Learning compositional tasks from language instructions

Anonymous Author(s)
Affiliation
Address
email

Abstract

Systematic compositionality – the ability to combine learned knowledge and skills to solve novel tasks – is a key aspect of generalization in humans that allows us to understand and perform tasks described by novel language utterances. While progress has been made in supervised learning settings, no work has yet studied compositional generalization of a reinforcement learning agent following natural language instructions in an embodied environment. We develop a set of tasks in a photo-realistic simulated kitchen environment that allow us to study the degree to which a behavioral policy captures the systematicity in language by studying its zero-shot generalization performance on held out natural language instructions. We show that our agent which leverages a novel additive action-value decomposition in tandem with attention-based subgoal prediction is able to exploit composition in text instructions to generalize to unseen tasks.

1 Introduction

Human language is characterized by systematic compositionality: one can combine known components – such as words or phrases – to produce novel linguistic combinations (Chomsky, 2009). This is a key aspect of generalization in humans and enables us to understand and perform tasks specified by novel language utterances over familiar words or phrases. If you know what a "laptop" and a "fridge" are, you can easily understand how to perform the task "place the laptop in the fridge" even if you have never placed a laptop in a fridge.

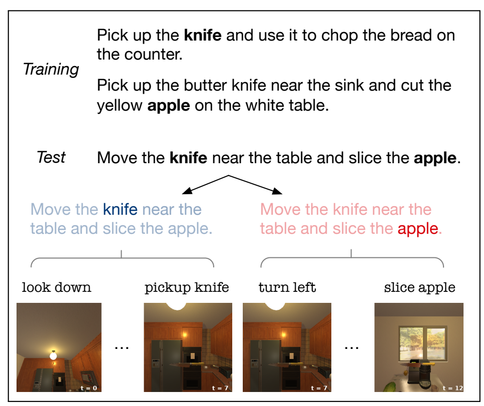

Figure 1: Zero-shot generalization to an unseen task of slicing an apple. The test task is composed of known primitive subtasks – *picking up a knife* and *slicing the apple* – each of which were encountered in training tasks. Our agent learns to decompose a natural language task description into subtasks using attention and executes them using low-level actions.

Prior work studying the linguistic "systematicity" of neural networks have focused on sequence mapping tasks in a supervised learning setting (Lake and Baroni, 2018; Lake, 2019; Andreas, 2019). In this work, we are interested in compositional generalization of a reinforcement learning agent following natural language instructions in an embodied environment. In particular, we explore the hypothesis that a language-conditioned reinforcement learning agent with a compositional inductive bias in its behavioral policy will exhibit systematic generalization to unobserved natural language instructions.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

There has been a flurry of recent work on embodied learning tasks such as question answering (Gupta et al., 2017), navigation (Anderson et al., 2018) and object interaction (Shridhar et al., 2020; Carvalho et al., 2020) in embodied settings. In particular, the ALFRED task (Shridhar et al., 2020) studies agents that exploit detailed natural language instructions to generalize to novel instructions in novel environments at test time. Such existing benchmarks offer limited flexibility to study systematic generalization since (i) the benchmarks were not built for this purpose and it is unclear to what extent systematic generalization skills are required to solve the tasks and (ii) the tasks demand challenging reasoning skills such as visual recognition and planning over large number of time-steps which makes it difficult to study compositional generalization ability in isolation.

In this work we develop a set of tasks in the AI2Thor virtual home environment (Kolve et al., 2017) which test the compositionality of embodied agents. In order to make progress in systematic generalization, we make two simplifying assumptions: we assume access to an oracle object recognizer and we study generalization in a single kitchen layout. This allows us to study the degree to which a policy captures the systematicity in language by studying its zero-shot generalization performance on held out natural language instructions.

Despite these simplifications, agents still need to understand the instruction to figure out the sequence of object interactions that need to be performed and act over many time-steps with limited guidance. In order to succesfully generalize at test time, an agent needs to learn to ground natural language instructions to temporally extended goal-oriented behaviors or "skills" in a compositional manner to perform novel tasks that are compositions of the tasks presented at train time. We leverage this setting to develop and study a policy with an inductive bias for compositionality and show that this enables systematic generalization in the context of combining behavioral skills learned purely from reward without expert demonstrations.

