

Embodied Question Answering

Abhishek Das^{1*}, Samyak Datta¹, Georgia Gkioxari², Stefan Lee¹, Devi Parikh^{2,1}, Dhruv Batra^{2,1}

¹Georgia Institute of Technology, ²Facebook 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 necessary visual information through first-person (egocentric) vision, and then answer the question (‘orange’).

EmbodiedQA requires a range of AI skills – language understanding, visual recognition, active perception, goal-driven navigation, commonsense reasoning, long-term memory, and grounding language into actions. In this work, we develop a dataset of questions and answers in House3D environments [1], evaluation metrics, and a hierarchical model trained with imitation and reinforcement learning.

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 [2]

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

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

with a dataset of questions in virtual environments, evaluation metrics, and a deep reinforcement learning (RL) model.

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 egocentric vision and can perform a few atomic actions (move-forward, turn, strafe, etc.). The goal of the agent is to intelligently navigate the environment and gather visual information necessary for answering the question.

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

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 perceives. The agent must learn to map its visual input to the correct actions 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.

Commonsense 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? Where is the garage with respect to me?*) similar to how humans navigate in an unfamiliar house (*The car is probably in the garage outside, so I should find an exit door*).

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 [3, 4]. In EmbodiedQA, we take a goal-driven view of grounding – our agent grounds a question not into pixels but into a sequence of actions (*‘garage’ means to navigate towards the exterior of the house 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 (without double/multiple counting), 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, ‘4’) – 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 failure, and when things do go wrong, it’s difficult for the agent to know why – was the question misunderstood? Or can the agent not recognize chairs? Or did the agent navigate incorrectly? Or was the counting wrong?

As a first step in this challenging space, we judiciously scope out a problem space – environments, question types, and learning paradigm – that allows us to augment sparse RL rewards with imitation learning (showing the agent expert trajectories) and reward shaping [5] (giving intermediate ‘closer/farther’ navigation rewards). Specifically, our approach follows the recent paradigm from robotics and deep RL [6, 7] – the training environments are sufficiently *instrumented*, and provide access to the agent location, RGB, depth & semantic annotations of the visual environment, and allow for computing obstacle-avoiding shortest navigable 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, no hand-coded knowledge about the environment or task, and no pre-processing phase for the agent to build a map of the environment. The agent in its entirety – vision, language, navigation, answering – is trained from raw sensory input (pixels and words) to goal-driven multi-room navigation to visual question answering!

Contributions. We make the following contributions:

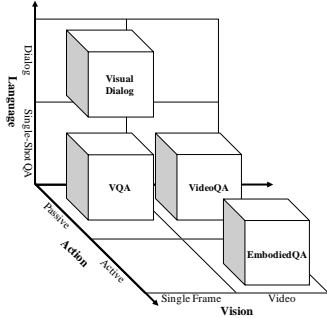
- We propose a new AI task: EmbodiedQA, where an agent is spawned in an environment and must intelligently navigate from egocentric vision to gather the necessary information to answer questions about the environment.
- We introduce a hierarchical navigation module that decomposes navigation into a ‘planner’ that selects actions, and a ‘controller’ that executes these primitive actions. When the agent decides it has seen the required visual information, it stops navigating and outputs an answer.
- We initialize our agents with imitation learning and show that agents can answer questions more accurately after fine-tuning with RL – that is, when allowed to control their own navigation *for the explicit purpose* of answering questions accurately. Unlike some prior work, we explicitly test *generalization to unseen environments*.
- We evaluate our agents in House3D [1], a rich, interactive 3D environment based on human-designed indoor scenes from SUNCG [8]. These diverse, simulated environments enable us to test generalization of our agent across floor-plans, object & room configurations – without safety, privacy, expense concerns inherent to real robotic platforms.
- We introduce EQA, a 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 enables us to control the distribution of question-types and answers, deter algorithms from exploiting dataset bias [4, 9], and provide fine-grained breakdown of performance by skill.
- We integrated House3D with Amazon Mechanical Turk (AMT), allowing humans to *remotely operate the agent in real time*, and collected expert demonstrations of question-guided navigation for EmbodiedQA that serve as a benchmark to compare our proposed and other future models.

All our code and data will be made publicly available¹.

