Hierarchical Agents by Combining Language Generation and Semantic Goal Directed RL

Bharat Prakash\textsuperscript{1}, Nicholas Waytowich\textsuperscript{2}, Tim Oates\textsuperscript{1}, Tinoosh Mohsenin\textsuperscript{1}

University of Maryland, Baltimore County\textsuperscript{1}
US Army Research Lab \textsuperscript{2}

Abstract

Learning to solve long horizon temporally extended tasks with reinforcement learning has been a challenge for several years now. We believe that it is important to leverage both the hierarchical structure of complex tasks and to use expert supervision whenever possible to solve such tasks. This work introduces an interpretable hierarchical agent framework by combining sub-goal generation using language and semantic goal directed reinforcement learning. We assume access to certain spatial and haptic predicates and construct a simple and powerful semantic goal space. These semantic goal representations act as an intermediate representation between language and raw states. We evaluate our framework on a robotic block manipulation task and show that it performs better than other methods, including both sparse and dense reward functions. We also suggest some next steps and discuss how this framework makes interaction and collaboration with humans easier.

1 Introduction

Deep reinforcement learning has been successful in many tasks, including robotic control, games, energy management, etc. \cite{Mnih2015, Schulman2017, Warnell2018}. However, it has many challenges, such as exploration under sparse rewards, generalization, safety, etc. This makes it difficult to learn good policies in a sample efficient way. Popular ways to tackle these problems include using expert feedback \cite{Christiano2017, Warnell2018} and leveraging the hierarchical structure of complex tasks. There is a long list of prior work which learns hierarchical policies to break down tasks into smaller sub-tasks \cite{Sutton1999, Fruit2017, Bacon2017}. Some of them discover options or sub-tasks in an unsupervised way. On the other hand, using some form of supervision, either by providing details about the sub-tasks, intermediate rewards or high-level guidance is a recent approach \cite{Prakash2021, Jiang2019, Le2018}.

This paper presents a framework for solving long-horizon temporally extended tasks with a hierarchical agent framework using semantic goal representations and goal generation using language. The agent has two levels of control and the ability to easily incorporate expert supervision and intervention. The high-level policy is a small text generation model which generates sub-goals in the form of text commands, given a high level goal and current state. The low-level policy is a goal-conditioned multi-task policy which is able to achieve sub-goals where these goals are specified using a semantic goal representation. There is an intermediate module which converts these text goals to semantic goal representation. The semantic goal representation is constructed using several predicate functions which define the behavior space of the agent. This representation has many benefits because it is much simpler than traditional state-based goal spaces as shown in \cite{Akakzia2020}. The language interface makes the framework more interpretable and easier for an expert to intervene and provide high-level feedback. The sub-goals which are in the form of language can be observed by a human expert and provide corrections if necessary.

We evaluate the framework using a robotic block manipulation environment. Our experiments show that this approach is able to solve different tasks by combining grasping, pushing and stacking blocks. Our contributions can be summarised as follows:

- A hierarchical agent framework where the high-level policy is a language generator and the low-level policy is learned using semantic goal representations.
- A language interface that can map natural language commands to symbolic goals. This module is also a natural interface humans to intervene and provide corrections.
- Evaluation on complex long horizon robotic block manipulation tasks to show feasibility and sample efficiency.

2 Methods

In this section, we present a framework for solving long horizon temporally extended tasks. We first describe the semantic goal representation and low-level policy training. Then, we show how the high-level policy is obtained using the sub-goal instruction generator and solve long horizon tasks.

2.1 Semantic goal representations

We represent goals using a list of semantic predicates which are determined based on domain knowledge. In our case we consider three spatial predicates - close, above, in-bin and one haptic predicate - holding. As demonstrated by Akakzia et al. [2020], these predicates define a much simpler behavior space instead of the traditional more complicated state space. This representation eliminates the need to write reward functions for every desired behavior. All these predicates are binary functions applied to pairs of objects. The close predicate is order-invariant. close($o_1, o_2$) denotes whether objects (in our case blocks) $o_1$ and $o_2$ are close to each other or not. The above predicate is applied to all permutations of objects. above($o_1, o_2$) is used to denote if $o_1$ is above $o_2$. The in-bin predicate is used to denote whether the block is inside the bin. Finally, holding is used to denote if the robot arm is holding an object using holding($o$). With these predicates we can form a semantic representation of the state by simply concatenating all the predicate outputs as shown in Fig 1.