We present an attention-based agent that learns to predict subgoals from language instructions via a learned attention mechanism. Our agent uses these subgoals with a novel policy parametrization which decomposes the action-value function in an additive fashion that enables estimating the action-value for novel object-interactions composed of objects and interactions experienced during training.

We show evidence that this parametrization facilitates exploiting the compositional nature of text instructions by showing systematic generalization to both unseek task descriptions and unseen tasks. We present an example in Fig. 1, where the agent is able to systematically generalize the behavior "pickup up the knife" to "move the knife" and "cut the yellow apple" to "slice the apple". Thanks to the additive inductive bias afforded by our action-value parametrization, it is able to compose these behaviors to perform the novel task "move the knife near the table and slice the apple" at test time.

2 Related work

Compositional generalization Prior work has studied compositional generalization in sequence mapping tasks. Benchmarks such as SCAN (Lake and Baroni, 2018) and gSCAN (Ruis et al., 2020) study translating synthetic text descriptions to an action sequence (e.g. jump twice $\rightarrow$ JUMP JUMP). gSCAN couples SCAN instances with entities in a grid environment and solving a task requires grounding the text and entities similar to our work. Prior approaches for these benchmarks impose compositional inductive biases in models by augmenting models with memory (Lake, 2019) and data augmentation (Andreas, 2019). In this work we use attention mechanisms and introduce a novel poilcy parameterization to impose compositional inductive biases.

Text based embodied control Advances in photo-realistic simulation environments such as DeepMind Lab (Beattie et al., 2016) and AI2Thor (Kolve et al., 2017) have driven recent progress in embodied agents that learn from text instructions. Chaplot et al. (2018) consider a simple navigation task where an agent has to move to an object specified by a set of attributes such as shape and color. They propose the gated attention model to generalize compositionally in the attribute space. Hill et al. (2019) consider systematic generalization in 2D and 3D environments with synthetic text instructions. Compared to these work, we consider object interaction tasks in a photo realistic simulated environment with human-authored language instructions.

ALFRED (Shridhar et al., 2020) couples tasks in the AI2Thor environment with detailed text descriptions of tasks. In contrast, we consider a simplified setup of learning compositional skills from

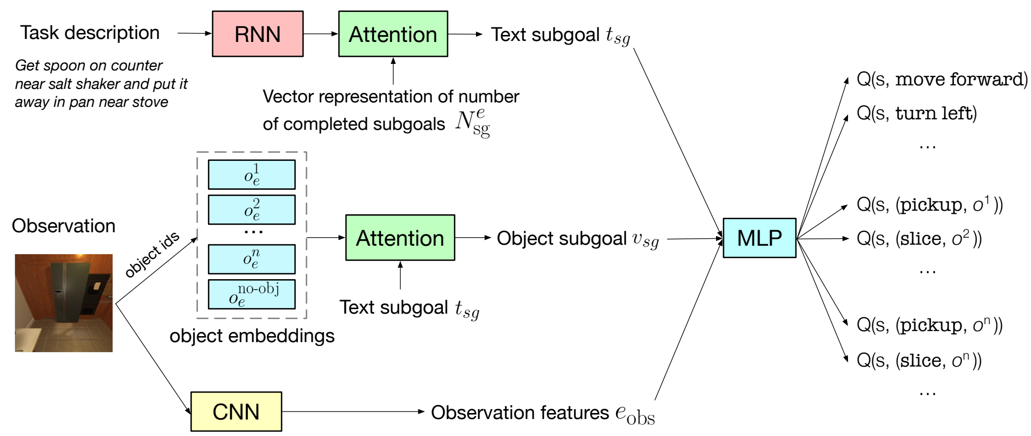

Figure 2: Approach Overview: We perform attention over the text instruction to construct an embedding t_{sg} that represents the current subgoal. The text embedding subgoal t_{sg} attends to scene object embeddings to construct an object subgoal representation v_{sg}. An MLP takes t_{sg}, v_{sg} and observation features e_{obs} as input and predicts state-action values $Q(s, a)$. The entire model is trained end-to-end using Q-learning. See text for details.

high-level task descriptions. We further do not assume access to expert task demonstrations. These assumptions allows us to focus on compositional generalization to zero-shot tasks, which is not the main goal of the ALFRED benchmark. However, the approach presented here can potentially be applicable to ALFRED when combined with learning from demonstrations.

