ALFRED

A Benchmark for Interpreting Grounded Instructions for Everyday Tasks

Mohit Shridhar[1] Jesse Thomason[1] Daniel Gordon[1] Yonatan Bisk[1,2,3]
Winson Han[3] Roozbeh Mottaghi[3] Luke Zettlemoyer[1] Dieter Fox[1,4]

AskForALFRED.com

Abstract

We present **ALFRED** *(***A**ction **L**earning **F**rom **R**ealistic **E**nvironments and **D**irectives*), a benchmark for learning a mapping from natural language instructions and egocentric vision to sequences of actions for household tasks. Long composition rollouts with non-reversible state changes are among the phenomena we include to shrink the gap between research benchmarks and real-world applications.* ALFRED *consists of expert demonstrations in interactive visual environments for 25k natural language directives. These directives contain both high-level goals like "Rinse off a mug and place it in the coffee maker." and low-level language instructions like "Walk to the coffee maker on the right."* ALFRED *tasks are more complex in terms of sequence length, action space, and language than existing vision-and-language task datasets. We show that a baseline model designed for recent embodied vision-and-language tasks performs poorly on* ALFRED*, suggesting that there is significant room for developing innovative grounded visual language understanding models with this benchmark.*

Figure 1: ALFRED consists of 25k+ language directives corresponding to expert demonstrations on household tasks. Above, we highlight several action sequence frames corresponding to portions of the accompanying language instruction. Unlike related datasets that focus only on navigation, ALFRED requires interactions with objects, keeping track of state changes, and callbacks to previous instructions.

1. Introduction

A robot operating in a human spaces needs to connect natural language to the world. This symbol grounding [21] problem has largely focused on connecting language to static images. However, robots need to understand task-oriented language, for example "Rinse off a mug and place it in the coffee maker" illustrated in Figure 1.

Platforms for translating language to action have become increasingly popular, spawning new test-beds [12, 3, 14, 41]. These benchmarks include language-driven navigation and embodied question answering, which have seen dramatic improvements in modeling thanks to environments like Matterport 3D [11, 3], AI2-THOR [25], and AI Habitat [43]. However, these datasets ignore complexities arising from describing task-oriented behaviors with objects.

We introduce **ALFRED**,a new benchmark for connecting human language to actions, behaviors, and objects in an interactive visual environment. Expert task demonstrations are accompanied by both high- and low-level human language instructions in 120 indoor scenes in the new AI2-THOR 2.0 [25]. These demonstrations involve partial observability, long action horizons, underspecified natural language, and irreversible actions.

ALFRED includes 25,743 English language directives describing 8,055 expert demonstrations averaging 50 steps each, resulting in 428,322 image-action pairs. Motivated by work in robotics on segmentation-based grasping [36], agents in ALFRED interact with objects visually, specifying a pixelwise interaction mask of the target object. This

[1]Paul G. Allen School of Computer Sci & Eng, Univ of Washington
[2]Carnegie Mellon University LTI & Microsoft Research AI
[3]Allen Institute for AI [4]NVIDIA

	— Language —		— Virtual Environment —			— Inference —		
	# Human Annotations	Granularity	Visual Quality	Movable Objects	State Changes	Vis. Obs.	Navigation	Interaction
TACoS [42]	**17k+**	**High&Low**	**Photos**	✗	✗	–	–	–
R2R [3]; Touchdown [14]	**21k+; 9.3k+**	**Low**	**Photos**	✗	✗	**Ego**	**Graph**	✗
EQA [15]	✗	**High**	**Low**	✗	✗	**Ego**	**Discrete**	✗
Matterport EQA [53]	✗	**High**	**Photos**	✗	✗	**Ego**	**Discrete**	✗
IQA [20]	✗	**High**	**High**	✗	✓	**Ego**	**Discrete**	**Discrete**
VirtualHome [41]	**2.7k+**	**High&Low**	**High**	✓	✓	**3rd Person**	✗	**Discrete**
VSP [56]	✗	**High**	**High**	✓	✓	**Ego**	✗	**Discrete**
ALFRED	**25k+**	**High&Low**	**High**	✓	✓	**Ego**	**Discrete**	**Discrete + Mask**

Table 1: **Dataset comparison.** ALFRED is the first interactive visual dataset to include high- and low-level natural language instructions for object and environment interactions. TACoS [42] provides detailed high- and low-level text descriptions of cooking videos, but does not facilitate task execution. For navigation, ALFRED enables discretized, grid-based movement, while other datasets use topological graph navigation or avoid navigation altogether. ALFRED requires an agent to generate spatially located interaction masks for action commands. By contrast, other datasets only require choosing from a discrete set of available interactions and object classes or offer no interactive capability.

inference is more realistic than simple object class prediction, where localization is treated as a solved problem. Existing beam-search [17, 51, 46] and backtracking solutions [23, 28] are infeasible due to the larger action and state space, long horizon, and inability to undo certain actions.

To establish baseline performance levels, we evaluate a sequence-to-sequence model shown to be successful on vision-and-language navigation tasks [27]. This model is not effective on the complex tasks in ALFRED, achieving less than 5% success rates. For analysis, we also evaluate individual sub-goals like the routine of cooking something in a microwave. While performance is better for isolated sub-goals, the model lacks the reasoning capacity for long-horizon and compositional task planning.

In summary, ALFRED facilitates learning models that translate from language to sequences of actions and predictions of visual interaction masks for object interactions. This benchmark captures many reasoning challenges present in real-world settings for translating human language to robot actions for accomplishing household tasks. Models that can overcome these challenges will begin to close the gap towards real-world, language-driven robotics.

2. Related Work

Table 1 summarizes the benefits of ALFRED relative to other visual action datasets with language annotations.

Vision & Language Navigation. In vision-and-language navigation tasks, either natural or templated language describes a route to a goal location through egocentric visual observations [30, 13, 12, 3, 14]. Since the proposal of R2R [3], researchers have dramatically improved the navigation performance of models [52, 17, 51, 23, 28] with techniques like progress monitoring [27], as well as introduced task variants with additional, on-route instructions [38, 37, 49]. Much of this research is limited to static environments. By contrast, ALFRED tasks include navigation, object interactions, and state changes.

Vision & Language Task Completion. There are several existing benchmarks based on simple block worlds and fully observable scenes [9, 33]. ALFRED provides more difficult tasks in richer, visually complex scenes, and uses partially observable environments. The CHAI benchmark [32] evaluates agents performing household instructions, but includes only a single interact action outside navigation. ALFRED has seven manipulation actions, such as pick up, turn on, and open, state changes like clean versus dirty, and variation in language and visual complexity.

Previous work using the original AI2-THOR environment also investigated the task of visual semantic planning [56, 19]. Artificial language in those datasets comes from templates, and environment interaction is handled with discrete class predictions, for example selecting *apple* as the target object from predefined options. ALFRED features human language instructions, and object selections are carried out with class-agnostic, pixelwise interaction masks. In VirtualHome [41], demonstration programs are generated from video demonstration and natural language instructions, but inference does not involve egocentric visual and action feedback or partial observability.

There is extensive literature on language-based instruction following in the natural language processing community. There, research has focused on mapping instructions to actions [13, 47, 5, 31, 35], but these works do not scope visual, interactive environments.

Embodied Question Answering. Existing datasets for

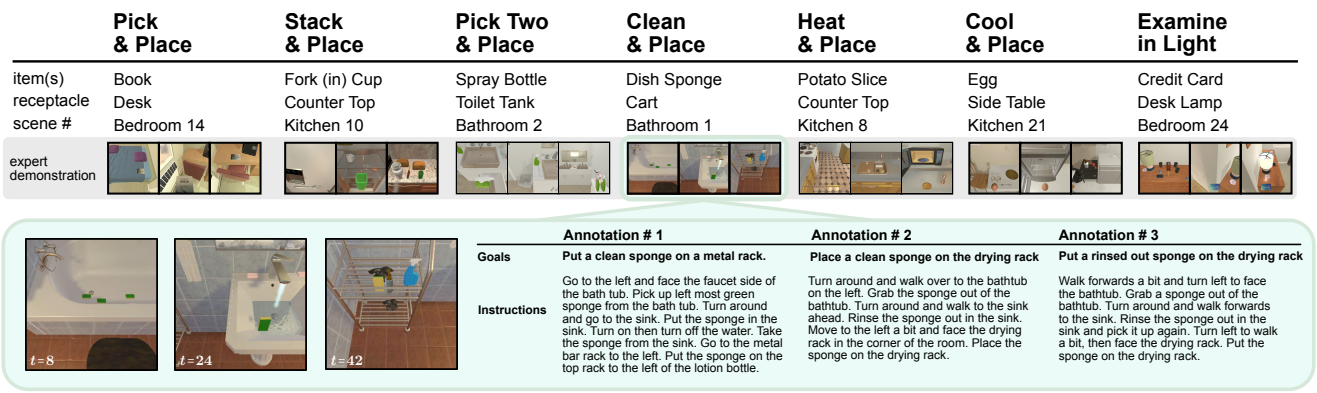


Figure 2: **ALFRED annotations.** We introduce seven different task types with many combinations of objects in 120 scenes. An example of each task type is given above. For the **Clean & Place** demonstration, we also show the three crowdsourced language directives. Please see the supplemental material for example demonstrations and language for each task.

visual question answering in embodied environments use templated language or static scenes [20, 15, 55, 53]. In ALFRED, rather than answering a question, the agent must complete a task specified using natural language, which requires both navigation and interaction with objects.

