Embodied Question Answering

Abhishek Das[1*], Samyak Datta[1], Georgia Gkioxari[2], Stefan Lee[1], Devi Parikh[2,1], Dhruv Batra[2,1]
[1]Georgia Institute of Technology, [2]Facebook AI Research
[1]{abhshkdz, samyak, steflee}@gatech.edu [2]{gkioxari, parikh, dbatra}@fb.com
embodiedqa.org

Abstract

*We present a new AI task – **Embodied Question Answering** (EmbodiedQA) – where an agent is spawned at a random location in a 3D environment and asked a question (*'What color is the car?'*). In order to answer, the agent must first intelligently navigate to explore the environment, gather information through first-person (egocentric) vision, and then answer the question (*'orange'*).*

This challenging task requires a range of AI skills – active perception, language understanding, goal-driven navigation, commonsense reasoning, and grounding of language into actions. In this work, we develop the environments, end-to-end-trained reinforcement learning agents, and evaluation protocols for EmbodiedQA.

1. Introduction

> The embodiment hypothesis is the idea that intelligence emerges in the interaction of an agent with an environment and as a result of sensorimotor activity.
>
> *Smith and Gasser [1]*

Our long-term goal is to build intelligent agents that can *perceive* their environment (through vision, audition, or other sensors), *communicate* (*i.e.*, hold a natural language dialog grounded in the environment), and *act* (*e.g.* aid humans by executing API calls or commands in a virtual or embodied environment). In addition to being a fundamental scientific goal in artificial intelligence (AI), even a small advance towards such intelligent systems can *fundamentally change our lives* – from assistive dialog agents for the visually impaired, to natural-language interaction with self-driving cars, in-home robots, and personal assistants.

As a step towards goal-driven agents that can perceive, communicate, and execute actions, we present a new AI task – ***Embodied Question Answering*** (EmbodiedQA), along with virtual environments, evaluation metrics, and a novel deep reinforcement learning (RL) model for this task.

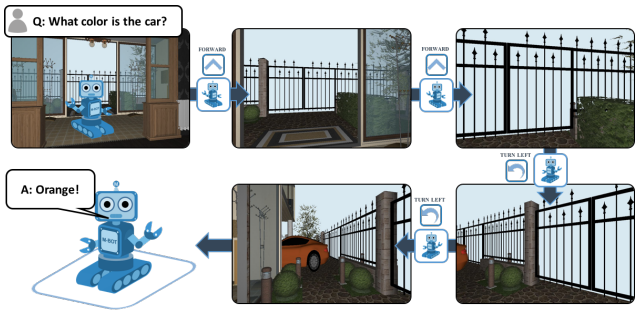


Figure 1: Embodied Question Answering – EmbodiedQA– tasks agents with navigating rich 3D environments in order to answer questions. These embodied agents must jointly learn language understanding, visual reasoning, and navigation to succeed.

Concretely, the EmbodiedQA task is illustrated in Fig. 1 – an agent is spawned at a random location in an environment (a house or building) and asked a question (*e.g. 'What color is the car?'*). The agent perceives its environment through first-person vision (a single RGB camera) and can perform a few atomic actions: move-{forward, backward, right, left} and turn-{right, left}. The goal of the agent is to intelligently navigate the environment and gather the visual information necessary to answer the question.

EmbodiedQA is a challenging task that subsumes several fundamental AI problems as sub-tasks. Clearly, the agent must understand language (*what is the question asking?*) and vision (*what does a car look like?*), but a successful agent must also learn to perform:

Active Perception: The agent may be spawned anywhere in the environment and may not immediately 'see' the pixels containing the answer to the visual question (*i.e.* the car may not be visible). Thus, the agent *must* move to succeed – controlling the pixels that it will perceive. The agent must learn to map its visual input to the correct action based on its perception of the world, the underlying physical constraints, and its understanding of the question.

*Work partially done during an internship at Facebook AI Research.

Common Sense Reasoning: The agent is not provided a floor-plan or map of the environment, and must navigate from egocentric views alone. Thus, it must learn common sense (*where am I? where are cars typically found in a housing compound? and where is the garage with respect to me?*) similar to how humans may navigate in a house they have never visited (*the car is probably in the garage outside, so I should find a door that leads out*).

Language Grounding: One commonly noted shortcoming of modern vision-and-language models is their lack of grounding – these models often fail to associate entities in text with corresponding image pixels, relying instead on dataset biases to respond seemingly intelligently even when attending to irrelevant regions [2, 3]. In EmbodiedQA, we take a goal-driven view of grounding – our agent grounds a visual question not into pixels but into a sequence of actions ('garage' *means* to navigate towards the house exterior where the 'car' is usually parked).

Credit Assignment: From a reinforcement learning perspective, EmbodiedQA presents a particularly challenging learning problem. Consider the question *'How many rooms contain chairs?'*. How does an agent discover that this question involves exploring the environment to visit 'rooms', detecting 'chairs', incrementing a count every time a 'chair' is in the view (except while the agent is in the same 'room'), and stopping when no more 'rooms' can be found? All without knowing what a 'room' is or how to find it, what a 'chair' looks like, or what counting is. To succeed, the agent must execute a somewhat precise sequence of hundreds of inter-dependent actions (forward, forward, turn-right, forward, forward, ..., turn-left, '5') – all to be learned from a reward signal that says '4' is the right answer and anything else is incorrect. The task is complex enough that most random action sequences result in negative reward, and when things do go wrong, it's difficult for the agent to know why – was the question misunderstood? Can the agent not detect chairs? Did the agent navigate incorrectly? Was the counting incorrect?

As the first step in this challenging space, we judiciously scope out a problem space – environments, question types, learning paradigm – that allow us to augment the sparse RL rewards with imitation learning (showing the agent example trajectories) and reward shaping [4] (giving intermediate 'getting closer or farther' navigation rewards). Specifically, our approach follows the recent paradigm from robotics and deep RL [5, 6] – that the training environments are assumed to be sufficiently *instrumented* – *i.e.*, provide access to the agent location, depth and semantic annotations of the environment, and allow for computing obstacle-avoiding shortest paths from the agent to any target location.

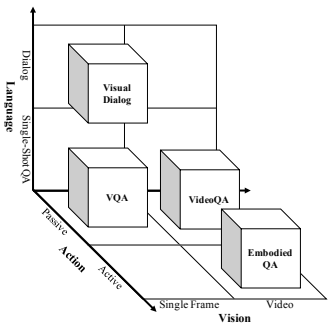
Crucially, at test time, our agents operate entirely from egocentric RGB vision alone – no structured representation of the environments, no access to a map, no explicit localization of the agent or mapping of the environment, no A* or any other heuristic planning, and no pre-processing or hand-coded knowledge about the environment or the task of any kind. The agent in its entirety – vision, language, navigation, answering modules – is trained completely end-to-end – from raw sensory input (pixels and words) to goal-driven multi-room indoor navigation to visual question answering!

Contributions. We make the following contributions:

- We propose a new AI task: EmbodiedQA, where an agent spawned in an environment must intelligently navigate from an egocentric view to gather the necessary information to answer visual questions about its environment.
- We introduce a novel Adaptive Computation Time [7] navigator – that decomposes navigation into a 'planner' that selects actions, and a 'controller' that executes these primitive actions a variable number of times before returning control to the planner. When the agent decides it has seen the required visual information to answer the question, it stops navigating and outputs an answer.
- We initialize our agents via imitation learning and show that agents can answer questions more accurately after fine-tuning with reinforcement learning – that is, when allowed to control their own navigation *for the express purpose* of answering questions accurately. Unlike some prior work, we explicitly test and demonstrate *generalization of our agents to unseen environments*.
- We evaluate our agents in *House3D* [8], a rich, interactive environment based on human-designed 3D indoor scenes from the SUNCG dataset [9]. These diverse virtual environments enable us to test generalization of our agent across floor-plans, objects, and room configurations – without the concerns of safety, privacy, and expense inherent to real robotic platforms.
- We introduce the EQA dataset of visual questions and answers grounded in House3D. The different question types test a range of agent abilities – scene recognition (`location`), spatial reasoning (`preposition`), color recognition (`color`). While the EmbodiedQA task definition supports free-form natural language questions, we represent each question in EQA as a *functional program* that can be programmatically generated and executed on the environment to determine the answer. This gives us the ability to control the distribution of question-types and answers in the dataset, deter algorithms from exploiting dataset bias [3, 10], and provide fine-grained breakdown of performance by skill.
- We integrated House3D renderer with Amazon Mechanical Turk (AMT), allowing subjects to *remotely operate the agent*, and collected expert demonstrations of question-based navigation that serve as a benchmark to compare our proposed and future algorithms.

All our code and data will be made publicly available.

2. Related Work



We place our work in context by arranging prior work along the axes of vision (from a single-frame to video), language (from single-shot question answering to dialog), and action (from passive observers to active agents). When viewed from this perspective, EmbodiedQA presents a novel problem configuration – single-shot QA about videos captured by goal-driven active agents. Next, we contrast this against various 2D slices in this space.

VQA: Vision + Language. Like EmbodiedQA, image and video question answering tasks [11–15] require reasoning about natural language questions posed about visual content. The crucial difference is the *lack of control* – these tasks present answering agents with a fixed view of the environment (*i.e*. one or more images from some fixed trajectory through the world) from which the agent must answer the question, never allowing the agents to *actively* perceive. In contrast, EmbodiedQA agents control their trajectory and fate, for good or ill. The task is significantly harder than VQA (*i.e*. most random paths are useless) but the agent has the flexibility to avoid confusing viewpoints and seek visual input that will maximize answer confidence.

Visual Navigation: Vision + Action. The problem of navigating in an environment based on visual perception has long been studied in vision and robotics (see [16] for an extensive survey). Classical techniques divide navigation into two distinct phases – mapping (where visual observations are used to construct a 3D model of the environment), and planning (which selects paths based on this map). Recent developments in deep RL have proposed fused architectures that go directly from egocentric visual observations to navigational actions [17–23]. We model our agents as similar pixel-to-action navigators. The key distinction in EmbodiedQA is how the goals are specified. Visual navigation typically specifies agent goals either implicitly via the reward function [19, 20] (thus training a separate policy for each goal/reward), or explicitly by conditioning on goal state representations [24] including images of target objects [18]. In contrast, EmbodiedQA specifies agent goals via language, which is inherently compositional and renders training a separate policy for every task (question) infeasible.