Hierarchical Reinforcement Learning Learning to directly map percepts to low-level action sequences can be challenging. An alternative hierarchical approach is to first come up with a sequence of subtasks, which can be considered as high-level actions (Andreas et al., 2017; Zhu et al., 2017). Each of those subtasks can then be realized using low-level actions. Our policy has an implicit hierarchical structure where latent subgoals are represented as text embeddings using attention. Language was used as an abstraction for the high-level policy in Jiang et al. (2019a) for object rearrangement tasks based on the CLEVR engine (Johnson et al., 2017).

Finally, generalization to unseen instructions has been considered in prior work such as Oh et al. (2017); Lynch and Sermanet (2020), although compositional generalization is not their main focus.

3 Problem

We consider an embodied agent acting in a kitchen environment to solve basic tasks from language instructions (See Fig. 4 for an example task). At the beginning of an episode the agent receives a text instruction τ. Our goal is to learn a policy $\pi(a|s, \tau); a \in \mathcal{A}, s \in \mathcal{S}$ that predicts actions in order to complete tasks. The agent state s is partially observable – it receives an egocentric observation *obs* of the scene. We further assume that an oracle object recognition model provides the object ids for objects in the egocentric observation.

The action space consists of navigation and object interaction actions $\mathcal{A} = \mathcal{A}_{\text{nav}} \cup \mathcal{A}_{\text{int}}$. There are 8 navigation actions $\mathcal{A}_{\text{nav}} = \{$*move forward, move back, move left, move right, turn left, turn right, look up, look down*$\}$. Interaction actions $\mathcal{A}_{\text{int}} = \mathcal{B} \times \mathcal{I}$ are specified using an interaction $b \in \mathcal{B}$ and an object id $o \in \mathcal{I}$ where $\mathcal{B} = \{$*pickup, place, slice, toggle on, toggle off, turn on, turn off*$\}$ and $\mathcal{I}$ is a pre-defined set of identifiers of objects that are available to the agent for interaction in the current observation.

The agent receives a positive reward for successfully completing a task. It also receives a small negative reward for every time-step. In addition, we also assume that every correct object interaction receives a positive reward. In addition to providing a denser learning signal, the rewards are also used to identify subgoals as described in section 4.1. In practice such dense rewards may be unavailable, but this is outside the scope of our study and left as future work.

4 Approach

We approach the problem by considering a task τ to be composed of subgoals $g_1, ..., g_n$, where each subgoal g_i involves navigating to a particular object an interacting with it. For example, the task *place an apple on the table* involves finding the apple and picking it up, followed by navigating to the table and putting down the apple, which can be considered to be the two subgoals for executing the task. Each object interaction required to complete the task thus corresponds to a subgoal. Since every subgoal completion receives a positive reward, the number of subgoals completed at every time-step N_{sg} is known to the agent. The subgoals themselves are not known to the agent – we use attention on the text instruction to compute a latent subgoal representation.

4.1 Text subgoal inference

Given instruction τ composed of the tokens $(w_1, ..., w_n)$, we obtain the corresponding token embeddings $E = (e_1, ..., e_n)$ and use an RNN to encode the instruction to obtain a sequence of contextualized token representations $H = (h_1, ..., h_n)$. We compute a *text subgoal* t_{sg} for a given time-step by computing attention on the instruction using N^e_{sg} as query where N^e_{sg} is a vector representation of N_{sg}. This is shown in Eq. (1) (Q, K, V are learnable parameter matrices).

$$t_{sg} = \text{Attention}(\text{query} = N^e_{sg}, \text{keys} = \text{values} = H) = \sum_{h \in H} \frac{\exp((Qs)^\top (Kh))}{\sum\limits_{h' \in H} \exp((Qs)^\top (Kh'))} Vh \tag{1}$$

We expect the attention to focus on words in the instruction relevant to executing the current subgoal. For instance, if the agent is expected to interact with an apple, the attention module could learn to focus on the word 'apple'.