¹embodiedqa.org

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 [10–14] 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. While the task is significantly harder than VQA (*i.e.* most random paths are useless), EmbodiedQA agents have the flexibility to avoid confusing viewpoints and intelligently seek visual input to 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 [15] 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 [16–22]. 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 [18, 19] (thus training a separate policy for each goal/reward), or explicitly by conditioning on goal state representations [23] like images of target objects [17, 24]. 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 recent work, Hermann *et al.* [20] and Chaplot *et al.* [16] both develop embodied agents in simple game-like environments to understand simple ‘go to X’ or ‘pick up X’ style commands where X could specify an object (and possibly some of its attributes). Similarly, Oh *et al.* [22] develop agents in a simple maze world and task them to follow a series of templated instructions. In contrast, 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 related to actions than questions presented in EQA.

Concurrent with our work, Anderson *et al.* [31] proposed goal-driven navigation from natural language instructions in real 3D environments, and Gordon *et al.* [32] studied the task of EmbodiedQA in the AI2-THOR environment [33].

Interactive Environments. There are several 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 [34] or Doom [16]), to more complex, realistic environments (*e.g.* AI2-THOR [18], Matterport3D [31], Stanford 2D-3D-S [35]). 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 [1] environment as it strikes a useful middle-ground between being sufficiently realistic and providing access to a large and diverse set of room layouts and object categories. See Sec. 3.1 for details.

Hierarchical Agents. We model our EmbodiedQA agents as deep hierarchical networks that decompose the overall control problem such that a higher-level planner invokes lower-level controls to issue primitive actions. Hierarchical modeling has recently shown promise in deep reinforcement learning [22, 28, 36]. Our model also draws inspiration from Graves’ work on Adaptive Computation Time [37].

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

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	bath tub	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						

(a) Sample environments [1, 8]

(b) Queryable rooms

(c) Queryable objects

Figure 2: The EQA dataset is built on a subset of House3D environments [1] and objects from the SUNCG [8] dataset. We show (a) top-down views of sample environments, and the (b) queryable rooms and (c) queryable objects that questions in the EQA dataset refer to.

3. EQA Dataset: Questions In Environments

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

3.1. House3D: Simulated 3D Environments

We instantiate EmbodiedQA in House3D [1], a recently introduced rich, simulated environment based on 3D indoor scenes from the SUNCG dataset [8]. Concretely, SUNCG consists of synthetic 3D scenes with realistic room and furniture layouts, manually designed and crowdsourced using an online interior design interface (Planner5D [38]). Each scene was further verified as realistic by majority vote of 3 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 simulated environments, where an agent (approximated as a 1 meter high cylinder) may navigate under simple physical constraints (not being able to pass through walls/objects). Fig. 2a shows top-down views of sample environments. Full details can be found in [1].

We build EQA on a pruned subset of environments from House3D. First, we only consider environments where all 3 SUNCG annotators consider the scene layout realistic. Next, we filter out atypical environments such as those lacking ‘ground’ or those too small/large (only keeping houses with an internal area of 300-800m² covering at least 1/3 the total ground area). Finally, we require houses to contain at least one kitchen, living room, dining room, and bedroom.

3.2. Question-Answer Generation

We would like to pose questions that test the agent’s ability to ground language, use common sense, reason visually, and navigate unseen environments. For *e.g.*, answering ‘*What color is the car?*’ requires grounding the symbol ‘*car*’, reasoning that cars are typically outside, navigating outside, exploring to find the car, and visually inspecting its color.

We draw inspiration from the CLEVR [39] dataset, and programmatically generate a dataset (EQA) of grounded questions and answers. This gives us the ability to carefully control the distribution of question-types and answers in the dataset, and deter algorithms from exploiting dataset bias.

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

Questions as Functional Programs. Each EQA question is represented as a functional program that can be executed on the environment yielding an answer. These programs are composed of a small set of elementary operations (*e.g.* `select()`, `singleton()`, `query()`, *etc.*) that operate on sets of room or object annotations and can be arbitrarily combined. 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 `location`:

`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) Ñ singleton(objects) Ñ query(location)`

The first function, `select(objects)`, retrieves all the objects from the environment. The second, `singleton(objects)`, retains objects that occur once in the entire house. Finally, `query(location)`, generates a question (by filling in the appropriate template) for each such object. Note that the `singleton(objects)` operation 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 located in?*’ is ambiguous, with potentially two different answers depending on which instance is being referred to.