Situated Language Learning: Language + Action. Inspired by the classical work of Winograd [25], a number of recent works have revisited grounded language learning by situating agents in simple globally-perceived environments and tasking them with goals specified in natural language. The form and structure of these goal specifications range from declarative programs [26], to simple templated commands [27,28], to free-form natural language instructions [29, 30]. One key distinction in EmbodiedQA, of course, is visual sensing – the environment is only partially observable, *i.e*. the agent does not have access to the floor plan, object labels, attributes, *etc*., and must extract this information purely from first-person visual sensing.

Embodiment: Vision + Language + Action. Closest to EmbodiedQA are recent works that extend the situated language learning paradigm to settings where agents' perceptions are local, purely visual, and change based on their actions – a setting we refer to as embodied language learning.

In concurrent and unpublished work, Hermann *et al*. [21] and Chaplot *et al*. [17] both develop embodied agents in simple game-like environments consisting of 1-2 rooms and a handful of objects with variable color and shape. In both settings, agents were able to learn to understand simple *'go to X'*/*'pick up X'* style commands where X would specify an object (and possibly some of its attributes). Similarly, Oh *et al*. [23] present embodied agents in a simple maze-world and task them to complete a series of instructions. In contrast to these approaches, our EmbodiedQA environments consist of multi-room homes (~8 per home) that are densely populated by a variety of objects (~54 unique objects per home). Furthermore, the instructions and commands in these works are low-level and more closely relate to actions than the questions presented in EmbodiedQA.

Interactive Environments. There are a number of interactive environments commonly used in the community, ranging from simple 2D grid-worlds (*e.g*. XWORLD [27]), to 3D game-like environments with limited realism (*e.g*. DeepMind Lab [31] or Doom [17]), to more complex, realistic environments (*e.g*. AI2-THOR [19] or Stanford 2D-3D-S [32]). While realistic environments provide rich representations of the world, most consist of only a handful of environments due to the high difficulty of their creation. On the other hand, large sets of synthetic environments can be programmatically generated; however, they typically lack realism (either in appearance or arrangement). In this work, we use the House3D [8] environment as it strikes a useful middle-ground between simple synthetic and realistic environments. See Sec. 3.1 for more details.

Hierarchical Agents. We model our EmbodiedQA agents as deep hierarchical agents that decompose the overall control problem such that a higher-level planner invokes lower-level controls to issue primitive actions. Such hierarchical modeling has recently shown promise in the deep reinforcement learning setting [23, 28, 33]. Our model also draws inspiration from the work on Adaptive Computation Time models of Graves [7].

(a) Sample Environments

gym	dining room
patio	living room
office	bathroom
lobby	bedroom
garage	elevator
kitchen	balcony

(b) Queryable Rooms

rug	piano	dryer	computer	fireplace	whiteboard	bookshelf	wardrobe cabinet
pan	toilet	plates	ottoman	fish tank	dishwasher	microwave	water dispenser
bed	table	mirror	tv stand	stereo set	chessboard	playstation	vacuum cleaner
cup	xbox	heater	bathtub	shoe rack	range oven	refrigerator	coffee machine
sink	sofa	kettle	dresser	knife rack	towel rack	loudspeaker	utensil holder
desk	vase	shower	washer	fruit bowl	television	dressing table	cutting board
ironing board		food processor					

(c) Queryable Objects

Figure 2: The EQA dataset is built on a subset of the environments and objects from the SUNCG [9] dataset. We show (a) sample environments and the (b) rooms and (c) objects that are asked about in the EmbodiedQA task.

3. EQA Dataset: Questions In Environments

Having placed EmbodiedQA in context, we now dive deeper by outlining the environments in which our agents are embodied and the questions they must answer. We will publicly release the environments, our curated EQA dataset, and our code to aid research in this nascent area.

3.1. House3D: Interactive 3D Environments

We instantiate EmbodiedQA in House3D [8], a recently introduced rich, interactive environment based on 3D indoor scenes from the SUNCG dataset [9]. Concretely, SUNCG consists of synthetic 3D scenes with realistic room and furniture layouts, manually designed using an online interior design interface (Planner5D [34]). Scenes were also further 'verified' as realistic by majority vote of three human annotators. In total, SUNCG contains over 45k environments with 49k valid floors, 404k rooms containing 5 million object instances of 2644 unique objects from 80 different categories. House3D converts SUNCG from a static 3D dataset to a set of virtual environments, where an agent (approximated as a cylinder 1 meter high) may navigate under simple physical constraints (not being able to pass through walls or objects). Fig. 2a shows top-down views of sample environments. Full details may be found in [8].

We build the EQA dataset on a pruned subset of environments from House3D. First, we only consider environments for which all three SUNCG annotators consider the scene layout realistic. Next, we filter out atypical environments such as those lacking ground or those that are too small or large (only keeping houses with an internal area of 300-800m^2 covering at least $1/3$ the total ground area). Finally, we exclude non-home environments by requiring at least one kitchen, living room, dining room, and bedroom.

3.2. Question-Answer Generation

We would like to pose questions to agents that test their abilities to ground language, use common sense, reason visually, and navigate the environments. For example, answering the question *'What color is the car?'* ostensibly requires grounding the symbol *'car'*, reasoning that cars are typically outside, navigating outside and exploring until the car is found, and visually inspecting its color.

We draw inspiration from the CLEVR [35] dataset, and programmatically generate a dataset (EQA) of grounded questions and answers. This gives us the ability to control the distribution of question-types and answers in the dataset, and deter algorithms from exploiting dataset bias.

Queryable Rooms and Objects. Figs. 2c, 2b show the queryable rooms (12) and objects (50) in EQA. We exclude objects and rooms from SUNCG that are obscure (*e.g.* loggia rooms) or difficult to resolve visually (*e.g.* very small objects like light switches). We merge some semantically similar object categories (*e.g.* teapot, coffee kettle) and singular vs plural forms of the same object type (*e.g.* (books, book)) to reduce ambiguity.

Questions as Functional Programs. Each question in EQA is represented as a functional program that can be executed on the environment yielding an answer[1]. These functional programs are composed of a small set of elementary operations ($select(\cdot)$, $unique(\cdot)$, $query(\cdot)$, *etc.*) that operate on sets of room or object annotations.

The number and the order of evaluation of these elementary operations defines a *question type* or template. For instance, one question type in EQA is the `location` template:

`location`: *'What room is the <OBJ> located in?'*

where <OBJ> refers to one of the queryable objects. The sequence of elementary operations for this question type is:

$$select(objects) \rightarrow unique(objects) \rightarrow query(location).$$

The first function, $select(objects)$, gets all the object names from the environment. The second, $unique(objects)$, retains only the objects that have a single instance in the entire house. The third, $query(location)$, generates a question (by filling in the appropriate template) for each such object. The 2nd operation, $unique(objects)$, is particularly important to generate unambiguous questions. For instance, if there are two air conditioners in the house, the question *'What room is the air conditioner lo-*

[1] or a response that the question is inapplicable (*e.g.* referring to objects not in the environment) or ambiguous (having multiple valid answers).

cated in?' is ambiguous, with potentially two different answers depending on which instance is being referred to.

Question Types. Associated with each question type is a template for generating a question about the rooms and objects, their attributes and relationships. We define nine question types and associated templates in EQA:

EQA v1

`location`:	*'What room is the <OBJ> located in?'*
`color`:	*'What color is the <OBJ>?'*
`color_room`:	*'What color is the <OBJ> in the <ROOM>?'*
`preposition`:	*'What is <on/above/below/next-to> the <OBJ> in the <ROOM>?'*
`existence`:	*'Is there a <OBJ> in the <ROOM>?'*
`logical`:	*'Is there a(n) <OBJ1> and a(n) <OBJ2> in the <ROOM>?'*
`count`:	*'How many <OBJs> in the <ROOM>?'*
`room_count`:	*'How many <ROOMs> in the house?'*
`distance`:	*'Is the <OBJ1> closer to the <OBJ2> than to the <OBJ3> in the <ROOM>?'*

The <ROOM> and <OBJ> tags above can be filled by any valid room or object listed in Fig. 2b and Fig. 2c respectively. Given these question templates, the possible answers are room names (`location`), object names (`preposition`), yes/no (`existence`, `logical` and `distance`), color names (`color`) or numbers (`count`).

These questions test a range of agent abilities including object detection (`existence`), scene recognition (`location`), counting (`count`), spatial reasoning (`preposition`), color recognition (`color`), and logical operators (`logic`). Moreover, many of these questions require multiple capabilities: *e.g.*, answering a `distance` question requires recognizing the room and objects as well as reasoning about their spatial relation. Furthermore, the agent must do this by navigating the environment to find the room, looking around the room to find the objects, and possibly remembering their positions through time (if all three objects are not simultaneously visible).

Different question types also require different degrees of navigation and memory. For instance, *'How many bedrooms in the house?'* requires significant navigation (potentially exploring the entire environment) and long-term memory (keeping track of the count), while a question like *'What color is the chair in the living room?'* requires finding a single room, the living room, and looking for a chair.

EQA is easily extensible to include new elementary operations, question types, and templates as needed to increase the difficulty of the task to match the development of new models. As a first step in this challenging space, our experiments focus on EQA v1, which consists of 4 question types – `location`, `color`, `color_room`, `preposition`. One virtue of these questions is that there is a single target queried object (<OBJ>), which enables the use of shortest paths from the agent's spawn location to the target as expert demonstrations for imitation learning (details in Section 4.1). We stress that EQA is not a static dataset, rather a curriculum of capabilities that we would like to achieve in embodied communicating agents.

Question-Answer Generation and Dataset Bias. In principle, we now have the ability to automatically generate all valid questions and their associated answers for each environment by executing the functional programs on the environment's annotations provided by SUNCG. However, careful consideration is needed to make sure the developed dataset is balanced over question types and answers.

For each filled question template (*e.g. 'What room is the refrigerator located in?'*), we execute its functional form on all associated environments in the dataset (*i.e.* those containing refrigerators) to compute the answer distribution for this question. We exclude questions for which the normalized entropy of the answer distribution is below 0.5 – *e.g.*, an agent can simply memorize that refrigerators are almost always in kitchens, so this question would be discarded. We also exclude questions occurring in fewer than four environments as the normalized entropy estimates are unreliable.