Instruction Alignment. There has also been work on aligning natural language with video clips to find visual correspondences between words and concepts [42, 54, 44, 1, 57]. ALFRED requires performing tasks in an interactive setting as opposed to learning from recorded videos.

Robotics Instruction Following. Instruction following is a long-standing topic of interest in robotics [7, 10, 34, 50, 29, 40, 39, 45]. Lines of research consider different tasks such as cooking [10], table clearing [39], and mobile manipulation [29]. In general, they are limited to a few scenes [34], consider a small number of objects [29], or use the same environment for training and testing [7]. In contrast, ALFRED includes 120 indoor scenes, many object classes with diverse appearance across scenes and states, and a test set of unseen environments.

3. The ALFRED Dataset

The ALFRED dataset comprises 25,743 language directives corresponding to 8,055 expert demonstration episodes. Each directive includes a high-level goal and a set of step-by-step instructions. Each expert demonstration can be deterministically replayed in the AI2-THOR 2.0 simulator.

3.1. Expert Demonstrations

Expert demonstrations are composed of an agent's egocentric visual observations of the environment and what action is taken at each timestep as well as ground-truth interaction masks. Navigation actions move the agent or change its camera orientation, while manipulation actions include picking and placing objects, opening and closing cabinets and drawers, and turning appliances on and off. Interactions can involve multiple objects, such as using a knife to slice an apple, cleaning a mug in the sink, and browning a potato in the microwave. Manipulation actions are accompanied by a ground truth segmentation of the target object. At inference time, interaction masks must be generated along with an action to indicate an object for interaction.

Figure 2 gives examples of the high-level agent tasks in ALFRED, like putting a cleaned object at a destination. These tasks are parameterized by the object of focus, the destination receptacle (*e.g.*, *table top*), the scene in which to carry out the task, and a base object for a stack (for **Stack & Place**). ALFRED contains expert demonstration of these seven tasks executed using combinations of 58 unique object classes and 26 receptacle object classes across 120 different indoor scenes. For objects classes like *potato slice*, the agent must first pick up a *knife* and find a *potato* to create slices. All object classes contain multiple variations with different shapes, textures, and colors. For example, there are 30 unique variants of the *apple* class. Indoor scenes include different room types: 30 each of kitchens, bathrooms, bedrooms, and living rooms.

For 2,685 combinations of task parameters, we gather three expert demonstrations per parameter set, for a total of 8,055 unique demonstrations with an average of 50 action steps. The distributions of actions steps in ALFRED demonstrations versus related datasets is given in Figure 3. As an example, for task parameters {**Heat & Place**, *potato*, *counter top*, KITCHEN-8}, we generate three different expert demonstrations by starting the agent and objects in randomly chosen locations. Object start positions have some commonsense class-specific constraints, for example a *fork* can start inside a *drawer*, but an *apple* cannot.

Contrasting navigation-only datasets where expert

	Train	Validation		Test	
		Seen	*Unseen*	*Seen*	*Unseen*
# Annotations	21,023	820	821	1,533	1,529
# Scenes	108	88	4	107	8

Table 2: **ALFRED Data Splits.** All expert demonstrations and associated language directives in the validation and test folds are distinct from those in the train fold. The validation and test sets are split into *seen* and *unseen* folds. Scenes in the *seen* folds of validation and test data are subsets of those in the train fold. Scenes in the *unseen* validation and test folds are distinct from the train folds and from each other.

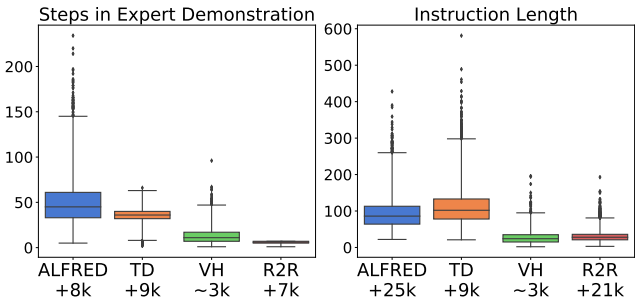


Figure 3: **Comparison to Existing Datasets.** Expert demonstration steps and instruction tokens of ALFRED compared to other datasets with human language for action sequences: Touchdown (TD) [14], VirtualHome (VH) [41], and Room-to-Room (R2R) [3]. The total number of demonstrations or annotations is given with the dataset label.

demonstrations can come from an A^* planner, our state space includes object positions and state changes. Thus, to generate expert demonstrations we encode the agent and object states, as well as high-level environment dynamics, into Planning Domain Definition Language (PDDL) rules [18]. We then define task-specific PDDL goal conditions, for example that a heated *potato* is resting on a *table top*. Note that the planner encodes the environment as fully observable and has perfect knowledge about world dynamics. For training and testing agent models, however, the environment is partially observable: it is only viewed through the agent's egocentric vision as actions are carried out.

We split these expert demonstrations into training, validation, and test folds (Table 2). Following work in vision-and-language navigation [3], we further split the validation and test into two conditions: *seen* and *unseen* environments. This split facilitates examining how well models generalize to entirely new spaces with novel object class variations.

3.2. Language Directives

For every expert demonstration, we collect open vocabulary, free-form language directives from at least three different annotators using Amazon Mechanical Turk (AMT), resulting in 25k total language directives. Language directives include a high-level goal together with low-level instructions, as shown in Figures 1 and 2. The distribution of language annotation token lengths in ALFRED versus related datasets is given in Figure 3.

AMT workers are told to write instructions to tell a "smart robot" how to accomplish what is shown in a video. We create a video of each expert demonstration and segment it such that each segment corresponds to an instruction. We consult the PDDL plan for the expert demonstration to identify task sub-goals, for example the many low-level steps to navigate to a knife, or the several steps to heat a potato slice in the microwave once standing in front of it. We visually highlight action sequences related to sub-goals via colored timeline bars below the video. In each HIT (Human Intelligence Task), a worker watches the video, then writes low-level, step-by-step instructions for each highlighted sub-goal segment. The worker also writes a high-level goal that summarizes what the robot should accomplish during the expert demonstration.

These directives are validated through a second HIT by at least two annotators, with a possible third tie-breaker. For validation, we show a worker all three language directive annotations without the video. The worker selects whether the three directives describe the same actions, and if not, which is most different. If a directive is chosen as most different by a majority of validation workers, it is removed and the demonstration is subsequently re-annotated by another worker. Qualitatively, these rejected annotations contain incorrect object referents (*e.g.*, "egg" instead of "potato") or directions (*e.g.*, "go left towards..." instead of "right").

4. Baseline Models

An agent trained for ALFRED tasks needs to jointly reason over vision and language input and produce a sequence of low-level actions to interact with the environment.

4.1. Sequence-to-Sequence Models

We model the interactive agent with a CNN-LSTM sequence-to-sequence (SEQ2SEQ) architecture. A CNN encodes the visual input, a bidirectional-LSTM generates a representation of the language input, and a decoder LSTM infers a sequence of low-level actions while attending over the encoded language. See Figure 4 for an overview and the supplementary material for implementation details.

Supervision. We train all models using the teacher-forcing paradigm on the expert trajectories, and this ensures the language directives match the visual inputs. At each timestep, the model is trained to produce the expert action and associated interaction mask for manipulation actions.