4.2 Cross-modal reasoning

Given the text subgoal t_{sg}, we use an attention mechanism to reason about objects in the scene within some distance to the agent. This helps the agent understand if objects of interest relevant to the subgoal are present nearby. Let the set of nearby scene objects be $O = \{o^1, ..., o^n\}$, where the $o^i \in \mathcal{I}$ are object ids provided by an oracle. The o^i's can thus be treated as indexes into an embedding table that produces object embeddings $O_e = \{o^1_e, ..., o^n_e\}$. The cross-modal attention is given by Eq. (2) where the text subgoal attends to the scene object embeddings (Q', K', V' are learnable parameter matrices). We augment the scene objects embeddings O_e with an additional learned embedding o^{no-obj}_e which is expected to absorb any probability mass not assigned to scene objects $O'_e = O_e \cup o^{no-obj}_e$. The attention produces an *object subgoal* embedding v_{sg}.

$$v_{sg} = \text{Attention}(\text{query} = t_{sg}, \text{keys} = \text{values} = O'_e) = \sum_{o_e \in O'_e} \frac{\exp((Q' t_{sg})^\top (K' o_e))}{\sum\limits_{o'_e \in O'_e} \exp((Q' t_{sg})^\top (K' o'_e))} V' o_e \tag{2}$$

4.3 Policy learning

We use a deep Q-learning algorithm to train a policy (Mnih et al., 2013), where a neural network is trained to approximate the state-action value function $Q(s, a)$. Given the current observation, text subgoal and object subgoal, the state-action value for a navigation action $a \in \mathcal{A}_{\text{nav}}$ is given by Eq. (3), where f_{nav} is an MLP (multi-layer perceptron) and $e_{\text{obs}} = f_{\text{CNN}}(\text{obs})$ is a feature vector of the observation image computed using a CNN encoder.

$$Q_{\text{nav}}(s, a) = f_{\text{nav}}(a | e_{\text{obs}}, t_{sg}, v_{sg}) \tag{3}$$

The state-action values for interaction actions $a = (b, o) \in \mathcal{B} \times \mathcal{I}$ can be analogously modeled as in Eq. (4). We found it helpful to decompose the state-action value in an additive fashion over an action score f^a_{int} and an object score f^o_{int} as in Eq. (5). Intuitively, f^a_{int} learns to model action preferences, whereas f^o_{int} learns to ground text goals to physical objects. In addition to sharing parameters across actions and objects, this decomposition allows us to model state-action values of object interactions not experienced during training, as long as the specific interaction and the object were encountered.

Task type	Task descriptions
pick up pot	Go to the stove and pick up the pot. Pick up the pot on the bottom right burner on the stove. Take the cooking pot from the stove.
place spoon in pan	get spoon on counter near salt shaker and put it away in pan near stove. Pick up the spoon from the table near the salt shaker and move it to the pan on the counter by the sink. Move spoon from the counter and into the pan.
slice bread with knife	Pick the knife and slice the bread. Take the knife with the yellow handle from the counter by the sink and use it to cut horizontal slices out of the loaf of bread on the white table. Pick up the sharp knife with a yellow handle, and slice the bread on the white table.

Table 1: Example task types and corresponding task descriptions. Note that the task descriptions are used for training and testing agents. The task types are not known to the agents.

Unless specified otherwise we use the decomposed value function $Q^{\text{add}}_{\text{int}}$ in our experiments.

$$Q^{\text{full}}_{\text{int}}(s,a) = f_{\text{int}}(a|e_{\text{obs}}, t_{sg}, v_{sg}) \tag{4}$$

$$Q^{\text{add}}_{\text{int}}(s,a) = f^{a}_{\text{int}}(b|e_{\text{obs}}, t_{sg}, v_{sg}) + f^{o}_{\text{int}}(o|t_{sg}) \text{ where } a=(b,o) \in \mathcal{B} \times \mathcal{I} \tag{5}$$

In summary, the state-action value function is modeled as in Eq. (6).

$$Q(s,a) = \begin{cases} Q_{\text{nav}}(s,a); & a \in \mathcal{A}_{\text{nav}} \\ Q^{\text{add}}_{\text{int}}(s,a); & a \in \mathcal{A}_{\text{int}} \end{cases} \tag{6}$$

The overall model (see Fig. 2 for an illustration) including parameters of the subgoal inference (Eq. 1) and cross-modal reasoning (Eq. 2) components, as well as the MLPs in Eqs. (3) and (5) are trained end-to-end using a double-DQN algorithm (Van Hasselt et al., 2016). Once the model has been trained we construct a greedy policy by choosing actions with the highest state-action values for inference.