Finally, in order to benchmark performance of agents vs human performance on EQA, it is important for the questions to not be tedious or frustrating for humans to answer. We do not ask `count` questions for objects with high counts (>=5) or `distance` questions between object triplets without clear differences in distance. We set these thresholds and room / object blacklists manually based on our experience performing these tasks.

Complete discussion of the question templates, functional programs, elementary operations, and various checks-and-balances can be found in the supplement.

	Environments	Unique Questions	Total Questions
`train`	643	147	4246
`val`	67	104	506
`test`	57	105	529

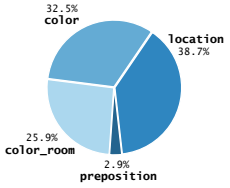


Figure 3: Overview of the EQA v1 dataset including dataset split statistics (left) and question type breakdown (right).

EQA v1 Statistics. The EQA v1 dataset consists of over 5000 question across over 750 environments, referring to a total of 45 unique objects in 7 unique room types. The dataset is split into `train`, `val`, `test` such that there is no overlap in environments across splits. Fig. 3 shows the dataset splits and question type distribution. Approximately 6 questions are asked per environment on average, 22 at most, and 1 at fewest. There are relatively few

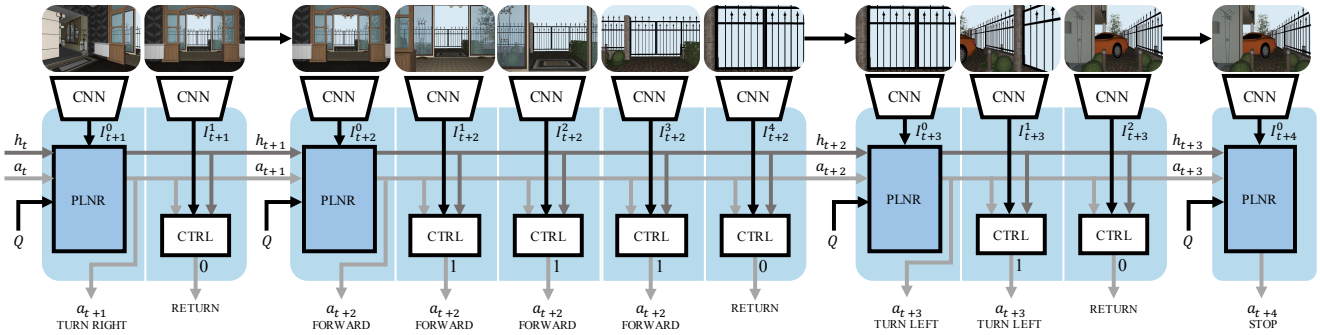


Figure 4: Our Adaptive Computation Time (ACT) navigator splits the navigation task between a planner and a controller module. The planner selects actions and the controller decides to continue performing that action for a variable number of time steps – resulting in a decoupling of direction (*'turn left'*) and velocity (*'5 times'*) and strengthening the long-term gradient flows of the planner module.

`preposition` questions as many frequently occurring spatial relations are too easy to resolve without exploration and fail the entropy thresholding. We will make EQA v1 and the entire generation engine publicly available.

4. A Hierarchical Model for EmbodiedQA

We now introduce our proposed neural architecture for an EmbodiedQA agent. Recall that the agent is spawned at a random location in the environment, receives a question, and perceives only through a single egocentric RGB camera. Importantly, unlike some prior work [26–30], in EmbodiedQA, the agent does not receive any global or structured representation of the environment (map, location, objects, rooms), or of the task (the functional program that generated the question).

Overview of the Agent. The agent has 4 natural modules – vision, language, navigation, answering – and is trained from raw sensory input (pixels and words) to goal-driven multi-room indoor navigation to visual question answering. The modules themselves are built up largely from conventional neural building blocks – Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). One key technical novelty in our model is the use of Adaptive Computation Time (ACT) RNNs by Graves [7], which is an elegant approach for allowing RNNs to learn how many computational steps to take between receiving an input and emitting an output by back-propagating through a 'halting' layer. We make use of this idea in our navigation module to cleanly separate the decision between – direction (where to move, decided by a 'planner') and velocity (how far to move, decided by a 'controller'). Fig. 4 illustrates the different modules in our agent, which we describe next.

Vision. Our agent takes egocentric 224×224 RGB images from the House3D renderer as input, which we process with a CNN consisting of 4 $\{5\times5$ Conv, ReLU, BatchNorm, 2×2 MaxPool$\}$ blocks, producing a fixed-size representation.

A strong visual system for EmbodiedQA should encode information about object attributes (*i.e.* colors and textures), semantics (*i.e.* object categories), and environmental geometry (*i.e.* depth). As such, we pretrain the CNN under a multi-task pixel-to-pixel prediction framework – treating the above CNN as an encoder network, we train multiple network heads to decode the 1) original RGB values, 2) semantic class, and 3) depth of each pixel (which can be obtained from the House3D renderer). See the supplementary material for full model and training details.

Language. Our agents also receive questions which we encode with 2-layer LSTMs with 128-dim hidden states. Note that we learn *separate* question encoders for the navigation and answering modules – as each may need to focus on different parts of the question. For instance, in the question *'What color is the chair in the kitchen?'*, *'color'* is irrelevant for navigation and *'kitchen'* matters little for question answering (once in the kitchen).

Navigation. We introduce a novel Adaptive Computation Time (ACT) navigator that decomposes navigation into a 'planner', that selects actions (forward, left, right), and a 'controller', that executes these primitive actions a variable number of times (1,2, ...) before returning control back to the planner. Intuitively, this structure separates the intention of the agent (*i.e.* get to the other end of the room) from the series of primitive actions required to achieve this directive (*i.e. 'forward, forward, forward, ...'*), and is reminiscent of hierarchical RL approaches [23, 28, 33]. This division also allows the planner to have variable time steps between decisions, strengthening long-term gradient flows.

Formally, let $t = 1, 2, \ldots, T$ denote planner timestamps, and $n = 0, 1, 2, \ldots N(t)$ denote the variable number of controller steps. Let I_t^n denote the encoding of the image observed at t-th planner-time and n-th controller-step. We instantiate the planner as an LSTM. Thus, the planner maintains a hidden state h^t (that is updated only at planner timesteps), and samples an action $a_t \in \{$forward, turn-left, turn-right, stop-navigation$\}$:

$$a_t,\ h_t \leftarrow \texttt{PLNR}\left(h_{t-1},\ I_t^0,\ Q,\ a_{t-1}\right), \tag{1}$$

where Q is the question encoding. After taking this action, the planner passes control to the controller, which considers the planner's state and the current frame to decide to continue performing a_t or to return control to the planner, *i.e.*

$$\{0, 1\} \ni c_t^n \leftarrow \texttt{CTRL}\left(h_t, a_t, I_t^n\right) \tag{2}$$

If $c_t^n = 1$ then the action a_t repeats and CTRL is applied to the next frame. Else if $c_t^n = 0$ or a max of 5 controller-steps has been reached, control is returned to the planner. We instantiate the controller as a feed-forward multi-layer perceptron with 1 hidden layer. Intuitively, the planner encodes 'intent' into the state encoding h_t and the chosen action a_t, and the controller keeps going until the visual input I_t^n aligns with the intent of the planner.

Question Answering. After the agent decides to stop (or a max number of actions have been taken), the question answering module is executed to provide an answer based on the sequence of frames $I_1^1, \ldots, I_T^n$ the agent has observed throughout its trajectory. The answering module attends to each of the last five frame, computes an attention pooled visual encoding based on image-question similarity, combines these with an LSTM encoding of the question, and outputs a softmax over the space of 172 possible answers.

4.1. Imitation Learning and Reward Shaping

We employ a two-stage training process. First, the navigation and answering modules are independently trained using imitation/supervised learning on automatically generated expert demonstrations of navigation. Second, the entire architecture is jointly fine-tuned using policy gradients.

Independent Pretraining via Imitation Learning. Most questions that could be asked in EmbodiedQA do not have a natural 'correct' navigation required to answer them. As mentioned in Section 3.2, one virtue of EQA v1 questions is that they contain a single target queried object (<OBJ>). This allows us to use the shortest path from the agent's spawn location to the target as an expert demonstration.

The navigation module is trained to mimic the shortest path actions in a teacher forcing setting - *i.e.*, given the history encoding, question encoding, and the current frame, the model is trained to predict the action that would keep it on the shortest path. We use a cross-entropy loss and train the model for 15 epochs. We find that even in this imitation learning case, it is essential to train the navigator under a distance-based curriculum. In the first epoch, we backtrack 10 steps from the target along the shortest path and initialize the agent at this point with the full history of the trajectory from the spawned location. We step back an additional 10 steps at each successive training epoch. We train for 15 epochs total with batch size ranging from 5 to 20 questions (depending on path length due to memory limitations).

The question answering module is trained into predict the correct answer based on the question and the frames seen on the shortest path. We apply standard cross-entropy training over 50 epochs with a batch size of 20.

Target-aware Navigational Fine-tuning. While the navigation and answering modules that result from imitation learning perform well on their independent tasks, they are poorly suited to dealing with each other. Specifically, both modules are used to following the provided shortest path, but when in control the navigator may generalize poorly and provide the question answerer with unhelpful views of target (if it finds it at all). Rather than try to force the answering agent to provide correct answers from noisy or absent views, we freeze it and fine-tune the navigator.

We provide two types of reward signals to the navigator: the question answering accuracy achieved at the end of the navigation and a reward shaping [4] term that gives intermediate rewards for getting closer to the target. Specifically, the answering reward is 5 if the agent chooses to stop and answers correctly and 0 otherwise. The navigational reward for forward actions is 0.005 times the change in distance to target object (there is no reward or penalty for turning).

We train the agent with REINFORCE [36] policy gradients with a running average baseline for the answer reward. As in the imitation learning setting, we follow a curriculum of increasing distance between spawn and target locations.

Training details. All LSTMs are 2-layered with a 128-d hidden state. We use Adam [37] with a learning rate of 10^{-3}, and clamp gradients to $[-5, 5]$. We incrementally load environments in memory and use a batch size of 10 both during the imitation learning and REINFORCE fine-tuning stages. One forward step corresponds to at most 0.25 metres, and it takes 40 turns to turn 360°, *i.e.* one right or left turn action leads to 9° change in viewing angle. Backward and strafe motions are not allowed. We snap the continuous renderer space to a 1000×1000 grid to check for obstacles. Our entire codebase will be publicly available.