We note that student-forcing in ALFRED is non-trivial,

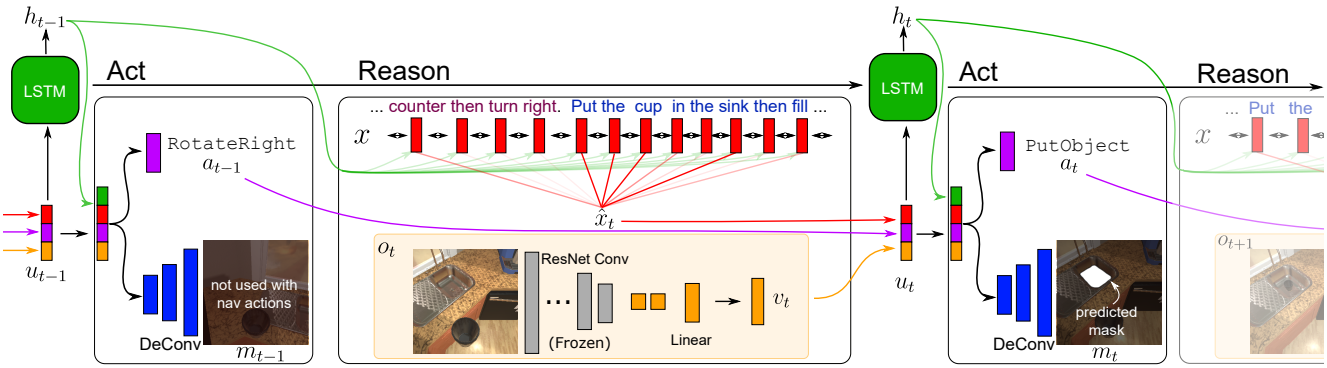


Figure 4: **Model overview.** At each step, our model reweights the instruction based on the history ($\hat{x}_t$), and combines the current observation features (v_t) and the previously executed action (a_{t-1}). These are passed as input to an LSTM cell to produce the current hidden state. Finally, the new hidden state (h_t) is combined with the previous features to predict both the next action (a_t) and a pixelwise interaction mask over the observed image to indicate an object.

even disregarding language alignment. Obtaining expert demonstration actions on the fly in navigation-only datasets like R2R [3] involves precomputing all optimal navigation paths. in ALFRED, obtaining these on the fly demonstrations requires re-planning, and in some cases is not possible at all. For example, if during a task of {**Clean & Place**, *apple*, *refrigerator*, KITCHEN-3} a sample-forcing model slices the only *apple* in the scene, the action cannot be recovered from and the task cannot be completed.

Visual encoding. Each visual observation o_t is encoded with a frozen ResNet-18 [22] CNN, where we take the output of the final convolution layer to preserve spatial information necessary for grounding specific objects in the visual frame. We embed this output using two more 1×1 convolution layers and a fully-connected layer. During training, a set of T observations from the expert demonstration is encoded as $\overline{V} = \langle v_1, v_2, \ldots, v_T \rangle$, where v_t is the visual feature vector at time-step t.

Language encoding. Given a natural language goal $\overline{G} = \langle g_1, g_2, \ldots g_{L_g} \rangle$ of L_g words, and step-by-step instructions $\overline{S} = \langle s_1, s_2 \ldots s_{L_s} \rangle$ of L_s words, we append them into a single input sequence $\overline{X} = \langle g_1, g_2, \ldots g_{L_g}, \texttt{<SEP>}, s_1, s_2 \ldots s_{L_s} \rangle$ with the `<SEP>` token indicating the separation between the high-level goal and low-level instructions. This sequence is fed into a bi-directional LSTM encoder to produce an encoding $x = \{x_1, x_2, \ldots, x_{L_x}\}$ for each word in $\overline{X}$.

Attention over language. The agent's action at each timestep is based on an attention mechanism that identifies relevant tokens in the instruction. We perform soft-attention on the language features x to compute the attention distribution α_t conditioned on the hidden state of the decoder h_{t-1} from the last timestep:

$$\begin{aligned} z_t &= (W_x h_{t-1})^\top x, \\ \alpha_t &= \text{Softmax}(z_t), \\ \hat{x}_t &= \alpha_t^\top x \end{aligned} \tag{1}$$

where W_x are learnable parameters of a fully-connected layer, z_t is a vector of scalar values that represent the attention mass for each word in x, and $\hat{x}_t$ is the weighted sum of x over the attention distribution α_t induced from z_t.

Action decoding. At each timestep t, upon receiving a new observation image o_t, the LSTM decoder takes in the visual feature v_t, language feature $\hat{x}_t$, and the previous action a_{t-1}, and outputs a new hidden state h_t:

$$\begin{aligned} u_t &= [v_t; \hat{x}_t; a_{t-1}], \\ h_t &= \text{LSTM}\,(u_t, h_{t-1}) \end{aligned} \tag{2}$$

where $[;]$ denotes concatenation. The hidden state h_t is used to obtain the attention weighted language feature $\hat{x}_{t+1}$.

Action and mask prediction. The agent interacts with the environment by choosing an action and producing dense pixelwise binary mask to indicate specific objects in the frame. Although AI2-THOR supports continuous control for agent navigation and object manipulation, we discretize the action space for modeling simplicity. The agent chooses from among 13 actions. There are 5 navigation actions: `MoveAhead`, `RotateRight`, `RotateLeft`, `LookUp`, and `LookDown` together with 7 interaction actions: `Pickup`, `Put`, `Open`, `Close`, `ToggleOn`, `ToggleOff`, and `Slice`. Interaction actions require a pixelwise mask to denote the object of interest. Finally, the agent predicts a `Stop` action to end the episode. We concatenate the hidden state h_t with the input features

u_t and train two separate networks to predict the next action a_t and interaction mask m_t:

$$\begin{aligned} a_t &= \text{argmax}\left(W_a\left[h_t; u_t\right]\right), \\ m_t &= \sigma\left(\textbf{deconv}\left[h_t; u_t\right]\right) \end{aligned} \tag{3}$$

where W_a is a fully-connected layer, **deconv** is a three-layer deconvolution network, and σ is a sigmoid activation function. Action selection is trained using softmax cross entropy with the expert action. The interaction masks are learned end-to-end in a supervised manner based on ground-truth object segmentations using binary cross-entropy loss. The mask loss is weight balanced to account for sparsity in these dense masks in which target objects can take up a small portion of the visual frame.

4.2. Progress Monitoring

ALFRED tasks require reasoning over long sequences of images and instruction words. We propose two auxiliary losses that use additional temporal information to reduce this burden, introducing a sequence-to-sequence model with progress monitoring (SEQ2SEQ+PM).

Ma *et al*. showed that agents benefit from maintaining an internal estimate of their progress towards the goal for navigation tasks [27]. Akin to learning a value function in reinforcement learning, progress monitoring helps to learn the utility of each state in the process of achieving the overall task. Intuitively, this allows our agent to better distinguish between visually similar states such as just before putting an object in the microwave versus just after taking the object out. We introduce a simple module that predicts progress, $p_t \in [0, 1]$, conditioned on the decoder hidden state h_t and the concatenated input u_t:

$$p_t = \sigma\left(W_p\left[h_t; u_t\right]\right) \tag{4}$$

where W_p are learnable parameters of a fully connected layer, and σ is a sigmoid activation function. The supervision for p_t is based on normalized time-step values t/T, where t is the current time-step, and T is the total length of the expert demonstration. We train with an L2 loss.

We also train the agent to predict the number of subgoals completed so far, c_t. These sub-goals represent segments in the demonstration corresponding to sequences of actions like navigation, pickup, and heating as identified in the PDDL plan, discussed in Section 3.2. Each segment has a corresponding language instruction, but the alignment must be learned. This sub-goal prediction encourages the agent to coarsely track its progress through the language directive. This prediction is also conditioned on the decoder hidden state h_t and the concatenated input u_t:

$$c_t = \sigma\left(W_c\left[h_t; u_t\right]\right) \tag{5}$$

where W_c are learnable parameters of a fully connected layer and σ is a sigmoid activation function. We train c_t in a supervised fashion by using the normalized number of subgoals accomplished by the expert at each timestep, c_t/C, as the ground-truth label for a task with C sub-goals. We again train with an L2 loss.

5. Experiments

We evaluate the baseline models in the AI2-THOR simulator. When evaluating on test folds, we run corresponding models with the lowest validation loss. Episodes that exceed 400 steps or cause more than 10 API execution failures are terminated. Execution failures arise from bumping into walls or predicting action interaction masks for incompatible objects, such as attempting to `Pickup` a *counter top*. These limitations encourage efficiency and reliability. We assess the overall and partial success of models' task executions across episodes.

5.1. Evaluation Metrics

ALFRED allows us to evaluate both full task and task goal-condition completion. In navigation-only tasks, one can only measure how far the agent is from the goal. In ALFRED, we can also evaluate whether task goal-conditions have been completed, for example that a *potato* has been sliced. For all of our experiments, we report both Task Success and Goal-Condition Success. Each Goal-Condition relies on multiple instructions, for example navigating to an object and then slicing it.

