-AWARE POLICIES

For the LSTM controller, we ran A2C with batch size 32, learning rate 0.001 with adam, weight decay 0.00001, gamma 0.99, entropy bonus 0.01 and advantage normalization. The reward function is designed as follows: for every subtask it propose, it gets a time penalty of 0.1; when the agent reach the target, it gets a success bonus of 2.

The input of the LSTM controller consists of (1) ${s_s}^{(t)}$ (K bits), (2) B (K bits), (3) last subtask T_k, and (4) the final target T_i. We convert T_i and T_k to a one-hot vector and combine the other two features to feed into the LSTM. Hence the input dimension of LSTM controller is $4K$, namely 32 in RoomNav.

For the semantic augmented LSTM policy, $\mu_s(\theta_s)$, we firstly use the CNN extract visual features from s_o and combine the input semantic features and the visual features as the combined input to the LSTM in the policy.

H ADDITIONAL DETAILS FOR TRAINING THE CNN SEMANTIC EXTRACTOR

We noticed that in order to have a room type classifier, only using the single first person view image is not enough. For example, the agent may face towards a wall, which is not informative, but is indeed inside the bedroom (and the bed is just behind).

So we take the panoramic view as input, which consists of 4 images, $s_o^1, \ldots, s_o^4$ with different first person view angles. The only exception is that for target "outdoor", we notice that instead of using a panoramic view, simply keeping the recent 4 frames in the trajectory leads to the best prediction accuracy. We use an CNN feature extractor to extract features $f(s_o^i)$ by applying CNN layers with kernel size 3, strides $[1, 1, 1, 2, 1, 2, 1, 2, 1, 2]$ and channels $[4, 8, 16, 16, 32, 32, 64, 64, 128, 256]$. We also use relu activation and batch norm. Then we compute the attention weights over these 4 visual features by $l_i = f(s_o^i) W_1^T W_2 \left[f(s_o^1), \ldots, f(s_o^4)\right]$ and $a_i = \text{softmax}(l_i)$. Then we compute the weighted average of these four frames $g = \sum_i a_i f(s_o^i)$ and feed it to a single layer perceptron with 32 hidden units. For each semantic signal, we generate 15k positive and 15k negative training data from $\mathcal{E}_{\text{train}}$ and use Adam optimizer with learning rate $5e$-4, weight decay $1e$-5, batch size 256 and gradient clip of 5. We keep the model that has the best prediction accuracy on $\mathcal{E}_{\text{valid}}$.

For a smooth prediction during testing, we also have a hard threshold and filtering process on the CNN outputs: $s_s(T_i)$ will be 1 only if the output of CNN has confidence over 0.85 for consecutively 3 steps.