5. Experiments and Results

The ultimate goal of an EmbodiedQA agent is to answer questions accurately. However, it is important to disentangle success/failure at the intermediate task of navigation from the ultimate downstream task of question answering.

Question Answering Accuracy. Our agent (and all baselines) produce a probability distribution over 172 possible answers (colors, rooms, objects). We report the mean rank (**MR**) of the ground-truth answer in the answer list sorted by the agent's beliefs, where the mean is computed over all `test` questions and environments.

Navigation Accuracy. We evaluate navigation performance on EQA v1 by reporting the distance to the target object at navigation termination ($\mathbf{d_T}$), change in distance

		Navigation																		QA		
		$\mathbf{d_T}$			$\mathbf{d_\Delta}$			$\mathbf{d_{min}}$			$\%\mathbf{r_T}$			$\%\mathbf{r_{\hookleftarrow}}$			$\%\mathbf{stop}$			MR		
		T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}
Baselines	Reactive	2.09	2.72	3.14	-1.44	-1.09	-0.31	0.29	1.01	1.82	50%	49%	**47%**	52%	53%	48%	-	-	-	3.18	3.56	3.31
	LSTM	1.75	2.37	2.90	-1.10	-0.74	-0.07	0.34	1.06	2.05	55%	53%	44%	59%	57%	50%	80%	75%	80%	3.35	3.07	3.55
	Reactive+Q	1.58	2.27	2.89	-0.94	-0.63	-0.06	0.31	1.09	1.96	52%	51%	45%	55%	57%	**54%**	-	-	-	3.17	3.54	3.37
	LSTM+Q	1.13	2.23	2.89	-0.48	-0.59	-0.06	0.28	0.97	1.91	**63%**	53%	45%	64%	59%	**54%**	80%	71%	68%	3.11	3.39	3.31
Us	ACT+Q	**0.46**	**1.50**	**2.74**	**0.16**	**0.15**	**0.12**	0.42	1.42	2.63	58%	54%	45%	60%	56%	46%	**100%**	**100%**	**100%**	**3.09**	3.13	3.25
	ACT+Q-RL	1.67	2.19	2.86	-1.05	-0.52	0.01	**0.24**	**0.93**	**1.94**	57%	**56%**	45%	**65%**	**62%**	52%	32%	32%	24%	3.13	**2.99**	**3.22**
Oracle	HumanNav*	0.81	0.81	0.81	0.44	1.62	2.85	0.33	0.33	0.33	86%	86%	86%	87%	89%	89%	-	-	-	-	-	-
	ShortestPath+VQA	-	-	-	0.85	2.78	4.86	-	-	-	-	-	-	-	-	-	-	-	-	3.21	3.21	3.21

Table 1: Quantitative evaluation of EmbodiedQA agents on navigation and answering metrics for the EQA v1 test set. Ill-defined cells are marked with '-' because 1) reactive models don't have a stopping action, 2) humans pick a single answer from a drop-down list, so mean rank is not defined, 3) most distance metrics are trivially defined for shortest paths since they always end at the target object by design.

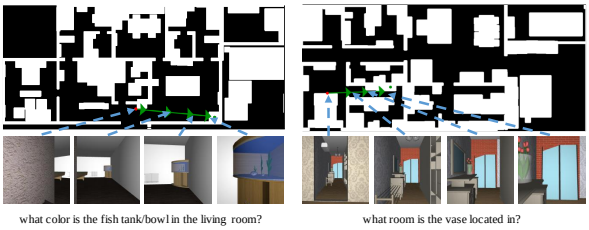


Figure 5: Sample trajectories from ACT+Q-RL agent projected on a floor plan (white areas are unoccupiable) and on-path egocentric views. The agent moves closer to already visible objects – potentially improving its perception of the objects. Note that the floor plan is shown only for illustration and not available to the agents.

to target from initial to final position ($\mathbf{d_\Delta}$), and the smallest distance to the target at any point in the episode ($\mathbf{d_{min}}$). All distances are measured in meters along the shortest path to the target. We also record the percentage of questions for which an agent either terminates in ($\%\mathbf{r_T}$) or ever enters ($\%\mathbf{r_{\hookleftarrow}}$) the room containing the target object(s). Finally, we also report the percent of episodes in which agents choose to terminate navigation and answer before reaching the maximum episode length ($\%\mathbf{stop}$). To sweep the difficulty of the task at test time, we spawn the agent 10, 30, or 50 actions away from the target and report each metric for T_{-10}, T_{-30}, T_{-50} settings.

Navigation Baselines. We compare our ACT navigator with a number of sophisticated baselines and ablations.

- **Reactive CNN.** This is a feedforward network that uses the last-n frames to predict the next action. We tried $n = \{1, 3, 5, 10\}$ and report $n = 5$, which worked best. Note that this is a target-agnostic baseline (*i.e.*, is not aware of the question). The purpose of this baseline is to check whether simply memorizing frames from training environments generalizes to test (it does not).
- **Reactive CNN+Question.** This combines the frame representation (as above) with an LSTM encoding of the question to predict the next action. This is similar to the approach of [18], with the difference that the goal is specified via a question encoding instead of a target image. Note that the action space for both the reactive baselines is {forward, turn-left, turn-right}, there is no stop token. At test time, the model is run for max no. of actions (= 100).
- **LSTM+Question.** The above two are memoryless navigators. This LSTM navigator takes as input the encodings of the question, current frame, and previous action, and predicts the next action. Note that these are identical inputs/outputs as our ACT navigator. The purpose of comparing to this ablation of our approach is to establish the benefit of our proposed planner-controller architecture. We also compare against an ablated version of this baseline without the question encoding as input (**LSTM**).

Navigation Oracles. We compare against two oracles:

- **HumanNav*** denotes goal-driven navigations by AMT workers remotely operating the agent (* denotes that data human studies were conducted on a subset set of `test`).
- **ShortestPaths+VQA** denotes the question answering performance achieved by our answering module when fed in shortest path at *test time*.

Table 1 shows the results of all baselines compared with our approach trained with just imitation learning (ACT+Q) and our approach fine-tuned with RL (ACT+Q-RL)[2]. We make a few key observations:

- **All baselines are poor navigators.** All baselines methods have *negative* $\mathbf{d_\Delta}$, *i.e.* they end up *farther* from the target than where they start. This confirms our intuition that EmbodiedQA is indeed a difficult problem.
- **Memory helps.** All models start equally far away from the target. Baselines augmented with memory (LSTM vs Reactive and LSTM-Q vs Reactive-Q) end closer to the target, *i.e.* achieve smaller $\mathbf{d_T}$, than those without.

[2] `youtube.com/watch?v=gVj-TeIJfrk` shows example navigation and answer predictions by our agent.

- **ACT Navigators performs best.** Our proposed navigator (ACT+Q) achieves the smallest distance to target at the end of navigation ($\mathbf{d_T}$), and the RL-finetuned navigator (ACT+Q-RL) achieves the highest answering accuracy.
- **RL agent overshoots.** Interestingly, we observe that while our RL-finetuned agent (ACT+Q-RL) gets closest to the target in its trajectory (*i.e.*, achieves least $\mathbf{d_{min}}$) and enters the target room most often (*i.e.*, achieves highest $\%\mathbf{r}_{\hookleftarrow}$), it does *not* end closest to the target (*i.e.*, does not achieve highest $\mathbf{d_\Delta}$). These statistics and our qualitative analysis suggests that this is because RL-finetuned agents learn to explore, with a lower stopping rate ($\%\mathbf{stop}$), and often overshoot the target. This is consistent with observations in literature [6]. In EmbodiedQA, this overshooting behavior does not hurt the question answering accuracy because the answering module can attend to frames along the trajectory. This behavior can be corrected by including a small negative reward for each action.
- **Shortest paths are not optimal for VQA.** A number of methods outperform ShortestPath+VQA in terms of answering accuracy. This is because while the shortest path clearly takes an agent to the target object, it may not provide the best vantage to answer the question. In future work, these may be improved by ray tracing methods to appropriately frame of the target object at termination.

6. Conclusion

We present Embodied Question Answering (EmbodiedQA) – a new AI task where an agent is spawned at a random location in a 3D environment and asked a question. In order to answer, the agent must first intelligently navigate to explore the environment, gather information through first-person (egocentric) vision, and then answer the question. We develop a novel neural hierarchical model that decomposes navigation into a 'planner' – that selects actions or a direction – and a 'controller' – that selects a velocity and executes the primitive actions a variable number of times – before returning control to the planner. We initialize the agent via imitation learning and fine-tune it using reinforcement learning for the goal of answering questions. We develop evaluation protocols for EmbodiedQA, and evaluate our agent in the House3D virtual environment. Additionally, we collect human demonstrations by connecting workers on Amazon Mechanical Turk to this environment to remotely control an embodied agents. All our code, data, and infrastructure will be made publicly available.

7. Acknowledgements

We are grateful to the developers of PyTorch [38] for building an excellent framework. We thank Yuxin Wu for help with the House3D environment. This work was funded in part by NSF CAREER awards to DB and DP, ONR YIP awards to DP and DB, ONR Grant N00014-14-1-0679 to DB, ONR Grant N00014-16-1-2713 to DP, an Allen Distinguished Investigator award to DP from the Paul G. Allen Family Foundation, Google Faculty Research Awards to DP and DB, Amazon Academic Research Awards to DP and DB, AWS in Education Research grant to DB, and NVIDIA GPU donations to DB. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government, or any sponsor.

Appendix Overview

This supplementary document is organized as follows:

- Sec. A presents the question-answer generation engine in detail, including functional programs associated with questions, and checks and balances in place to avoid ambiguities, biases, and redundancies.
- Sec. B describes the CNN models that serve as the vision module for our EmbodiedQA model. We describe the model architecture, along with the training details, quantitative as well as qualitative results.
- Sec. C describes the answering module in our agent.
- Sec. D reports machine question answering performance conditioned on human navigation paths (collected via human studies on Amazon Mechanical Turk).
- Finally, `youtube.com/watch?v=gVj-TeIJfrk` shows example navigation and answer predictions by our agent.

A. Question-Answer Generation Engine

Recall that each question in EQA is represented as a functional program that can be executed on the environment yielding an answer[3]. In this section, we describe this process in detail. In the descriptions that follow, an 'entity' can refer to either a queryable room or a queryable object from the House3D [8] environment.