Question Types. Overall, we have the following question types and associated templates in the EQA dataset:

EQA v1	location:	<i>'What room is the <OBJ> located in?'</i>
	color:	<i>'What color is the <OBJ>?'</i>
	color_room:	<i>'What color is the <OBJ> in the <ROOM>?'</i>
	preposition:	<i>'What is <on/above/below/next-to> the <OBJ> in the <ROOM>?'</i>
	existence:	<i>'Is there a <OBJ> in the <ROOM>?'</i>
	logical:	<i>'Is there a(n) <OBJ1> and a(n) <OBJ2> in the <ROOM>?'</i>
	count:	<i>'How many <OBS> in the <ROOM>?'</i>
	room_count:	<i>'How many <ROOMS> in the house?'</i>
	distance:	<i>'Is the <OBJ1> closer to the <OBJ2> than to the <OBJ3> in the <ROOM>?'</i>

These questions test a diverse set of agent abilities including object detection (existence), scene recognition (location), counting (count), spatial reasoning (preposition), color recognition (color), and logic (logical). Moreover, many of these questions require multiple skills – e.g., answering a distance question requires recognizing the room and objects as well as reasoning about their spatial relationships. Further, to do so, the agent must navigate the environment to find the room, then look around to find the objects, and remember their positions through time (if they are not simultaneously visible).

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). Finally, we stress that EQA is not a static dataset, but rather a test for a curriculum of capabilities that we would like to achieve in embodied communicating agents.

Question-Answer Generation and Dataset Bias. In theory, we now have the ability to automatically generate all valid questions with associated answers by executing the functional programs on the environment’s annotations provided by SUNCG. However, careful consideration is needed to ensure the developed dataset is balanced over answers.

For each generated question (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, to benchmark performance of agents vs. humans on EQA, it is important that questions are not tedious for humans to answer. We do not ask count questions for objects with high counts (> 5) or distance questions for object triplets with imperceptible distance differences. We pick these thresholds and room/object blacklists manually.

Complete discussion of the question templates, functional programs, elementary operations, answer distributions, and various checks and balances is available in the supplement.

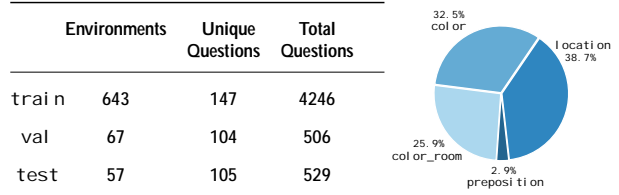


Figure 3: Overview of the EQA v1 dataset including dataset split statistics (left) and question type breakdown (right).

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 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 question generation engine publicly available.

4. A Hierarchical Model for EmbodiedQA

We now introduce our neural architecture for EmbodiedQA. 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, rooms, objects), 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). Our navigation module is inspired by the idea of Adaptive Computation Time (ACT) RNNs by Graves [37], which is an elegant approach for allowing RNNs to learn how many computational steps to take between receiving an input and emitting an output. We make use of this idea in

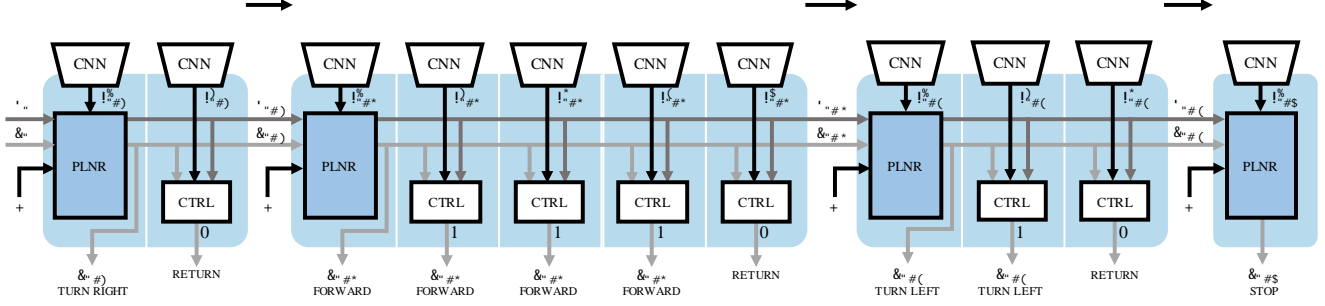


Figure 4: Our PACMAN navigator decomposes navigation into a planner and a controller. The planner selects actions and the controller executes these actions a variable number of times. This enables the planner to operate on shorter timescales, strengthening gradient flows.

our ‘planner-controller’ navigation module (PACMAN) to cleanly separate the decision between – direction (where to move, decided by a ‘planner’) and velocity (how far to move in one timestep, decided by a ‘controller’). PACMAN is illustrated in Fig. 4. We describe the different modules 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 to 5 Conv, BatchNorm, ReLU, 2×2 MaxPoolu 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 a shared base encoder network, we train multiple decoder heads for 1) RGB reconstruction, 2) semantic segmentation, and 3) depth estimation (annotations for which are available from the House3D renderer). See the supplementary material for model and training details.