5 Experiments

5.1 Tasks

We use the AI2Thor (Kolve et al., 2017) environment as a testbed for our experiments. While there exist prior benchmarks that couple language instructions with embodied environments such as ALFRED Shridhar et al. (2020), they were not designed to study compositional generalization. We thus construct a new task setup that allows us to flexibly vary tasks and object arguments. We consider the following task types in our experiments,

- *pickup x*: Find and pick up object x
- *place x in y*: Find and pick up object x, followed by navigating towards y and placing it.
- *slice x with y*: Secure cutting instrument y, find object x and perform the slice action on it.

We use Amazon Mechanical Turk to collect natural language descriptions of tasks for training and evaluation. A turker is shown key observation frames during the execution of a particular task and is asked to describe in a sentence how they would describe the task to a robot. Turkers were instructed to do their best to correctly identify task relevant objects. But often descriptions from the turkers incorrectly identify objects such as identifying a potato as an avocado. Such descriptions were manually fixed so that the correct object identities are mentioned in the instructions. We collected 5 natural language descriptions each for 35 tasks that include *pickup*, *place* and *slice* tasks. The descriptions consist of 170 unique tokens and have an average length of 12 tokens. Table 1 shows example descriptions collected for some tasks. See appendix B for instructions given to Turkers in the data collection process.

The pickup tasks are used for evaluating multi-task and zero-shot generalization with seen and unseen descriptions of tasks. We use 10 *pickup* tasks - *pickup X* where $X \in \{$*apple, bread, tomato, potato,*

	place tasks	**slice tasks**
Training tasks	*place apple in plate*	*slice apple with knife*
	place butterknife in plate	*slice tomato with knife*
	place spoon in plate	*slice bread with knife*
	place butterknife in pan	*slice apple with butterknife*
	place potato in pan	*slice potato with butterknife*
	place spoon in pan	*slice bread with butterknife*
	place apple in pot	
	place butterknife in pot	
	place potato in pot	
Test tasks (obj-obj setting)	*place potato in plate*	*slice potato with knife*
	place apple in pan	*slice tomato with butterknife*
	place spoon in pot	
Test tasks (task-obj setting)	*place knife in plate*	*slice lettuce with knife*
	place knife in pan	*slice lettuce with butterknife*
	place knife in pot	

Table 2: Task types used for training and testing on place and slice tasks. The *obj-obj* setting considers test tasks composed of unseen combinations of objects. The *task-obj* setting considers generalization to unseen combinations of tasks and objects (e.g. learning to slice lettuce when taught how to slice objects and how to pickup lettuce).

lettuce, spoon, bread, butter knife, plate, pot}. These tasks are used for evaluating generalization to seen and unseen descriptions of known short-horizon tasks. They are also used in generalization to longer horizon tasks as described later in this section.

The *place* and *slice* tasks are used for evaluating generalization to longer-horizon unseen tasks. Table 2 shows tasks used for training and evaluation. In addition to multitask generalization, we use these tasks to study zero-shot compositional generalization to unseen task descriptions. The unseen descriptions can correspond to tasks that were encountered during training, similar to the *pickup* tasks. A more challenging generalization scenario is to generalize to text descriptions of unseen tasks.

We consider two types of tasks in the latter scenario. The *obj-obj setting* examines the ability of the agent to generalize to tasks composed of unseen combinations of objects. For instance, in the test task *place* ***potato*** *in* ***plate***, the relevant objects ***potato***, ***plate*** were encountered during training in tasks such as *place* ***potato*** *in pan* and *place apple in* ***plate***.

The *task-obj setting* is a harder generalization problem where the agent is expected to generalize to unseen combinations of tasks and objects. For the test task *slice* ***lettuce*** *with* ***knife***, the object ***lettuce*** was never observed in the context of a *slice* task during training. However, the agent has access to *pickup* tasks and is expected to learn to interact with lettuce by using the *pickup* ***lettuce*** task. This can be challenging because the agent was only taught how to pick up lettuce, and did not learn to associate lettuce with slice tasks.

The training tasks in Table 2 were designed such that each object argument appears in multiple tasks. Furthermore, when choosing object arguments for a given task type, we prioritized objects that appear in as many tasks as possible. For instance, in the pickup and place tasks setup, the objects were plate, pan, pot, spoon, etc. where each object appears in at least three of the training tasks. This ensures that there are enough occurrences of each object type for the agent to understand and ground the object type. It also helps the agent disentangle the notion of an object versus a task in a given instruction.