Functional Forms of Questions. The functional programs are composed of elementary operations described below:

1. $select(entity)$: Fetches a list of entities from the environment. This operation is similar to the 'select' query in relational databases.
2. $unique(entity)$: Performs filtering to retain entities that are unique (*i.e.* occur exactly once). For example, calling $unique(rooms)$ on the set of rooms [*'living_room'*, *'bedroom'*, *'bedroom'*] for a given house will return [*'living_room'*].
3. $blacklist(template)$: This function operates on a list of object entities and filters out a pre-defined list of objects that are blacklisted for the given template. We do not ask questions of a given template type corresponding to any of the blacklisted objects. For example, if the blacklist contains the objects {*'column'*, *'range_hood'*, *'toy'*} and the objects list that the function receives is {*'piano'*, *'bed'*, *'column'*}, then the output of the $blacklist()$ function is: {*'piano'*, *'bed'*}
4. $query(template)$: This is used to generate the questions strings for the given template on the basis of the entities that it receives. For example, if $query(location)$ receives the following set of object entities as input: [*'piano'*, *'bed'*, *'television'*], it produces 3 question strings of the form: *what room is the <OBJ> located in?* where *<OBJ>* = {*'piano'*, *'bed'*, *'television'*}.
5. $relate()$: This elementary operation is used in the functional form for preposition questions. Given a list of object entities, it returns a subset of the pairs of objects that have a {*'on'*, *'above'*, *'under'*, *'below'*, *'next to'*} spatial relationship between them.
6. $distance()$: This elementary operation is used in the functional form for distance comparison questions. Given a list of object entities, it returns triplets of objects such that the first object is closer/farther to the anchor object than the second object.

Having described the elementary operations that make up our functional forms, the explanations of the functional forms for each question template is given below. We categorize the question types into 3 categories based on the objects and the rooms that they refer to.

1. **Unambiguous Object**: There are certain question types that inquire about an object that must be unique and unambiguous throughout the environment. Examples of such question types are `location` and `color`. For example, we should ask *'what room is the piano located in?'* if there is only a single instance of a *'piano'* in the environment. Hence, the functional forms for location and color have the following structure:

$$select(objects) \rightarrow unique(objects) \rightarrow query(location/color).$$

$select(objects)$ gets the list of all objects in the house and the $unique(objects)$ only retains objects that are unique, thereby ensuring unambiguity. The $query(template)$ function prepares the question string by filling in the slots in the template string.

2. **Unambiguous Room + Unambiguous Object**: In continuation of the above, there is another set of question types that talk about objects in rooms where in addition to the objects being unique, the rooms should also be unambiguous. Examples of such question types include `color_room`, `preposition`, and `distance`. The additional unambiguity constraint on the room is because the question *'what is next to the bathtub in the bathroom?'* would become ambiguous if there are two or more bathrooms in the house. The functional forms for such types are given by the following structure:

[3] or a response that the question is inapplicable (*e.g.* referring to objects not in the environment) or ambiguous (having multiple valid answers).

Template	Functional Form
location	$select(objects) \rightarrow unique(objects) \rightarrow blacklist(location) \rightarrow query(location)$
color	$select(objects) \rightarrow unique(objects) \rightarrow blacklist(color) \rightarrow query(color)$
color_room	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow blacklist(color) \rightarrow query(color\_room)$
preposition	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow blacklist(preposition) \rightarrow relate() \rightarrow query(preposition)$
existence	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow blacklist(existence) \rightarrow query(exist)$
logical	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow blacklist(existence) \rightarrow query(logical)$
count	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow blacklist(count) \rightarrow query(count)$
room_count	$select(room) \rightarrow query(room\_count)$
distance	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow blacklist(distance) \rightarrow distance() \rightarrow query(distance)$

Table 2: Functional forms of all question types in the EQA dataset

$$select(rooms) \rightarrow unique(rooms) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow query(template).$$

The first two operations in the sequence result in a list of unambiguous rooms whereas the next two result in a list of unambiguous objects in those rooms. Note that when $select(\cdot)$ appears as the first operation in the sequence (*i.e.*, select operates on an empty list), it is used to fetch the set of rooms or objects across the entire house. However, in this case, $select(object)$ operates on a set of rooms (the output of $select(rooms) \rightarrow unique(rooms) \rightarrow$), so it returns the set of objects found in those specific rooms (as opposed to fetching objects across all rooms in the house).

3. **Unambiguous Room**: The final set of question types are the ones where the rooms need to be unambiguous, but the objects in those rooms that are being referred to do not. Examples of such question types are: `existence`, `logical`, and `count`. It is evident that for asking about the existence of objects or while counting them, we do not require the object to have only a single instance. *'Is there a television in the living room?'* is a perfectly valid question, even if there are multiple televisions in the living room (provided that there is a single living room in the house). The template structure for this is a simplified version of (2):

$$select(rooms) \rightarrow unique(rooms) \rightarrow select(objects) \rightarrow query(template).$$

Note that we have dropped $unique(objects)$ from the sequence as we no longer require that condition to hold true.

See Table 2 for a complete list of question functional forms.

Checks and balances. Since one of our goals is to benchmark performance of our agents with human performance, we want to avoid questions that are cumbersome for a human to navigate for or to answer. Additionally, we would also like to have a balanced distribution of answers so that the agent is not able to simply exploit dataset biases and answer questions effectively without exploration. This section describes in detail the various checks that we have in place to ensure these properties.

1. **Entropy+Frequency-based Filtering**: It is important to ensure that the distribution of answers for a question is not too 'peaky', otherwise the mode guess from this distribution will do unreasonably well as a baseline. Thus, we compute the normalized entropy of the distribution of answers for a question. We drop questions where the normalized entropy is below 0.5. Further, we also drop questions that occur in less than 4 environments because the entropy counts for low-frequency questions are not reliable.

2. **Object Instance Count Threshold**: We do not ask counting questions (`room_count` and `count`) when the answer is greater than 5, as they are tedious for humans.

3. **Object Distance Threshold**: We consider triplets of objects within a room consisting of an anchor object, such that the difference of distances between two object-anchor pairs is at least 2 metres. This is to avoid ambiguity in *'closer'*/*'farther'* object distance comparison questions.

4. **Collapsing Object Labels**: Object types that are visually very similar (*e.g.* *'teapot'* and *'coffee_kettle'*) or semantically hierarchical in relation (*e.g.* *'bread'* and *'food'*) introduce unwanted ambiguity. In these cases we collapse the object labels to manually selected labels (*e.g.* (*'teapot'*, *'coffee_kettle'*) → *'kettle'* and (*'bread'*, *'food'*) → *'food'*).

5. **Blacklists**:
   - **Rooms**: Some question types in the EQA dataset have room names in the question string (e.g. `color_room`, `exist`, `logical`). We do not generate such questions for rooms that have obscure or esoteric names such as *'loggia'*, *'freight elevator'*, *'aeration' etc*. or names from which the room being referred might not be immediately obvious *e.g.* *'boiler room'*, *'entryway' etc*.
   - **Objects**: For each question template, we maintain a list of objects that are not to be included in questions. These are either tiny objects or whose name descriptions are too vague e.g. *'switch'* (too small), *'decoration'* (not descriptive enough), *'glass'* (transparent), *'household appliance'* (too vague). These blacklists are manually defined based on our experiences performing these tasks.

	Pixel Accuracy	Mean Pixel Accuracy	Mean IOU
single	0.780	0.246	0.163
hybrid	0.755	0.254	0.166

(a) Segmentation

	Smooth-ℓ_1
single	0.003
hybrid	0.005

(b) Depth

	Smooth-ℓ_1
single	0.003
hybrid	0.003

(c) Autoencoder

Table 3: Quantitative results for the autoencoder, depth estimation, and semantic segmentation heads of our multi-task perception network. All metrics are reported on a held out validation set.

B. CNN Training Details

The CNN comprising the visual system for our EmbodiedQA agents is trained under a multi-task pixel-to-pixel prediction framework. We have an encoder network that transforms the egocentric RGB image from the House3D renderer [8] to a fixed-size representation. We have 3 decoding heads that predict 1) original RGB values (*i.e.* an autoencoder), 2) semantic class, and 3) depth for each pixel. The information regarding semantic class and depth of every pixel is available from the renderer. The range of depth values for every pixel lies in the range $[0, 1]$ and the segmentation is done over 191 classes.

Architecture. The encoder network has 4 Conv blocks, comprising of 5×5 Conv filters, ReLU, BatchNorm and 2×2 MaxPool. Each of the 3 decoder branches of our network upsample the encoder output to the spatial size of the original input image. The number of channels in the output of the decoder depends on the task head – 191, 1 and 3 for the semantic segmentation, depth and autoencoder branches respectively. The upsampling is done using bilinear interpolation. The architecture also has skip connections from the 2nd and the 3rd Conv layers.

Training Details. We use cross-entropy loss to train the segmentation branch of our hybrid network. The depth and autoencoder branches are trained using the Smooth-ℓ_1 loss. The total loss is a linear combination of the 3 losses, given by $overall\_loss = seg\_loss + 10 \times depth\_loss + 10 \times reconstruction\_loss$. We use the Adam optimizer with a learning rate of 10^{-3} and a batch size of 20. The hybrid network is trained for a total of 5 epochs on a dataset of $100k$ RGB training images from the renderer.

Quantitative Results. Table 3 shows some quantitative results. For each of the 3 different decoding heads of our multi-task CNN, we report results on the validation set for two settings - when the network is trained for all tasks at once (hybrid) or each task independently (single). For segmentation, we report the overall pixel accuracy, mean pixel accuracy (averaged over all 191 classes) and the mean IOU (intersection over union). For depth and autoencoder, we report the Smooth-ℓ_1 on the validation set.

Qualitative Results. Some qualitative results on images from the validation set for segmentation, depth prediction and autoencoder reconstruction are shown in Figure 6.

C. Question Answering Module

The question answering module predicts the agents' beliefs over the answer given the agent's navigation. It first encodes the question with an LSTM, last five frames of the navigation each with a CNN, and then computes dot product attention over the five frames to pick the most relevant ones. Next, it combines attention-weighted sum of image features with the question encoding to predict a softmax distribution over answers. See Fig. 7.