Task Success. Each expert demonstration is parameterized by a task to be performed, as illustrated in Figure 2. Task Success is defined as 1 if the object positions and state changes correspond correctly to the task goal-conditions at the end of the action sequence, and 0 otherwise. Consider the task: "Put a hot potato slice on the counter". The agent succeeds if, at the end of the episode, any *potato slice* object has changed to the *heated* state and is resting on any *counter top* surface.

Goal-Condition Success. The goal-condition success of a model is the ratio of goal-conditions completed at the end of an episode to those necessary to have finished a task. For example, in the previous **Heat & Place** example, there are four goal-conditions. First, a *potato slice* must be created from a full *potato*. Second, a *potato slice* should become *heated*. Third, a *potato slice* should come to rest on a *counter top*. Fourth, the same *potato slice* that is *heated* should be on the *counter top*. If the agent slices a *potato*, then moves a slice to the counter top without heating it, then the goal-condition success score is $2/4 = 0.5$. On average, tasks in ALFRED have 2.55 goal conditions. The final score is calculated as the average goal-condition success of each episode. Task success is 1 only if goal-condition success is 1.

	Validation				Test			
	Seen		*Unseen*		*Seen*		*Unseen*	
Model	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond
VISION-ONLY	0.1 (0.0)	6.4 (3.8)	0.0 (0.0)	6.5 (4.8)	0.0 (0.0)	4.7 (2.9)	0.2 (0.0)	6.5 (4.0)
LANGUAGE-ONLY	0.0 (0.0)	5.7 (4.7)	0.0 (0.0)	6.9 (6.1)	0.0 (0.0)	3.9 (3.2)	0.2 (0.1)	6.6 (4.6)
GOAL-ONLY	0.1 (0.0)	7.0 (4.6)	0.1 (0.0)	7.0 (4.4)	0.3 (0.2)	5.5 (3.9)	0.2 (0.1)	7.0 (4.5)
INSTRUCTIONS-ONLY	2.2 (1.3)	9.5 (6.2)	0.0 (0.0)	7.0 (5.0)	3.4 (1.7)	9.1 (6.3)	**0.6** (**0.3**)	7.2 (4.7)
SEQ2SEQ	3.0 (1.2)	10.0 (6.0)	0.1 (0.0)	6.9 (4.9)	3.2 (1.5)	8.3 (5.3)	**0.6** (0.2)	7.1 (4.6)
+ PM PROGRESS-ONLY	2.8 (1.4)	9.4 (6.5)	0.0 (0.0)	**7.1** (**5.2**)	3.7 (**2.2**)	8.6 (6.2)	0.4 (0.1)	**7.4** (4.5)
+ PM SUBGOAL-ONLY	2.4 (1.6)	9.6 (6.5)	**0.2** (**0.1**)	6.7 (4.9)	4.1 (1.8)	9.1 (6.1)	**0.6** (0.2)	7.1 (4.5)
SEQ2SEQ+PM (both)	**3.8** (**2.2**)	**10.9** (**6.9**)	0.1 (0.0)	6.9 (4.7)	**4.2** (2.1)	**9.5** (**6.5**)	0.5 (0.1)	**7.4** (**4.8**)

Table 3: **Task and Goal-Condition success percentages.** For each metric, the corresponding path weighted metrics are given in parentheses. The highest values per fold and metric are shown in **blue**.

Path Weighted Metrics. We include a Path Weighted version of both metrics that considers the length of the expert demonstration [2]. Expert demonstrations found via a PDDL solver on global information are not guaranteed to be optimal. However, they avoid exploration, use shortest path navigation, and are generally efficient. The path weighted score p_s for metric s is given as

$$p_s = s \times \frac{L^*}{max(L^*, \hat{L})} \tag{6}$$

where $\hat{L}$ is the number of actions the model took in the episode, and L^* is the number of actions in the expert demonstration. Intuitively, a model receives half-credit for taking twice as long as the expert to accomplish a task.

5.2. Sub-Goal Evaluation

Completing the entire sequence of actions required to finish a task is challenging. In addition to assessing full task success, we study the ability of a model to accomplish the next sub-goal conditioned on the preceding expert sequence. The agent is tested by first forcing it to follow the expert demonstration to maintain a history of states leading up to the sub-goal, then requiring it to complete the sub-goal conditioned on the entire language directive and current visual observation. For the task "Put a hot potato slice on the counter" for example, we can evaluate the sub-goal of navigating to the potato after rolling the expert demonstration forward through picking up a knife. The tasks in ALFRED contain on average 7.5 such sub-goals (results in Table 4).

6. Analysis

Results from our experiments are presented in Table 3. We find that the initial model, without spatial or semantic maps, object segmentations, or explicit object-state tracking, performs poorly on ALFRED's long-horizon tasks with high-dimensional state-spaces. The SEQ2SEQ model achieves 10% goal-condition success rate, showing that the agent does learn to partially complete some tasks. This headroom motivates further research into models that can perform the complex vision-and-language planning introduced by ALFRED. The model performance starkly contrasts other vision-and-language datasets focused on navigation, where sequence-to-sequence models with progress monitoring perform well [27].

6.1. Random Agent

A random agent is commonly employed as a baseline in vision-and-language tasks. In ALFRED, an agent that chooses a uniform random action and generates a uniform random interaction mask at each timestep achieves 0% on all folds, even without an API failure limit.

6.2. Unimodal Ablations

Previous work established that agents without visual inputs, language inputs, or both performed better than random agents and were competitive with initial baselines for several navigation and question answering tasks. These performance gaps were due to structural biases in the datasets or issues with model capacity [48]. We evaluate these ablation baselines to study vision and language bias in ALFRED.

The unimodal ablation performances in Table 3 indicate that both vision and language modalities are necessary to achieve the tasks in ALFRED. The VISION-ONLY model finishes some goal-conditions by interacting with familiar objects seen during training. The LANGUAGE-ONLY model similarly finishes some goal-conditions by following low-level language instructions for navigation and memorizing interaction masks for common objects like *microwaves* that are centered in the visual frame.

6.3. Model Ablations

We ablate the amount of language supervision available to the model, as language directives are given as both a high-

level goal and step-by-step instructions. Providing only high-level, underspecified goal language is insufficient to complete the tasks, but is enough to complete some goal-conditions. Using just low-level, step-by-step instructions performs similarly to using both high- and low-levels. Thus, this simple model does not seem to exploit the goal instruction to plan out sub-goals for step-by-step execution.

The two progress monitoring signals are marginally helpful, increasing the success rate from ∼3% to ∼4%. Progress monitoring lead to more efficient task completion, as indicated by the consistently higher path weighted scores. They may help avoid action repetition and with the prediction of the `Stop` action.

The agent takes more steps than the expert in all cases, as indicated by the lower path weighted scores. Sometimes, this is caused by failing to keep track of state-changes, for example heating up an *egg* in the *microwave* multiple times. Further, the models also do not generalize to *unseen* scenes, due to the overall visual complexity in ALFRED arising from new scenes and novel object class instances.

6.4. Human evaluation

For a random subset of 73 expert demonstrations from the *unseen* test fold, we obtained a human evaluation of the 219 corresponding language directives. Six new annotators viewed the language directives alongside the expert demonstration. The annotators marked "Yes" or "No" to indicate whether they felt they could carry out the demonstration based on the language and egocentric frames. This evaluation corresponds to a model with "perfect" language understanding and human-level understanding of the rendered visual scene. For a conservative evaluation, we asked that annotators mark "No" if any step in the language directive was incorrect or confusing. On average, annotators marked 85% of the directives as sufficient to accurately produce the expert demonstration, indicating that the language directives in ALFRED well-aligned with the demonstrations.

6.5. Sub-Goal Performance

We examine performance of the SEQ2SEQ model on individual sub-goals in ALFRED. For this experiment, we use the expert trajectory to move the agent through the episode up to the sub-task. Then, the agent begins inference based on the language directive and current visual frame.

Table 4 presents path-length weighted success scores for 8 sub-goals. **Goto** and **Pickup** are the lowest performing sub-tasks with the SEQ2SEQ+PM model achieving ∼48% and ∼35%, respectively, even in *seen* environments. Visual semantic navigation is considerably harder in *unseen* environments. Similarly, interaction masks for **Pickup** actions in *unseen* environments are worse due to unfamiliar object-background contexts and new object instances. Simple sub-goals like **Cool**, and **Heat** are achieved at a high success rate of ∼90% because these tasks are mostly object-agnostic. For example, the agent becomes familiar with using microwaves to heat things regardless of the object in-hand, because microwaves have little visual diversity across kitchens. Overall, the sub-goal evaluations indicate that models that exploit modularity and hierarchy, or make use of pretrained object segmentation models, may make headway on full task sequences.