Language. Questions are encoded with 2-layer LSTMs with 128-d hidden states. While LSTMs may be overkill for the simple questions in EQA v1, it gives us the flexibility to expand to human-written questions or more complex question templates in future. 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. As mentioned earlier, our PACMAN navigator decomposes navigation into a ‘planner’, that selects actions (forward, turn-left, turn-right, stop), 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 [22, 28, 36]. This also enables the planner to operate on shorter timescales, strengthening long-term gradient flows.

Formally, let $t = 1, 2, \dots, T$ denote planner timesteps, and $n = 0, 1, 2, \dots, N_{ptq}$ denote the variable number of controller timesteps. Let I_t^n denote the encoding of the observed image at t -th planner-time and n -th controller-time. The planner is instantiated as an LSTM. Thus, it maintains a hidden state h^t (updated only at planner timesteps), and samples action $a_t \in \{P, F, T, S\}$ forward, turn-right, turn-left, stop:

$$a_t, h_t \in \text{PLNR}(h_{t-1}, I_t^0, Q, a_{t-1}), \quad (1)$$

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.*

$$c_t^n, 1 \leq n \leq N_{ptq} \in \text{CTRL}(h_t, a_t, I_t^n) \quad (2)$$

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-time 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 h_t and a_t , and the controller keeps going until the visual input I_t^n aligns with the intent of the planner.

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 bin the continuous renderer space to a 1000×1000 grid world to check for obstacles.

Question Answering. After the agent decides to stop or a max number of actions ($= 100$) have been taken, the question answering module is executed to provide an answer based on the sequence of frames I_1^1, \dots, I_T^N the agent has observed. The answering module computes image-question similarity for each of the last 5 frames via a dot product between (a) image features (passed through an fc-layer to align with question features) and (b) question encoding Q . These I-Q similarities are converted into attention weights via a softmax, and the attention-weighted image features are combined with Q (via concatenation) and passed through a softmax classifier to predict a distribution over 172 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 navigation architecture is 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 Sec. 3.2, one virtue of EQA v1 questions is that they contain a single target queried object ($\langle \text{OBJ} \rangle$). This lets us use shortest paths from the agent’s spawn location to the target as expert demonstrations for imitation learning.

The navigation module is trained to mimic the shortest path in a teacher-forcing setting - *i.e.*, given the navigation history encoded in the hidden state of the planner, 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 model 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 full history of the trajectory from spawned location. We step back an additional 10 steps at each successive epoch. We train for 15 epochs total with batch size ranging from 5 to 20 (depending on path length due to memory limitations).

The answering module is trained to predict the correct answer from the question and shortest path navigation frames, with cross-entropy loss for 50 epochs and batch size of 20.

Target-aware Navigational Fine-tuning. While the navigation and answering modules that result from imitation learning perform well independently, they are poorly suited to work well with each other. Specifically, both modules are used to following the shortest path, but when in control, the navigator may generalize poorly *i.e.* go off the shortest path and still be required to decide which action to take next, a state it did not encounter during teacher-forced training. This may consequently provide unhelpful views of the target to the answering module. Thus, rather than force the answerer to provide correct answers based on noisy or absent views, we freeze it and fine-tune just the navigator.

We provide two types of rewards to the navigator: the question answering accuracy achieved at the end of the navigation and a reward shaping [5] 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 reduction in navigable distance to target (there is no reward or penalty for turning).

We train with REINFORCE [40] with a running average of

Figure 5: Sample trajectories from PACMAN-RL 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.

the reward as baseline. As in imitation learning, we follow a curriculum of distance between spawn and target locations.

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 downstream task of question answering.

Question Answering Accuracy. Our agent (and all baselines) produces a probability distribution over 172 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, computed over test questions from EQA.

Navigation Accuracy. We evaluate navigation performance by reporting the distance to the target object at navigation termination $pd_{T,q}$, change in distance to target from initial to final position $pd_{I,q}$, and the smallest distance to the target at any point in the episode $pd_{min,q}$. 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 $p\%r_{T,q}$ or ever enters $p\%r_{eq}$ the room containing the target object(s). Finally, we also report the percentage of episodes in which agents choose to terminate navigation and answer before reaching the maximum episode length $p\%stop_q$. To sweep over the difficulty of the task, we spawn the agent 10, 30, or 50 actions away from the target and report each metric for $T=10, T=30, T=50$.