D. Human Navigation + Machine QA

In order to contrast human and shortest-path navigations with respect to question answering, we evaluate our QA model on the last 5 frames of human navigations collected through our Amazon Mechanical Turk interface. We find the mean rank of the ground truth answer to be 3.51 for this setting (compared to 3.26 when computed from shortest-paths). We attribute this difference primarily to a mismatch between the QA system training on shortest paths and testing on human navigations. While the shortest paths typically end with the object of interest filling the majority of the view, humans tend to stop earlier as soon as the correct answer can be discerned. As such, human views tend to be more cluttered and pose a more difficult task for the QA module. Fig. 10 highlights this difference by contrasting the last 5 frames from human and shortest-path navigations across three questions and environments.

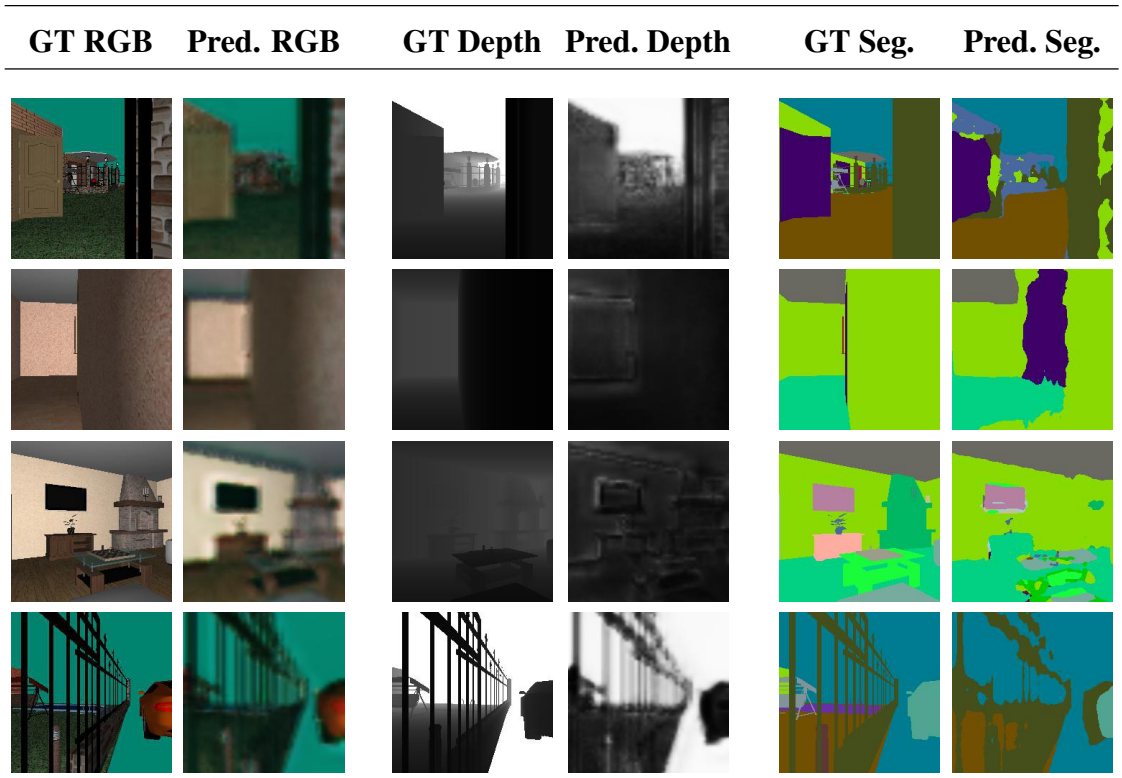


Figure 6: Some qualitative results from the hybrid CNN. Each row represents an input image. For every input RGB image, we show the reconstruction from the autoencoder head, ground truth depth, predicted depth as well as ground truth segmentation and predicted segmentation maps.

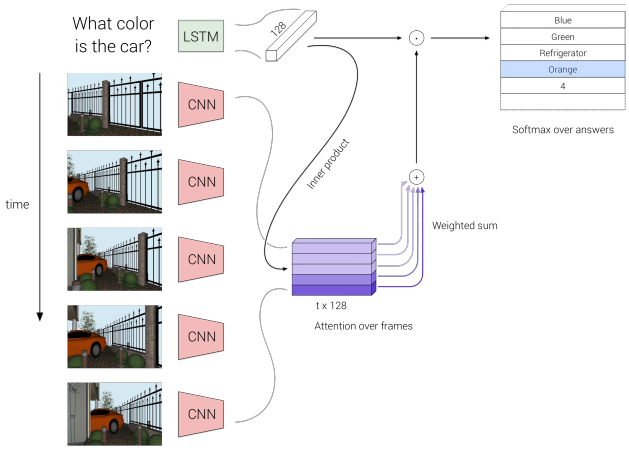


Figure 7: Conditioned on the navigation frames and question, the question answering module computes dot product attention over the last five frames, and combines attention-weighted combination of image features with question encoding to predict the answer.

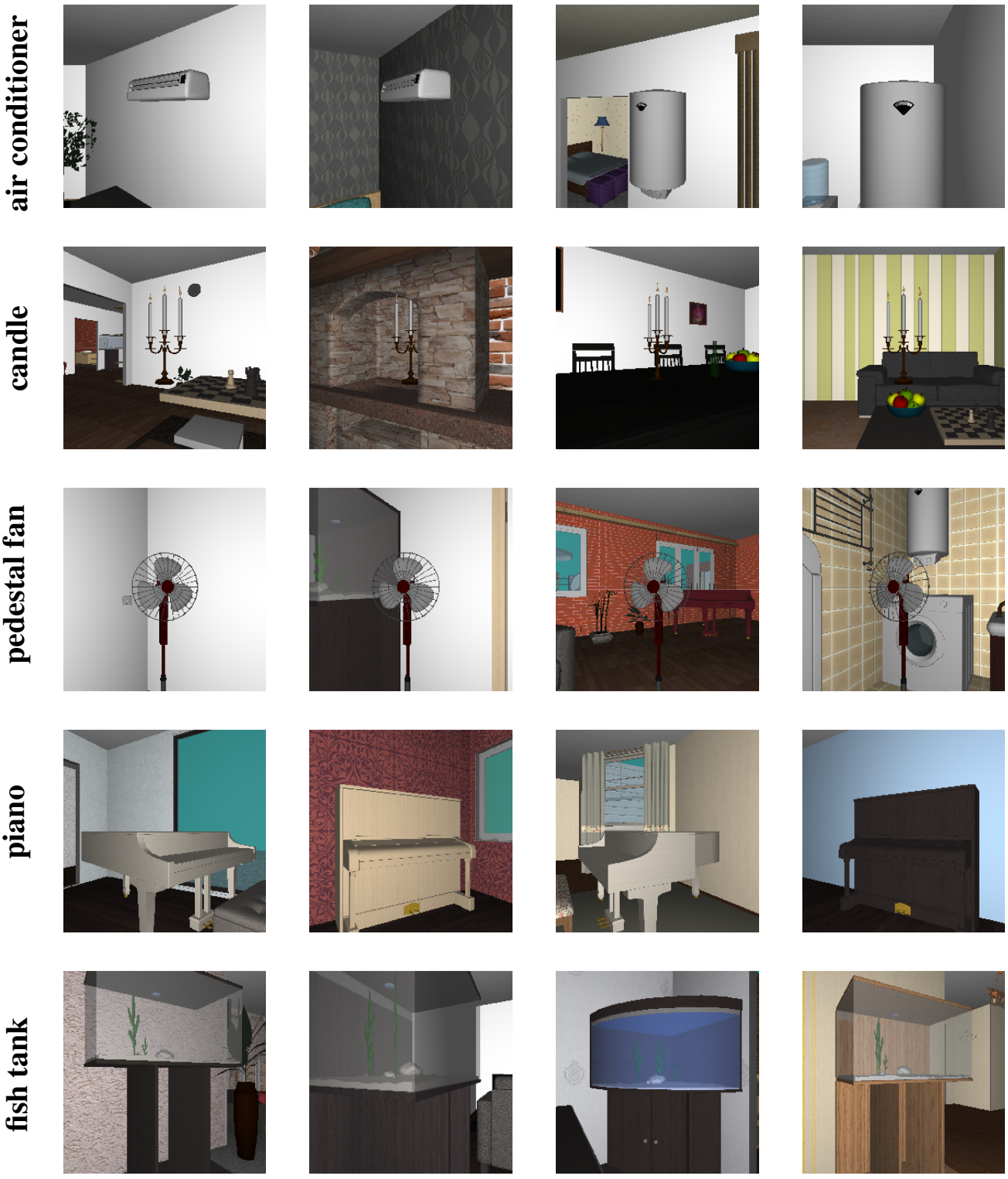
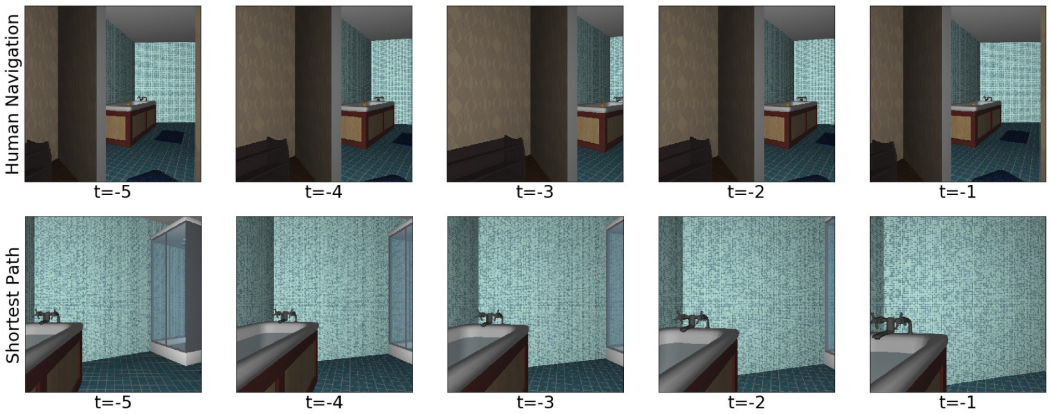
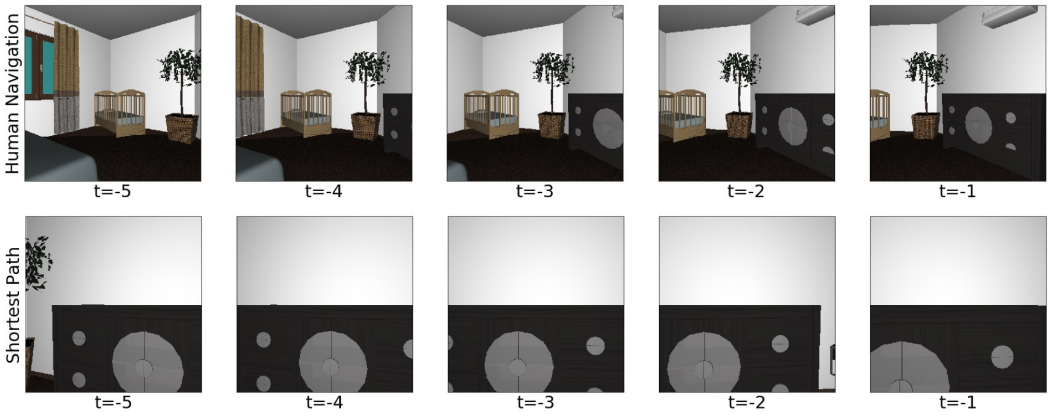


Figure 8: Visualizations of queryable objects from the House3D renderer. Notice that instances within the same class differ significantly in shape, size, and color.