Sub-Goal Ablations - **Validation**

	VISION-ONLY		SEQ2SEQ		SEQ2SEQ+PM	
Sub-Goal	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*
Goto	25.3	15.8	46.8	18.4	**47.8**	**20.3**
Pickup	25.4	13.2	34.7	22.6	**35.0**	**23.6**
Put	69.4	31.9	78.0	**53.0**	**79.8**	43.4
Cool	**89.6**	74.9	87.9	94.3	87.1	**95.2**
Heat	**87.6**	88.6	85.5	87.7	84.9	**89.6**
Clean	65.2	16.8	82.0	29.9	**83.8**	**56.0**
Slice	22.6	14.1	27.3	18.8	**31.6**	**25.6**
Toggle	90.4	5.0	96.8	53.7	**100.0**	**57.9**
Average	59.4	32.5	67.3	47.3	**68.7**	**51.4**

Table 4: **Evaluations by path weighted sub-goal success.** The highest values per fold and task are shown in **blue**. We note that the LANGUAGE-ONLY model achieves less than 2% on all sub-goals. See supplemental material for more.

7. Conclusions

We introduced ALFRED, a benchmark for learning to map natural language instructions and egocentric vision to sequences of actions. ALFRED poses a challenging modeling problem, towards the long-term vision of language-driven robots capable of navigation and interaction. The environment dynamics and interaction mask predictions required in ALFRED narrow the gap between agents in simulation and robots operating in the real world [36].

We use ALFRED to evaluate a sequence-to-sequence model with progress monitoring, shown to be effective in vision-and-language navigation tasks [27]. While this model is relatively competent at accomplishing some sub-goals (*e.g.* operating microwaves is similar across **Heat & Place** tasks), the overall task success rates are poor. The long horizon of ALFRED tasks poses a significant challenge with sub-problems including visual semantic navigation, object detection, referring expression grounding, and action grounding. These challenges may be approachable by models that exploit hierarchy [26, 8], modularity [4, 16], and structured reasoning and planning [6]. Such approaches have not been applied to data with the language complexity and long horizon action sequences of ALFRED. We are encouraged by the possibilities and challenges that the ALFRED benchmark introduces to the community.

Acknowledgements

We thank colleagues at the University of Washington for their helpful feedback on this paper and the ALFRED benchmark, in particular: Aaron Walsman, Chris Paxton, Chris Xie, Xiangyun Meng, Zoey Chen, Ramya Korlakai Vinayak, Gabriel Ilharco, Sewon Min, Maxwell Forbes, and Nikolaos Pappas. We also thank Eli VanderBilt and Eric Kolve for their help in adjusting the AI2-THOR simulator to enable the collection of ALFRED, and Victor Zhong for early modeling design. Finally, thanks to Ranjay Krishna for providing a starting point for our Mechanical Turk annotation interface. This research was supported in part by the ARO (ARO-W911NF-16-1-0121), the NSF (IIS1252835, IIS-1562364, NSF-NRI-1637479), the Allen Institute for Artificial Intelligence, a Paul G. Allen School Fellowship to the first author, and an NVIDIA Graduate Research Fellowship to the third author. We would also like to thank NVIDIA for support via the UW NVIDIA AI Lab (NVAIL).

References

[1] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *CVPR*, 2016. 3

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 7

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2, 4, 5, 13

[4] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, June 2016. 8

[5] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 2013. 2

[6] Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *AAAI*, 2018. 8

[7] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *IEEE-RAS*, 2011. 3

[8] Yonatan Bisk, Daniel Marcu, and William Wong. Towards a dataset for human computer communication via grounded language acquisition. In *AAAI Workshop on Symbiotic Cognitive Systems*, 2016. 8

[9] Yonatan Bisk, Deniz Yuret, and Daniel Marcu. Natural language communication with robots. In *NAACL*, 2016. 2

[10] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *ISER*, 2012. 3

[11] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D data in indoor environments. *3DV*, 2017. 1

[12] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *AAAI*, 2017. 1, 2

[13] David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, 2011. 2

[14] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019. 1, 2, 4

[15] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *CVPR*, 2018. 2, 3

[16] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural Modular Control for Embodied Question Answering. In *CoRL*, 2018. 8

[17] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018. 2

[18] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl the planning domain definition language. 1998. 4

[19] Daniel Gordon, Dieter Fox, and Ali Farhadi. What should i do now? marrying reinforcement learning and symbolic planning. *arXiv preprint arXiv:1901.01492*, 2018. 2

[20] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *CVPR*, 2017. 2, 3

[21] Stevan Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990. 1

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 12

[23] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*, 2019. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 12

[25] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*, 2017. 1, 16

[26] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, pages 3675–3683, 2016. 8

[27] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019. 2, 6, 7, 8

[28] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*, 2019. 2

[29] James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael L. Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *RSS*, 2015. 3

[30] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI*, 2006. 2

[31] Jonathan Malmaud, Earl Wagner, Nancy Chang, and Kevin Murphy. Cooking with semantics. In *ACL Workshop on Semantic Parsing*, 2014. 2

[32] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *EMNLP*, 2018. 2

[33] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017. 2

[34] Dipendra Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me daved: Context-sensitive grounding of natural language to manipulation instructions. In *RSS*, 2014. 3

[35] Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. Environment-driven lexicon induction for high-level instructions. In *ACL*, 2015. 2

[36] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *ICCV*, 2019. 1, 8

[37] Khanh Nguyen and Hal Daumé III. Help, Anna! Visual Navigation with Natural Multimodal Assistance via Retrospective Curiosity-Encouraging Imitation Learning. In *EMNLP*, 2019. 2

[38] Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *CVPR*, 2019. 2

[39] Daniel Nyga, Subhro Roy, Rohan Paul, Daehyung Park, Mihai Pomarlan, Michael Beetz, and Nicholas Roy. Grounding robot plans from natural language instructions with incomplete world knowledge. In *CoRL*, 2018. 3

[40] Rohan Paul, Jacob Arkin, Derya Aksaray, Nicholas Roy, and Thomas M. Howard. Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms. *IJRR*, 2018. 3

[41] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, 2018. 1, 2, 4

[42] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *TACL*, 2013. 2, 3

[43] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 1

[44] Ozan Sener, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *ICCV*, 2015. 3

[45] Mohit Shridhar and David Hsu. Interactive visual grounding of referring expressions for human-robot interaction. In *RSS*, 2018. 3

[46] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*, 2019. 2

[47] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *ICRA*, 2010. 2

[48] Jesse Thomason, Daniel Gordon, and Yonatan Bisk. Shifting the baseline: Single modality performance on visual navigation & qa. In *NAACL*, 2019. 7

[49] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *CoRL*, 2019. 2

[50] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *IJCAI*, 2015. 3

[51] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 2

[52] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *ECCV*, 2018. 2

[53] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *CVPR*, 2019. 2, 3

[54] Haonan Yu and Jeffrey Mark Siskind. Grounded language learning from video described with sentences. In *ACL*, 2013. 3

[55] Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L. Berg, and Dhruv Batra. Multi-target embodied question answering. In *CVPR*, 2019. 3

[56] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. Visual semantic planning using deep successor representations. In *ICCV*, 2017. 2

[57] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *CVPR*, 2019. 3

Appendix A. Additional ALFRED Details

We give additional information about the generation of expert demonstrations in AI2-THOR, language directives, the annotation interface used to collect directives, and samples of annotations with their associated demonstrations.

A.1. Expert Demonstrations

When sampling task parameters, we employ an active strategy to maximize data heterogeneity. Figure F1 shows the distribution of high-level task across train, validation seen, and validation unseen folds. Figures F6 and F5 give the distributions of pickup objects and receptacles across the dataset. Each task parameter sample is defined by (t, s, o, r, m), where

- t = the task type;
- s = the scene in AI2-THOR;
- o = the object class to be picked up;
- r = the final destination for o or $\emptyset$ for **Examine**;
- m = the secondary object class for **Stack & Place** tasks ($\emptyset$ for other task types).