2.2 Training the low-level policy

The low-level policy is trained to perform several individual sub-tasks, which can eventually be used to solve longer high-level tasks. We use Hindsight experience replay (HER) Andrychowicz et al. [2017] along with Soft-Actor critic (SAC) Haarnoja et al. [2018] to train the goal conditioned policy. Goals are sampled from a set of configurations based on the environment where an expert can be used to optionally create a curriculum. The semantic goal representation makes it easier to do both of these things. The agent explores the environment to collect experience and updates its policy using SAC. As stated earlier, there is no need to write reward functions for each desired behavior. A reward can be generated by checking whether the current semantic configuration matches the goal configuration. Example sub-goals for the three environments we use are listed in Table 1.

2.3 Training the high-level policy

The high-level policy is a sub-goal instruction generator which takes in the current state and high-level task description and outputs a sub-goal in the form of a language instruction. It is trained using a
Table 1: Low-level policy is trained on the above set of semantic goals. The semantic goal representation is built using the predicates as described in the previous section. For 2 blocks version, X/Y can be block of color \([red, green]\). For the 3 blocks version it can be of color \([red, green, blue]\).
Figure 2: This figure shows all the tasks we used in our experiments. The top row shows examples of random initial states and bottom row shows the goal states. MC-2: Move 2 blocks closer, MA-2: Move 2 blocks away, Swap-2: Swap 2 stacked blocks, MC-3: Move 3 blocks closer, MA-3: Move 3 blocks away, DC-2 to DC-4: Desk clean up with 2, 3 and 4 blocks.

<table>
<thead>
<tr>
<th>Method</th>
<th>MC-2</th>
<th>MA-2</th>
<th>Swap-2</th>
<th>MC-3</th>
<th>MA-3</th>
<th>DC-2</th>
<th>DC-3</th>
<th>DC-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat Semantic</td>
<td>10%</td>
<td>80%</td>
<td>0%</td>
<td>5%</td>
<td>10%</td>
<td>30%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Flat Continuous</td>
<td>5%</td>
<td>10%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Option Critic</td>
<td>5%</td>
<td>5%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>H-Planner</td>
<td>95%</td>
<td>100%</td>
<td>92%</td>
<td>95%</td>
<td>96%</td>
<td>94%</td>
<td>91%</td>
<td>90%</td>
</tr>
<tr>
<td>H-Lang (Ours)</td>
<td>91%</td>
<td>94%</td>
<td>90%</td>
<td>90%</td>
<td>90%</td>
<td>92%</td>
<td>90%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Table 2: Task completion % This table shows the task completion % for our experiments. The tasks’ names are explained in Figure 4. As seen, our method consistently outperforms all the other baselines. We train each agent for 2M steps and roll out 50 episodes using the trained policy. The values are an average of runs from three different seeds.

3.3 Results

We calculate task completion % for all the tasks using the fully trained agent. We train each agent for 2M steps and roll out 50 episodes using the trained policy. The values are an average of runs from three different seeds. As seen in Table 2, only the 2 methods, H-Planner and our method is able to solve all the tasks. H-planner uses an off the shelf planner which is makes use of the semantic predicates and produces a plan. Although it slightly outperforms our method, it is less interpretable and does not have a natural language interface for humans to intervene.

4 Conclusion

In this paper we show that combining a high-level language generator, semantic goal representations and a low-level goal conditioned reinforcement learning policy is indeed a promising approach to build interpretable hierarchical agents. This also makes it easier for a human to intervene at the high-level to provide appropriate sub-goals using language in case there is a failure in the high-level policy.

There are several directions in which this framework can be extended. With the current state space, we assumed access to predicate functions. But with more complex observation like images, one can learn these predicate functions using a small amount of labelled data. To further demonstrate the capabilities of the framework we plan to perform experiments on more complex environments, real robots and qualitative analysis using human users. To measure the benefits of the language interface more systematically, user studies could be performed with humans and real robots to perform a qualitative analysis. This work is a step towards simple and interpretable hierarchical agents and we hope to build upon it.

References