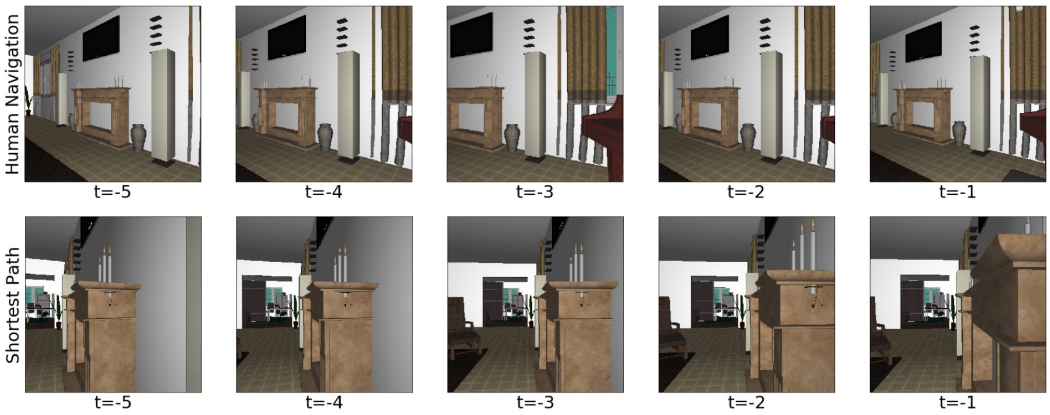
Figure 9: Visualizations of queryable rooms from the House3D renderer.



Q: What color is the bathtub?

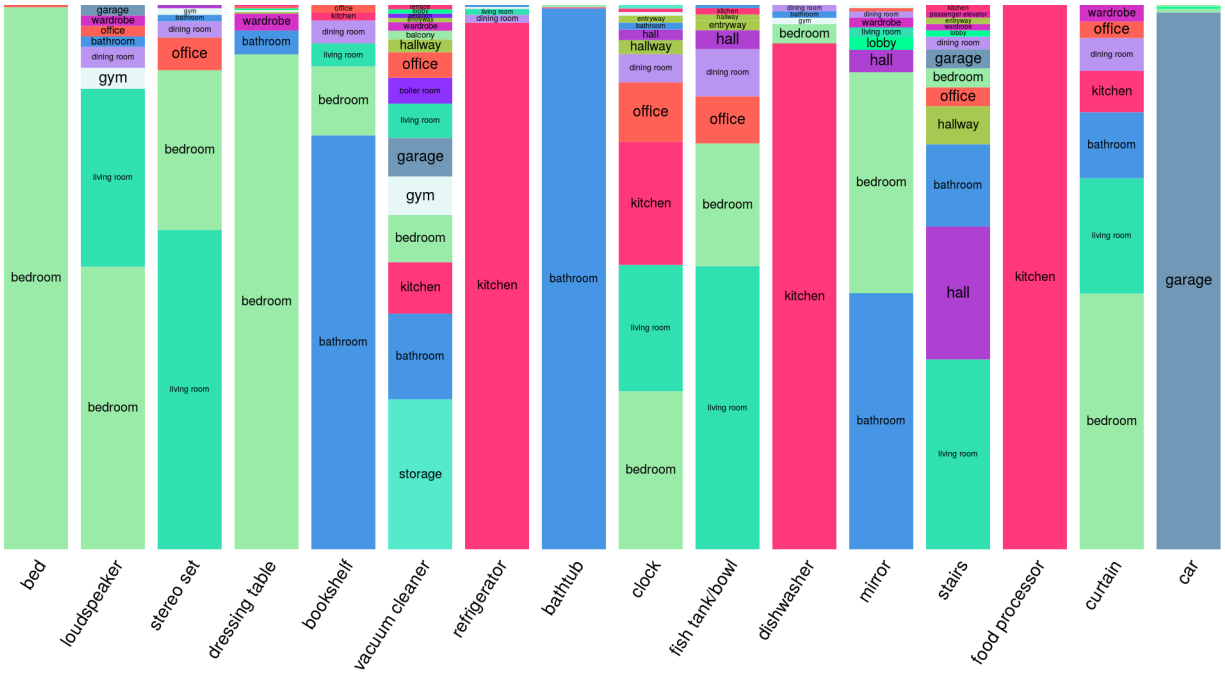


Q: What color is the dresser?

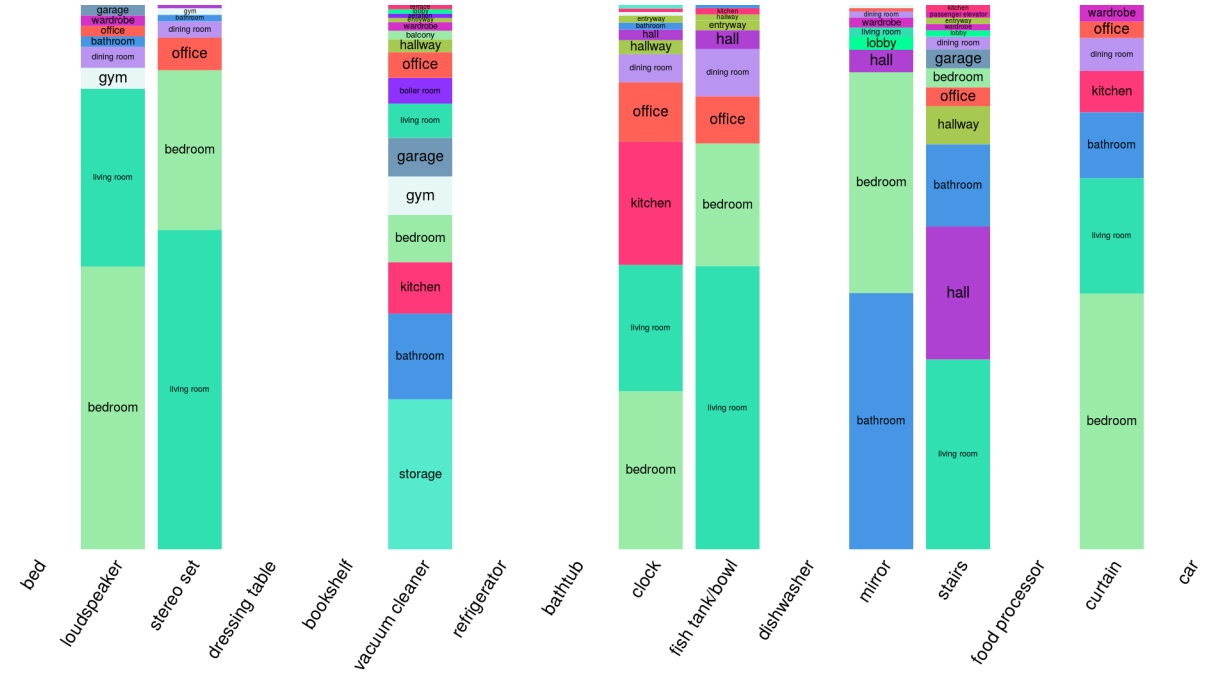


Q: What color is the fireplace in the living room?

Figure 10: Examples of last five frames from human navigation vs. shortest path.