To construct the next tuple, we first find the largest source of imbalance in the current set of tuples. For example if $o = apple$ is more common than $o = plunger$, $o = plunger$ will be ranked higher than $o = apple$. We additionally account for the prior distribution of each entity (*e.g.*, if *cup* is already represented in the data often as both o and m, it becomes disfavored by the sampling algorithm for all slots). We do this greedily across all slots until the tuple is complete. Given any partial piece of information about the task, the distributions of the remaining task parameters remain heterogeneous under this sampling, weakening baseline priors such as ignoring the language input and always executing a common task in the environment.

Once a task parameter sample is complete, the chosen scene is instantiated, objects and agent start position are randomized, and the relevant room data is encoded into PDDL rules for an expert demonstration. If the PDDL planner cannot generate an expert demonstration given the room configuration, or if the agent fails an action during execution, for example by running into walls or opening doors onto itself due to physical constraints, the episode is abandoned. We gather three distinct expert demonstrations per task parameter sample. These demonstrations are further vetted by rolling them forward using our wrapper to the AI2-THOR API to ensure that a "perfect" model can reproduce the demonstration. The full sampling generation and verification code will be published along with the dataset.

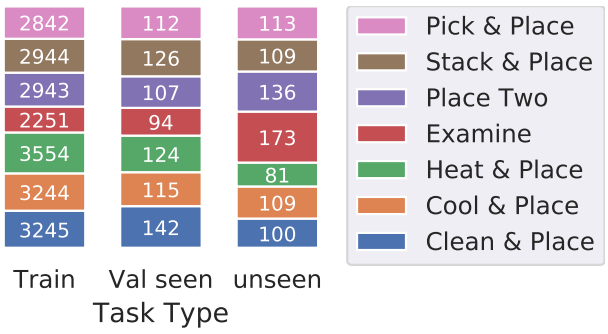


Figure F1: Task distribution across train, validation seen and unseen dataset splits.

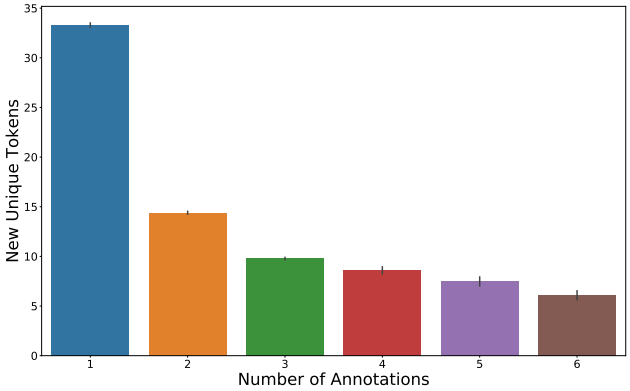


Figure F2: The number of unique tokens introduced per annotation of language directives.

A.2. Example Language Directives

We chose to gather three directives per demonstration empirically. For a subset of over 700 demonstrations, we gathered up to 6 language directives from different annotators. We find that after three annotations, fewer than 10 unique tokens on average are introduced by additional annotators (Figure F2).

A.3. Annotation Interface

Figure F3 shows the Mechanical Turk interface used to gather language annotations. Workers were presented with a video of the expert demonstration with timeline segments indicating sub-goals. The workers annotated each segment while scrubbing through the video, and wrote a short summary description for the entire sequence. We payed workers $0.7 per annotation. During vetting, annotators were paid $0.35 per HIT (Human Interaction Task) to compare 5 sets of three directives each. These wages were set based on local minimum-wage rates and average completion time.

A.4. Dataset Examples

Figure F7 shows 7 expert trajectories (one per task type) and their accompanied annotations.

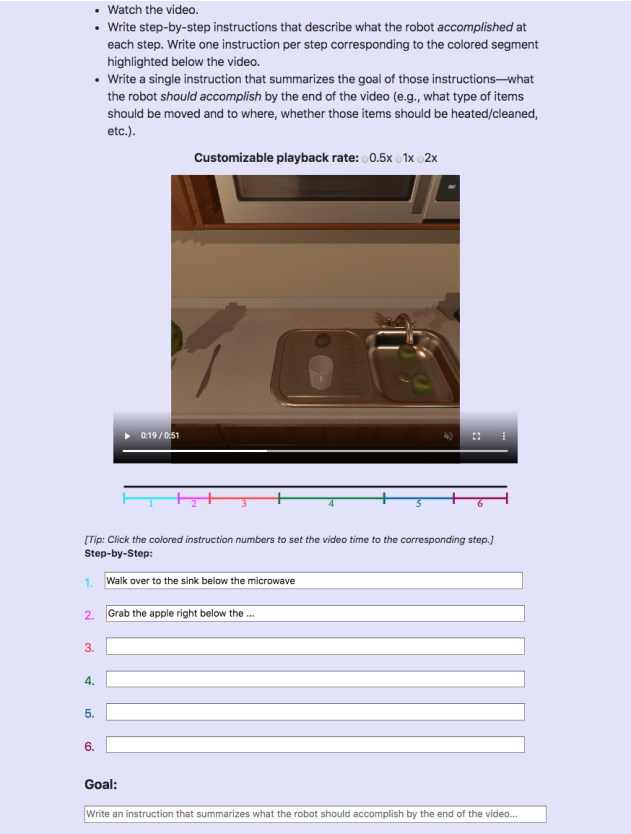


Figure F3: Mechanical Turk Annotation Interface.

Appendix B. Implementation Details

We describe implementation and training details of our baseline Sequence-to-Sequence models.

Preprocessing We tokenize the language directives and convert all tokens to lower-case. During dataset generation, we save images from AI2-THOR 300×300 pixels, and later resize them to 224×224 during training. The generation pipeline saves initialization information for objects and the agent, so all demonstration can be perfectly replayed in the simulator. Researchers can use this replay feature to augment the dataset by saving high-res images, depth maps, or object-segmentation masks.

Network Architecture We use a pretrained ResNet-18 [22] to extract $512 \times 7 \times 7$ features from the `conv5` layer. These features are fed into a two-layer CNN with 1×1 convolutions to reduce the channel dimension from 512 to 64. The $64 \times 7 \times 7$ output is flattened, and a fully-connected layer produces a 2500-dimensional visual feature v_t.

The language encoder is a bi-directional LSTM with a hidden-dimension of 100. We do not use pretrained language models to initialize the LSTM, and the encodings are learned from scratch in an end-to-end manner. We also use a self-attention mechanism to attend over the encodings to initialize the hidden-state of the decoder LSTM.

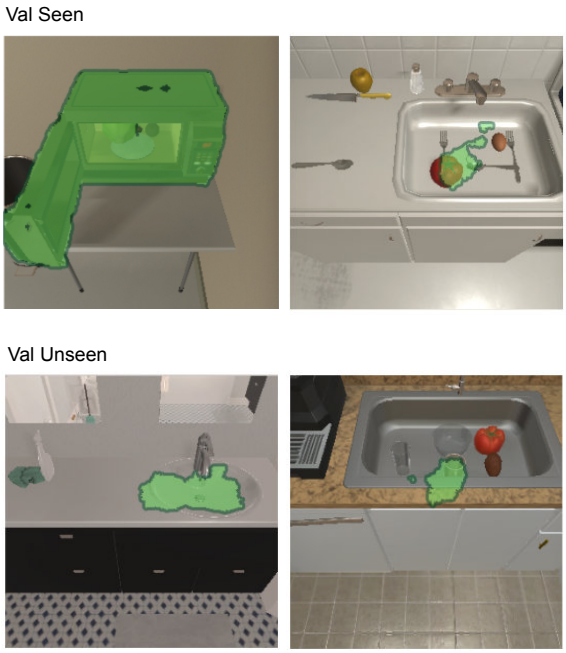


Figure F4: **Predicted interaction masks.** Masks generated by the SEQ2SEQ+PM model are displayed in green.

The action decoder is an LSTM with a hidden-dimension of 512. The actor is a fully-connected layer that outputs logits for 13 actions. The mask decoder is a three-layer deconvolution network, which takes in the concatenated vector u_t and transforms it into $64 \times 7 \times 7$ features with a fully-connected layer. These features are subsequently up-scaled into a $1 \times 300 \times 300$ binary mask through three layers of deconvolutions and up-sampling with bi-linear interpolation.

Training The models were implemented with PyTorch and trained with the Adam optimizer [24] at a learning rate of 1e-4. We use dropout of 0.3 on the visual features and the decoder hidden state, tuned on the validation data. Both the action and mask losses are weighted equally, while the auxiliary losses are scaled with a factor of 0.2. For evaluation, we choose models with the lowest loss on the validation *seen* set. It should be noted that, due to the nature of the tasks, low validation loss might not directly lead to better evaluation performance since the agent does not have to exactly imitate the expert to complete the task.