5.2 Baselines

We compare the proposed approach against the following baselines.

RNN In this baseline we replace the attentional model with an RNN that produces an embedding of the text instruction. While this model can potentially work for unseen instructions, we examine if the encoding effectively captures the compositional information present in the instructions.

Gated Attention This architecture (Chaplot et al., 2018) combines the instruction representation with the visual observation using a gated attention operation. The fused representation is fed to an MLP which models the state-action values. All models and baselines are trained using the DDQN Q-learning algorithm.

Tasks		Training tasks		Test tasks	
Descriptions		seen	unseen	unseen	unseen
				obj-obj	task-obj
Model	RNN	0.65	0.65	0.26	0.13
	Gated Attention	0.92	0.85	0.66	0.34
	Ours				
	(a) $Q_{\text{int}}^{\text{add}}$ (no cross modal)	0.89	0.76	0.84	0.77
	(b) $Q_{\text{int}}^{\text{full}}$ + cross modal	0.93	0.85	0.44	0.34
	(c) $Q_{\text{int}}^{\text{add}}$ + cross modal	**0.95**	**0.87**	**0.94**	**0.91**

Table 3: Task success rates of models under different generalization settings. Models are evaluated on seen/unseen descriptions of seen tasks and on unseen descriptions of unseen tasks. For unseen tasks, we further evaluate under unseen combinations of objects as well as unseen combinations of tasks and objects. Best numbers are boldfaced.

5.3 Hyperparameters

Word embeddings and the RNN have representation size 32. Objects are represented by embeddings of size 32 from an embedding table. The CNN observation features have size 512 and the CNN encoder has 1.7M parameters, which constitues 90% of the overall model parameters. The MLPs in Eqs. (3) and (4) are single hidden layer MLPs with 256 hidden units and ReLU activation.

5.4 Results

Short-horizon tasks We first consider pickup tasks that involve a single object interaction. In these tasks the agent has to identify the object reference mentioned in the text description and then find and pick up the relevant object. We train and evaluate on 10 pickup task types. Four text descriptions of each task type are part of the training set and the remaining descriptions (i.e., 1 per task type) are part of the test set. Identifying the correct subgoal for these tasks involves paying attention to the verbs and nouns in the task description as well as the overall context. On the training and test descriptions, our agent trained from scratch achieves success rates of $0.9, 0.92$ respectively.

Longer-horizon Tasks We now consider tasks that involve two subgoals, which includes the place and slice tasks in Table 2. Jointly learning text grounding and subgoal inference for long horizon tasks can be challenging. We thus consider a curriculum learning strategy where an agent is gradually trained on tasks of increasingly longer horizon. The agent is first pre-trained on the pickup tasks as described in the previous section, and then fine-tuned on the training tasks in Table 2. Fig. 3 compares the learning progress of agents trained from scratch and an agent that has been pre-trained on the pickup tasks. The pre-trained agent learns twice as fast compared to the agent trained from scratch and achieves perfect success rate on training tasks.

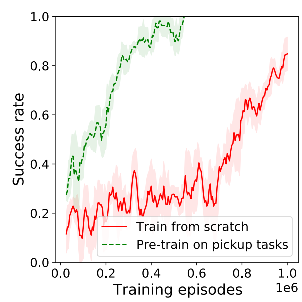

Figure 3: Learning progress of agent trained from scratch and agent pre-trained on pickup tasks.

Generalization Table 3 shows the average task completion success rate of models under different generalization scenarios. The RNN and Gated Attention baselines are limited by the fact that the text instruction is represented using the same encoding across all time-steps, which has limited ability to capture compositional information. The inductive bias of Gated Attention enables better performance, but it has difficulty generalizing to unseen tasks. The attention based model outperforms these baselines, which indicates that the attention mechanism helps exploit compositional information in the instruction better than a fixed encoding.

In addition to better performance, the attention model has the advantage of being more interpretable. Fig. 4 shows the agent's actions and the attention pattern over time for an example task. The agent learns to identify object references in the instruction and uses attention as a sub-goal representation. This mimics a hierarchical policy where a high-level controller provides a sub-goal and a low-level controller executes it Jiang et al. (2019b). The agent further learns to ground object references in the

Instruction	Subgoal 1	bring the potato from the table to the plate on the right of the oven .
attention	Subgoal 2	bring the potato from the table to the plate on the right of the oven .