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

- **Reactive CNN.** This is a feed-forward network that uses concatenated features from the last- n frames to predict the next action. We tried $n \in \{1, 3, 5, 10\}$ and report $n = 5$, which worked best. This is a target-agnostic baseline (*i.e.*, not aware of the question). The purpose of this baseline is to check whether 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

		Navigation															QA					
		d_T			d			d_{min}			$\%r_T$			$\%r_e$			$\%stop$			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	PACMAN+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
	PACMAN-RL+Q	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. III-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.

approach of [17], 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 PACMAN navigator. The purpose of comparing to this ablation is to establish the benefit of our proposed hierarchical 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 human studies were conducted on a subset of test).
- **ShortestPaths+VQA** denotes QA performance when the answerer is fed in shortest paths at *test time*. This will have the same mean rank across T_{-10} , T_{-30} , T_{-50} by design, since the answering module sees identical frames.

Tab. 1 shows results of baselines compared with our model trained with imitation learning (PACMAN+Q) and fine-tuned with RL (PACMAN-RL+Q)². A few observations:

- **All baselines are poor navigators.** All baselines methods have *negative* d , 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 d_T , than those without.
- **PACMAN Navigator performs best.** Our proposed navigator (PACMAN+Q) achieves the smallest distance to target at termination (d_T), and the RL-finetuned navigator (PACMAN-RL+Q) achieves highest answering accuracy.

²Fig. 5 and embodiedqa.org show example navigation and answer predictions by our agent.

- **RL agent overshoots.** We observe that while PACMAN-RL+Q gets closest to the target (least d_{min}) and enters the target room most often (highest $\%r_e$), it does *not* end closest to the target (does not achieve lowest d_T). These and our qualitative analysis suggests that this is because RL-finetuned agents learn to explore, with a lower stopping rate ($\%stop$), and often overshoot the target. This is consistent with observations in literature [7]. This does not hurt QA accuracy because the answerer can attend to the last 5 frames along the trajectory, and can potentially be corrected by including a small penalty 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 takes an agent right up to the target object, it may not provide the best vantage or context to answer the question.

6. Conclusion

We present EmbodiedQA – a new 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 intelligently navigate to explore the environment, gather information through first-person (egocentric) vision, and then answer the question. We develop EQA, a dataset of questions in House3D environments, and a novel hierarchical neural model, trained via imitation and reinforcement learning. We benchmark performance of our model against baselines and oracles. We also collect human demonstrations by connecting our environment to Mechanical Turk and letting workers remotely control embodied agents in real time. All our code, data, and infrastructure will be publicly available.

Acknowledgements. We thank the developers of PyTorch [41] for building an excellent framework, and Yuxin Wu for help with House3D environments. 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, DARPA XAI grant to DB and DP, 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.

References

- [1] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3D environment," *arXiv preprint arXiv:1801.02209*, 2018. 1, 2, 3, 4
- [2] L. Smith and M. Gasser, "The development of embodied cognition: six lessons from babies," *Artificial life*, vol. 11, no. 1-2, 2005. 1
- [3] 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
- [4] 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
- [5] 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
- [6] 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
- [7] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *EMNLP*, 2017. 2, 8
- [8] 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
- [9] 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
- [10] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual Question Answering," in *ICCV*, 2015. 3
- [11] 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
- [12] 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," in *CVPR*, 2018. 3
- [13] M. Tapaswi, Y. Zhu, R. Stiefelhausen, A. Torralba, R. Urtasun, and S. Fidler, "MovieQA: Understanding Stories in Movies through Question-Answering," *CVPR*, 2016. 3
- [14] 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
- [15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005. 3
- [16] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumathi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *AAAI*, 2018. 3
- [17] 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
- [18] 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
- [19] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *CVPR*, 2017. 3
- [20] 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
- [21] S. Brahmabhatt and J. Hays, "DeepNav: Learning to Navigate Large Cities," in *CVPR*, 2017. 3
- [22] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *ICML*, 2017. 3, 6
- [23] 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
- [24] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *ICLR*, 2018. 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, 5
- [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, 5
- [28] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *ICML*, 2017. 3, 5, 6
- [29] D. K. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *EMNLP*, 2017. 3, 5
- [30] S. I. Wang, P. Liang, and C. D. Manning, "Learning language games through interaction," in *ACL*, 2016. 3, 5
- [31] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR*, 2018. 3
- [32] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: Visual question answering in interactive environments," in *CVPR*, 2018. 3
- [33] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "A12-THOR: An Interactive 3D Environment for Visual AI," *arXiv preprint arXiv:1712.05474*, 2017. 3
- [34] 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
- [35] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, 2017. 3
- [36] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learn-

- ing in minecraft," in *AAAI*, 2017. 3, 6
- [37] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016. 3, 5
- [38] "Planner5d." <https://planner5d.com/>. 4
- [39] 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
- [40] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992. 7
- [41] "PyTorch." <http://pytorch.org/>. 8