Notes on Random Agent Unlike discretized navigation where taking random actions might allow the agent to stumble upon the goal, ALFRED tasks are much harder to achieve by chance. The action space branching factor of

Room-to-Room navigation [3], for example, is $4^6 \approx 4000$ (6 average steps and 4 navigation actions). By contrast, the ALFRED average branching factor is $12^{50} \approx 10^{53}$ (50 average steps for 12 actions). Beyond action type prediction, the ALFRED state space resulting from dynamic environments and the need to produce pixel-wise masks for interactive actions explodes further.

B.1. Predicted Masks

Figure F4 shows a few examples of masks generated by the SEQ2SEQ+PM model in seen and unseen validation scenes. The *Microwave* mask accurately captures the contours of the object since the model is familiar with receptacles in seen environments. In contrast, the *Sink* mask in the unseen scene poorly fits the unfamiliar object topology.

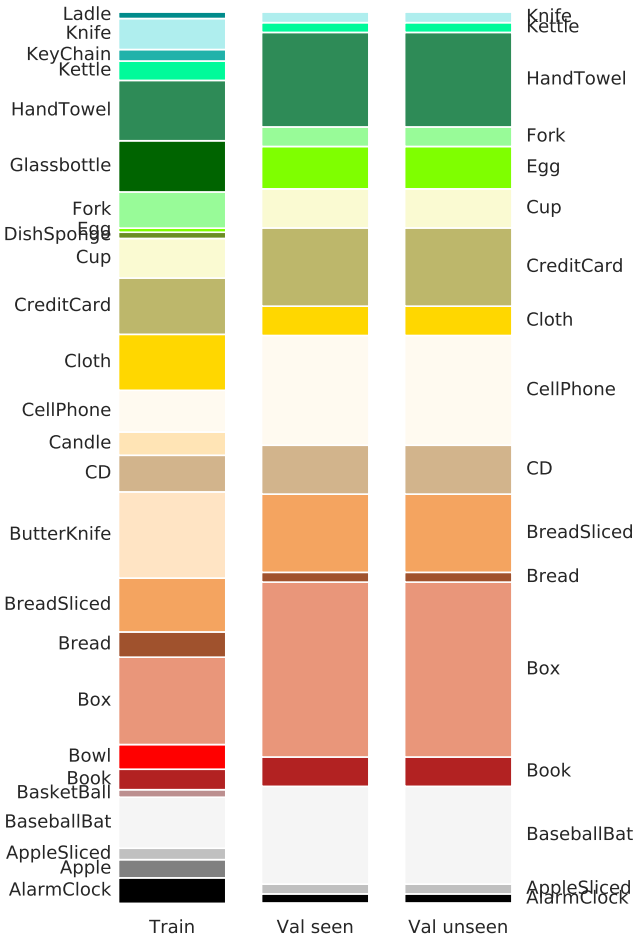


Figure F5: Receptacle distributions in the train, validation *seen* and *unseen* folds.

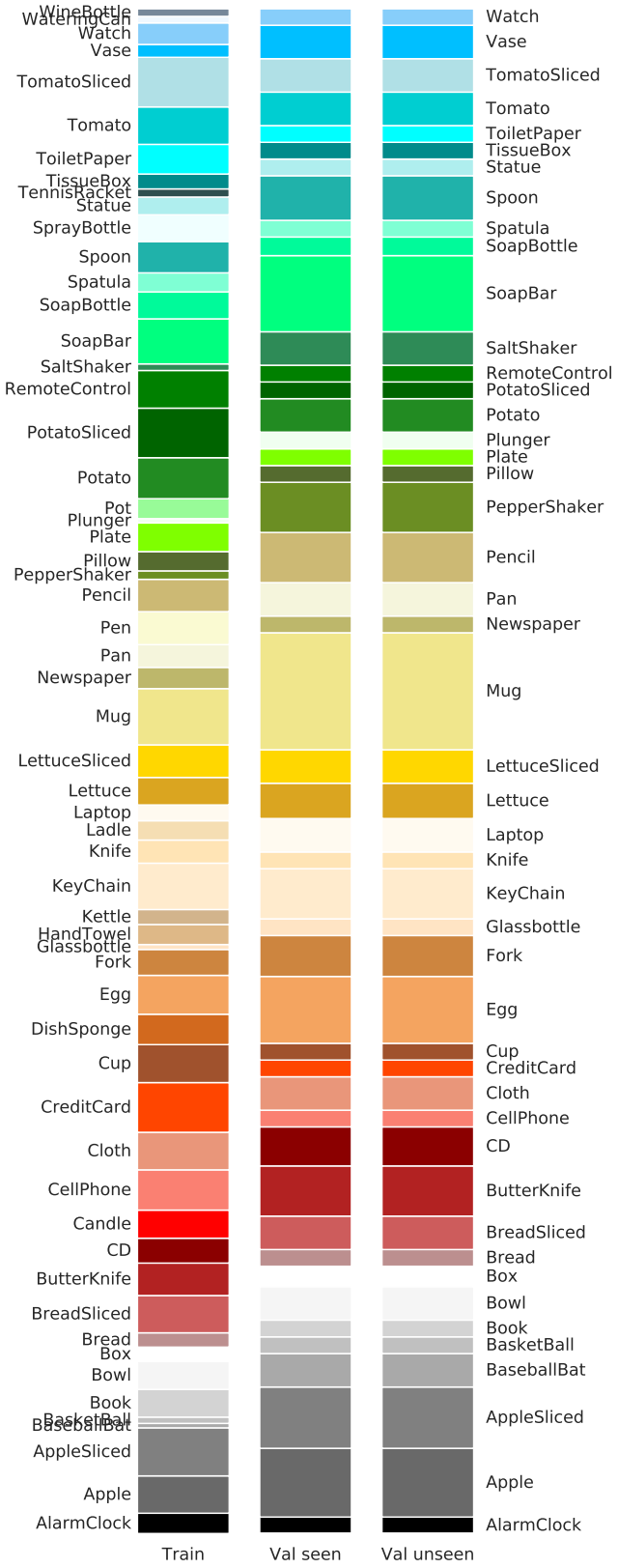
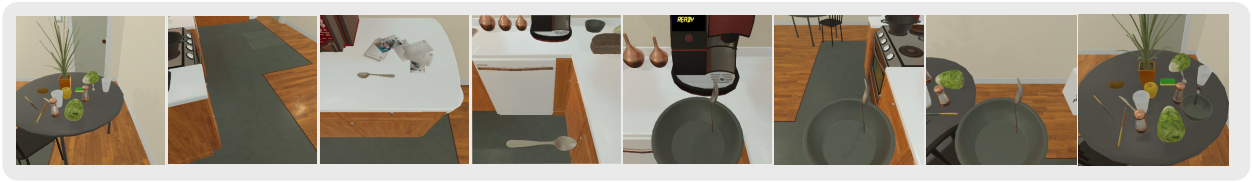


Figure F6: Pickup distributions in the train, validation *seen* and *unseen* folds.

Pick & Place

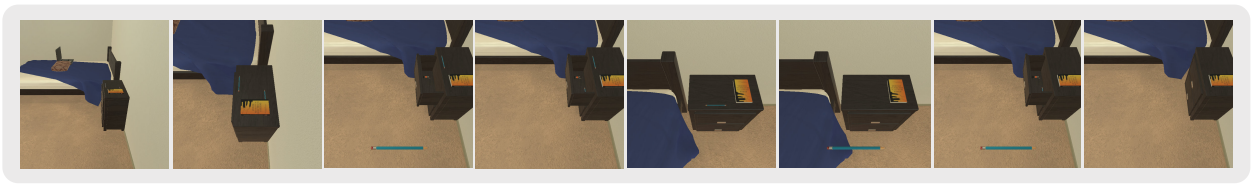
	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Put a watch on a table.**	**Put the watch on the coffee table.**	**Move the watch to the coffee table.**
Instructions	Go to the right and turn around to face the end of the cabinet with the television on it. Pick the watch up from the cabinet. Go to the right and then turn to face the coffee table in front of the couch. Put the watch on the table.	Turn right and go to the TV stand. Pick up the watch from the stand. Turn around and face the coffee table. Put the watch on the coffee table.	Turn right, go straight, turn right, step forward, then turn right to face the table with the TV on it. Pick up the watch on the table, to the right of the remote. Turn right, move forward, turn left, move forward, then turn right to face the coffee table. Put the watch on the front left corner of the coffee table.