Instruction	Subgoal 1	pick up the butter knife on the counter , and horizontally slice the lettuce .
attention	Subgoal 2	pick up the butter knife on the counter , and horizontally slice the lettuce .

Figure 4: Agent's observation at different time-steps while performing a place task and a slice task. The attention distribution in the text goal inference component while executing each subgoal is also given below the agent observations.

text instruction to objects in the scene. Notably, these attention patterns and grounding are learned from the reward signal alone without any other supervision. More example of agent trajectories are given in appendix A.

5.5 Ablations

We perform ablations to study the impact of cross-modal reasoning and decomposing the value function in an additive fashion.

Cross modal reasoning We examine model performance without the cross modal reasoning component. In this case the MLPs in Eqs. (3) and (5) only receive the text subgoal and observation encoding as inputs and the visual subgoal v_{sg} is omitted. From rows (a) and (c) in table Table 3 it is clear that the cross-modal reasoning components helps ground text in scene objects and enables better generalization across all settings.

Interaction Q-values We examine the benefit of decomposing the value function approximation of interaction actions in an additive fashion in $Q^{\text{add}}_{\text{int}}$ (Eq. (5)). We compare it against $Q^{\text{full}}_{\text{int}}$ (Eq. (4)), which treats each (interaction, object) pair as a separate atomic action. Comparing rows (b), (c) in Table 3 we see that the additive decomposition is crucial for generalization to unseen tasks.

6 Conclusion

In this work we proposed attention based agents that can exploit the compositional nature of language instructions to generalize to unseen tasks. The policy mimics a hierarchical process where a text embedding obtained via attention represents the subgoal to be executed and the policy network executes the low level actions. The proposed method performs strongly against baselines on a testbed we created based on a photorealistic simulated environment and provides some interpretability.

Compared to existing benchmarks such as ALFRED we made simplifying assumptions such as oracle visual recognition, relatively short horizon tasks and generalization within single kitchen layout which allows us to focus on compositional generalization in embodied settings. However, the ideas presented here can potentially be combined with curriculum learning and learning from human demonstrations to perform complex tasks that require planning over hundreds of time-steps such as in the ALFRED setting, and we leave this to future work.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Jacob Andreas. 2019. Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545*.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. http://arxiv.org/abs/1611.01796 Modular Multi-task Reinforcement Learning with Policy Sketches. *arXiv:1611.01796 [cs]*. ArXiv: 1611.01796.

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

Wilka Carvalho, Anthony Liang, Kimin Lee, Sungryull Sohn, Honglak Lee, Richard L Lewis, and Satinder Singh. 2020. Reinforcement learning for sparse-reward object-interaction tasks in first-person simulated 3d environments. *arXiv preprint arXiv:2010.15195*.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2018. Gated-attention architectures for task-oriented language grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Noam Chomsky. 2009. *Syntactic structures*. De Gruyter Mouton.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. 2017. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. 2019. Environmental drivers of systematicity and generalization in a situated agent. *arXiv preprint arXiv:1910.00571*.

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. 2019a. Language as an abstraction for hierarchical deep reinforcement learning. *arXiv preprint arXiv:1906.07343*.

YiDing Jiang, Shixiang (Shane) Gu, Kevin P Murphy, and Chelsea Finn. 2019b. http://papers.nips.cc/paper/9139-language-as-an-abstraction-for-hierarchical-deep-reinforcement-learning.pdf Language as an Abstraction for Hierarchical Deep Reinforcement Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle AlchÃ©-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9419–9431. Curran Associates, Inc.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. *arXiv preprint arXiv:1906.05381*.

Corey Lynch and Pierre Sermanet. 2020. http://arxiv.org/abs/2005.07648 Grounding Language in Play. *arXiv:2005.07648 [cs]*. ArXiv: 2005.07648.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. http://arxiv.org/abs/1706.05064 Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. *arXiv:1706.05064 [cs]*. ArXiv: 1706.05064.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. 2020. http://arxiv.org/abs/2003.05161 A Benchmark for Systematic Generalization in Grounded Language Understanding. *arXiv:2003.05161 [cs]*. ArXiv: 2003.05161.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. http://arxiv.org/abs/1912.01734 ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. *arXiv:1912.01734 [cs]*. ArXiv: 1912.01734.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. 2017. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision*, pages 483–492.