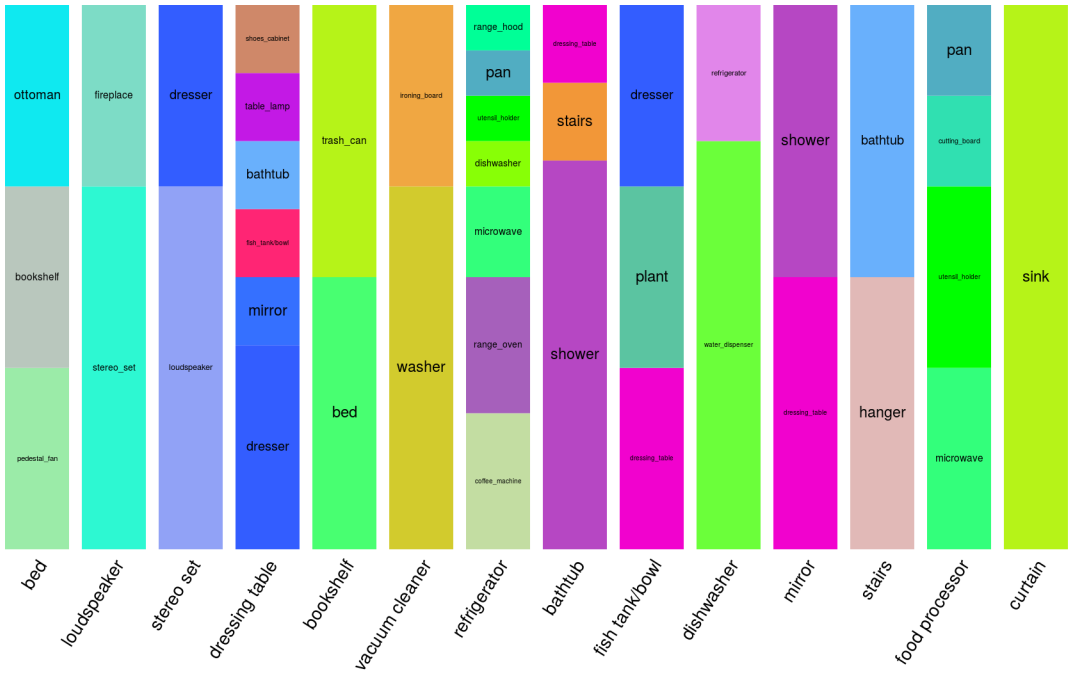


(a) `location` questions before entropy+count based filtering

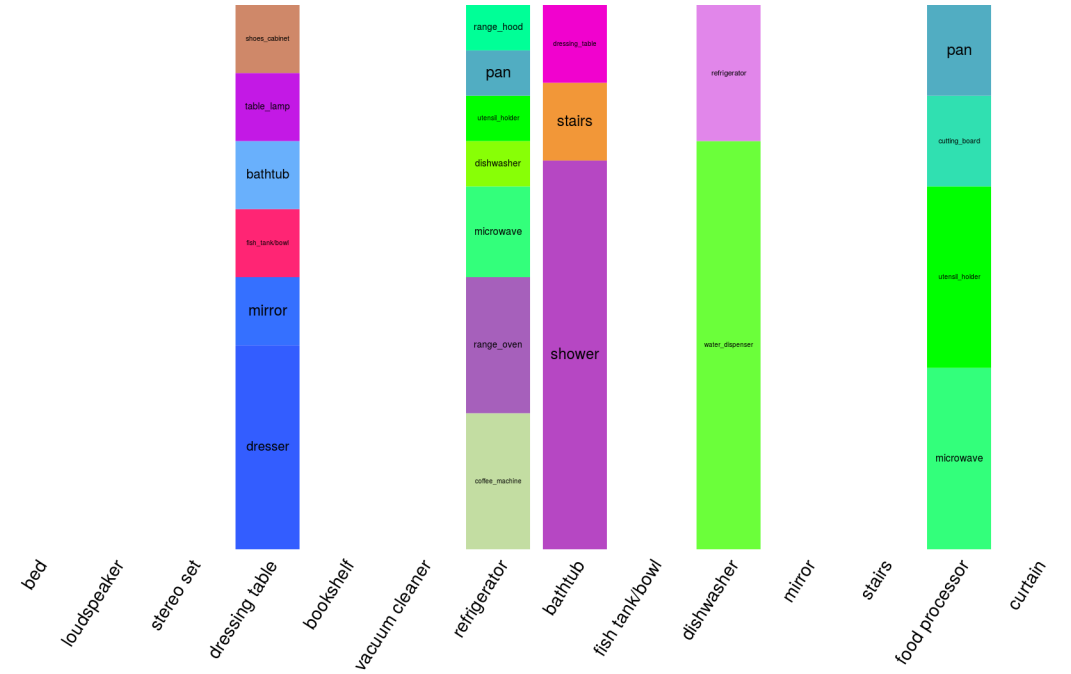


(b) `location` questions after entropy+count based filtering

Figure 11: The answer distribution for `location` template questions. Each bar represents a question of the form *'what room is the <OBJ> located in?'* and shows a distribution over the answers across different environments. The blank spaces in 11b represent the questions that get pruned out as a result of the entropy+count based filtering.

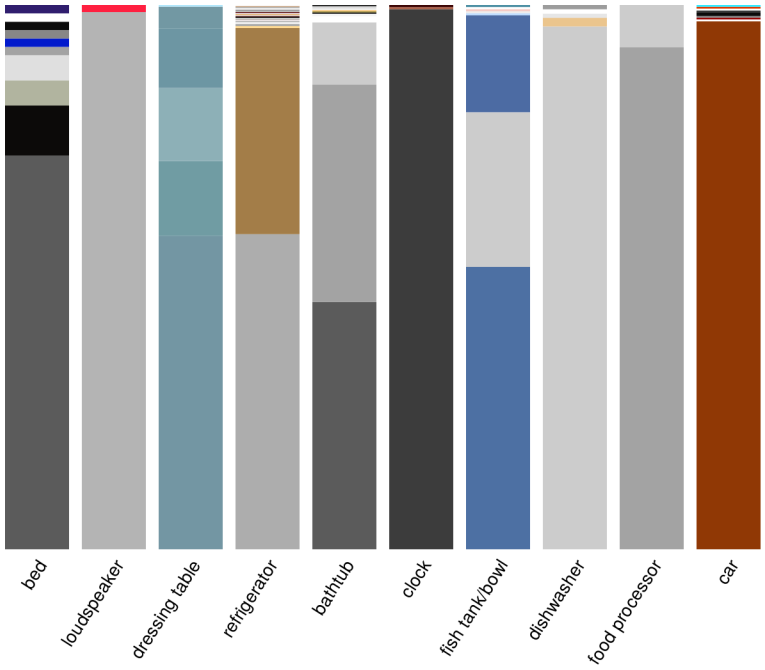


(a) `preposition` questions before entropy+count based filtering

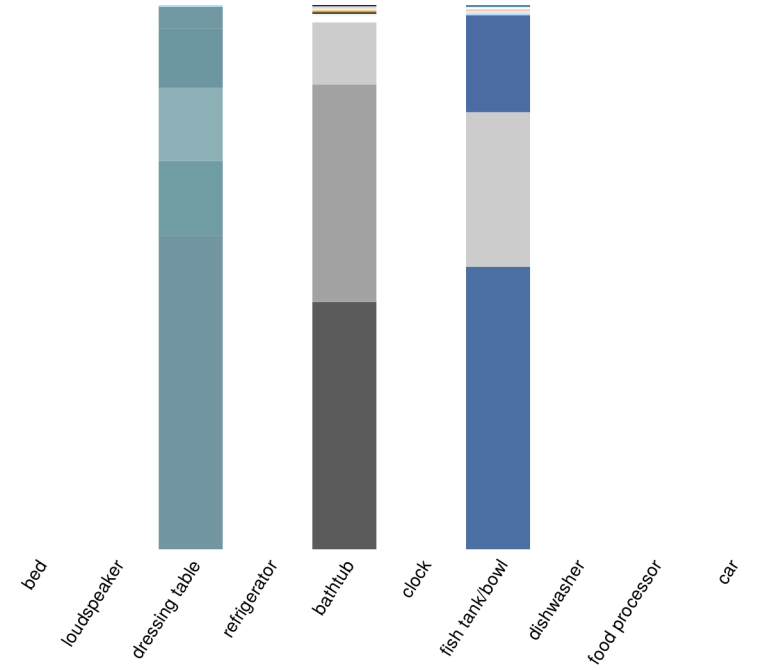


(b) `preposition` questions after entropy+count based filtering

Figure 12: The answer distribution for `preposition` template questions. Each bar represents a question of the form *'what is next to the <OBJ>?'* and shows a distribution over the answers across different environments. The blank spaces in 12b represent the questions that get pruned out as a result of the entropy+count based filtering.



(a) `color` questions before entropy+count based filtering



(b) `color` questions after entropy+count based filtering

Figure 13: The answer distribution for `color` template questions. Each bar represents a question of the form *'what color is the <OBJ>?'* and shows a distribution over the answers across different environments (the color of each section on a bar denotes the possible answers). The blank spaces in 13b represent the questions that get pruned out as a result of the entropy+count based filtering.

References

[1] L. Smith and M. Gasser, "The development of embodied cognition: six lessons from babies.," *Artificial life*, vol. 11, no. 1-2, 2005. 1

[2] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and D. Batra, "Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions?," in *EMNLP*, 2016. 2

[3] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *CVPR*, 2017. 2

[4] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, 1999. 2, 7

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *JMLR*, vol. 17, pp. 1334–1373, Jan. 2016. 2

[6] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *EMNLP*, 2017. 2, 9

[7] A. Graves, "Adaptive computation time for recurrent neural networks," *CoRR*, vol. abs/1603.08983, 2016. 2, 3, 6

[8] Anonymous, "Building generalizable agents with a realistic and rich 3d environments," in *ICLR*, 2018. 2, 3, 4, 10, 12

[9] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *CVPR*, 2017. 2, 4

[10] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and Yang: Balancing and Answering Binary Visual Questions," in *CVPR*, 2016. 2

[11] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual Question Answering," in *ICCV*, 2015. 3

[12] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," in *EMNLP*, 2016. 3

[13] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," *arXiv preprint arXiv:1707.07998*, 2017. 3

[14] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "MovieQA: Understanding Stories in Movies through Question-Answering," *CVPR*, 2016. 3

[15] Y. Jang, Y. Song, Y. Yu, Y. Kim, and G. Kim, "TGIF-QA: toward spatio-temporal reasoning in visual question answering," in *CVPR*, 2017. 3

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005. 3

[17] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," *arXiv preprint arXiv:1706.07230*, 2017. 3

[18] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*, 2017. 3, 8

[19] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual Semantic Planning using Deep Successor Representations," in *ICCV*, 2017. 3

[20] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *CVPR*, 2017. 3

[21] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. Czarnecki, M. Jaderberg, D. Teplyashin, *et al.*, "Grounded language learning in a simulated 3d world," *arXiv preprint arXiv:1706.06551*, 2017. 3

[22] S. Brahmbhatt and J. Hays, "DeepNav: Learning to Navigate Large Cities," in *CVPR*, 2017. 3

[23] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *ICML*, 2017. 3, 6

[24] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *ICRA*, 2017. 3

[25] T. Winograd, "Understanding natural language," *Cognitive Psychology*, vol. 3, no. 1, pp. 1 – 191, 1972. 3

[26] M. Denil, S. G. Colmenarejo, S. Cabi, D. Saxton, and N. de Freitas, "Programmable agents," *arXiv preprint arXiv:1706.06383*, 2017. 3, 6

[27] H. Yu, H. Zhang, and W. Xu, "A deep compositional framework for human-like language acquisition in virtual environment," *arXiv preprint arXiv:1703.09831*, 2017. 3, 6

[28] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proceedings of the 34th International Conference on Machine Learning*, 2017. 3, 6

[29] D. K. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *EMNLP*, 2017. 3, 6

[30] S. I. Wang, P. Liang, and C. D. Manning, "Learning language games through interaction," in *ACL*, 2016. 3, 6

[31] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen, "Deepmind lab," *CoRR*, vol. abs/1612.03801, 2016. 3

[32] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, 2017. 3

[33] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in *AAAI*, 2017. 3, 6

[34] "Planner5d." https://planner5d.com/. 4

[35] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning," in *CVPR*, 2017. 4

[36] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992. 7

[37] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015. 7

[38] "PyTorch." http://pytorch.org/. 9