Stack & Place



	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Put a bowl with a spoon in it on the table.**	**Move a spoon and bowl to a table.**	**To put a spoon in a bowl plus moving them to the kitchen table.**
Instructions	Turn left, and walk across the room to the microwave. Pick up the spoon. Turn left, and walk to the coffee machine .Put the spoon in the bowl. Pick up the bowl. Turn left, walk to the table, and turn left. Put the bowl on the table.	Move a spoon and bowl to a table. Go to the counter right of the microwave. Pick up the spoon from the counter. Go to the coffeemaker. Place the spoon in a bowl next to the coffeemaker. Pick up the bowl. Go to the black table. Place the bowl down on the table.	Turn left and walk across the room to face a spoon on the right end of the counter. Pick up the spoon on the end of the counter. Turn left and walk across the room to face the coffee maker on the counter. Put the spoon in the bowl to the right of the coffee maker on the counter. Pick up the bowl with a spoon in it on the counter. Turn left and walk across the room and turn left to face the kitchen table. Place the bowl with the spoon in it on the kitchen table.

Pick Two & Place



	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Putting pencils inside of a cabinet.**	**Put two pencils in a drawer.**	**Place the two pencils in the stand.**
Instructions	Walk to the bedside table in front of you. Grab the pencil that's on the table. Move slightly to the left and open the top cabinet of the bedside table. Place the pencil inside the cabinet. Face the front of the bedside table. Grab the pencil off of the bedside table. Open the top cabinet on the table. Place the pencil inside the cabinet and then close it.	Walk straight ahead to the end table. Pick up the blue pencil on the right. Walk around to the front of the end table. Put the pencil in the top drawer of the end table. Look up at the top of the end table. Pick up the pencil on the table. Look down at the drawer. Put the pencil in the drawer on top of the other pencil.	Walk to the stand next to the bed. Grab the pencil from the stand. Open the shelf inside of the stand. Place the pencil in the top shelf of the stand. Close the shelf, walk back to the stand. Grab the other pencil from the stand. Walk back to the stand next to the bed. Place the pencil in the top shelf of the stand.

Clean & Place

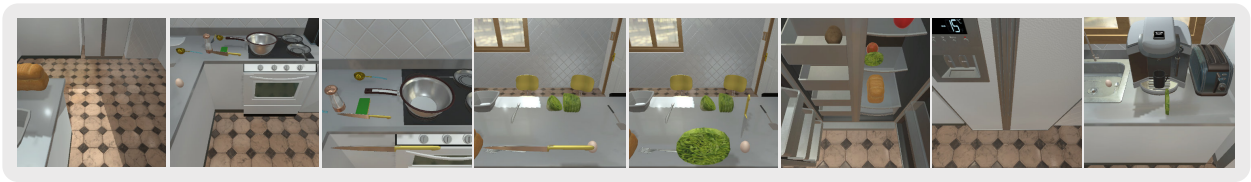
	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Put a clean rag on the top shelf of a barred rack.**	**Wash the pink towel on the shelf, put it back on the shelf.**	**Clean a red cloth.**
Instructions	Turn around, go to the barred rack. Pick up the rag from the bottle shelf of the barred rack. Go to the sink on the left. Put the rag in the sink, turn on then turn off the water. Go to the barred rack to the right of the sink. Put the rag on the top shelf of the barred rack.	Wash the pink towel on the shelf, put it back on the shelf. Turn around and go the shelf. Pick up the pink towel on the shelf. Turn around and put the towel in the sink. Fill the sink with water and wash the towel, take the towel out. Go back to the shelf. Put the towel back on the shelf.	walk on over to the towel drying rack. pick up a dirty red cloth from the towel rack. walk over to the left side of the bathroom sink. turn on the water to rinse the dirty red cloth and pick it back up again. walk back over to the towel drying rack. place the clean cloth on the drying rack.

Figure F7: **Dataset Examples.** Annotations for seven expert demonstrations.

Heat & Place

	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Put a cooked potato slice on the counter.**	**Place a slice of cooked potato onto the counter.**	**Put a piece of cooked potato on the counter.**
	Turn right, turn right, walk past the sink, turn left to face round table with tablecloth. Pick up the yellow-handled knife from the table. Cut a slice in the potato on the table. Turn left, turn left, turn right at counter, cross room, turn left at refrigerator to face counter. Put knife down on the table. Turn left, walk past sink, turn left to face round table. Pick the potato slice up from the table. Turn left, make right around corner of counter, turn left to face stove and microwave. Put potato in microwave, cook it, take it out of microwave. Turn right, cross room, turn left at counter with blue plate on it. Put potato on the counter in front of the blue plate.	Turn right, move to the table. Pick up the knife from the table. Slice the potato on the table. Turn left, move to the counter left of the bread. Put the knife on the counter near the soap container. Turn left, move to the table. Pick up a slice of potato from the table. Turn left, move to the counter in front of the stove. Put the potato slice into the microwave, cook it, pick it back up. Turn right, move to the counter left of the bread. Put the cooked potato slice on the counter.	Turn right and cross the room, then turn left and go to face the gray table. Pick up the knife from in between the lettuce and the apple. Use the knife to slice the potato that's on the gray table. Bring the knife with you and go face the kitchen counter with the loaf of bread. Put the knife down in front of the soap dispenser on the counter. Go back over to the gray table. Pick up a slice of the cut potato from the table. Bring the potato with you and go over to the stove, then look up at the microwave. Cook the potato slice in the microwave, then take it out again. Bring the potato slice over to the counter top with the loaf of bread and the knife you used to cut it. Put the potato slice down in front of the blue plate.

Cool & Place



	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Put a slice of cold lettuce on a counter.**	**Put a chilled slice of lettuce on the counter.**	**Slice some lettuce and cool it in the refrigerator so you can put it on the counter top.**
Instructions	Turn left, go forward past the counter, turn left, go forward to the counter to the left of the oven. Take the knife to the left of the large spoon from the counter. Turn around, go forward a step, turn right to the counter. Cut the lettuce on the counter into slices. Turn right, go forward a step, turn left to the counter. Put the knife behind the egg on the counter. Turn left, go forward a step, turn right to the counter. Take a slice of lettuce from the counter. Turn left, go forward, turn right to the fridge. Go to the fridge. Chill the lettuce in the fridge in front of the apple. Take the lettuce from the fridge. Turn left, go forward, turn right to face the coffee maker. Put the lettuce in front of the coffee maker on the counter.	Turn left, head toward the fridge, turn left and go to the stove. Pick up the knife beside the spoon on the counter. Turn around and turn to the right to face the counter with the egg. Cut the lettuce on the counter. Move right and back left to the counter. Put the knife behind the egg on the counter. Move the left and turn back right towards the counter. Pick up a slice of lettuce. Turn around and head to the fridge. Put the lettuce on the second shelf of the fridge, close the fridge, open the fridge and pick it back up. Turn left, go halfway across the room and turn right toward the coffee maker. Put the slice of lettuce on the counter in front of the coffee maker.	turn left and go around the counter top, then go straight to the stove top. pick up the knife with the yellow handle from behind the salt shaker on the counter top. turn left and face the counter top to your left. slice the lettuce on the counter top. face the counter top with the knife in hand. place the knife down next to the lettuce slices on the counter top. face the lettuce on the counter top. pick up a slice of lettuce on the counter top. turn left, then face the opposite wall behind you to face the refrigerator. open the refrigerator, and place the slice of lettuce in front of the apple one shelf above the bread, then shut the door and open it up again after several seconds to pick the lettuce slice up. turn left, then face forward to the part of the counter top on which the coffee maker sits. place the slice of lettuce in front of the coffee maker.

Examine in Light

	Annotation # 1	Annotation # 2	Annotation # 3
Goals	**Read a book by lamp light.**	**Examine a book with a lamp.**	**Pick up a book and turn on a lamp.**
Instructions	Head forward to the bed in front of you. Pick up the blue book that is sitting on the bed; the book that says Probabilistic Robotics. Turn to your right and walk to the night stand. Turn on the lamp that is sitting on the night stand.	walk forward a few steps, turn right, take two steps, turn left, walk to bed. pick up the book that is on the bed. turn around, take a step, turn left to face small table. turn the lamp on.	Walk forward to face the bed. Pick the book up from the bed. Turn to the right and face the night stand with the lamp. Turn the lamp on.

Figure F7: **Dataset Examples.** Annotations for seven expert demonstrations.

Figure F8: **Visual diversity of AI2-THOR [25] scenes.** Top to bottom rows: kitchens, living rooms, bedrooms, and bathrooms. Object locations are randomized based on placeable surface areas and class constraints. See https://ai2thor.allenai.org/demo/ for an interactive demo.