IQA: Visual Question Answering in Interactive Environments

Daniel Gordon[1] Aniruddha Kembhavi[2] Mohammad Rastegari[2,4]
Joseph Redmon[1] Dieter Fox[1,3] Ali Farhadi[1,2]
[1]Paul G. Allen School of Computer Science, University of Washington
[2]Allen Institute for Artificial Intelligence
[3]Nvidia [4]Xnor.ai

Abstract

*We introduce Interactive Question Answering (*IQA*), the task of answering questions that require an autonomous agent to interact with a dynamic visual environment.* IQA *presents the agent with a scene and a question, like: "Are there any apples in the fridge?" The agent must navigate around the scene, acquire visual understanding of scene elements, interact with objects (e.g. open refrigerators) and plan for a series of actions conditioned on the question. Popular reinforcement learning approaches with a single controller perform poorly on* IQA *owing to the large and diverse state space. We propose the Hierarchical Interactive Memory Network (*HIMN*), consisting of a factorized set of controllers, allowing the system to operate at multiple levels of temporal abstraction. To evaluate* HIMN*, we introduce* IQUAD V1*, a new dataset built upon AI2-THOR [35], a simulated photo-realistic environment of configurable indoor scenes with interactive objects.* IQUAD V1 *has 75,000 questions, each paired with a unique scene configuration. Our experiments show that our proposed model outperforms popular single controller based methods on* IQUAD V1*. For sample questions and results, please view our video: https://youtu.be/pXd3C-1jr98.*

1. Introduction

A longstanding goal of the artificial intelligence community has been to create agents that can perform manual tasks in the real world and can communicate with humans via natural language. For instance, a household robot might be posed the following questions: *Do we need to buy more milk?* which would require it to navigate to the kitchen, open the fridge and check to see if there is sufficient milk in the milk jug, or *How many boxes of cookies do we have?* which would require the agent to navigate to the cabinets, open several of them and count the number of cookie boxes. Towards this goal, Visual Question Answering (VQA), the problem of answering questions about visual content, has received significant attention from the computer vision and natural language processing communities. While there has been a lot of progress on VQA, research by and large focuses on answering questions passively about visual content, i.e. without the ability to interact with the environment generating the content. An agent that is only able to answer questions passively is limited in its capacity to aid humans in their tasks.

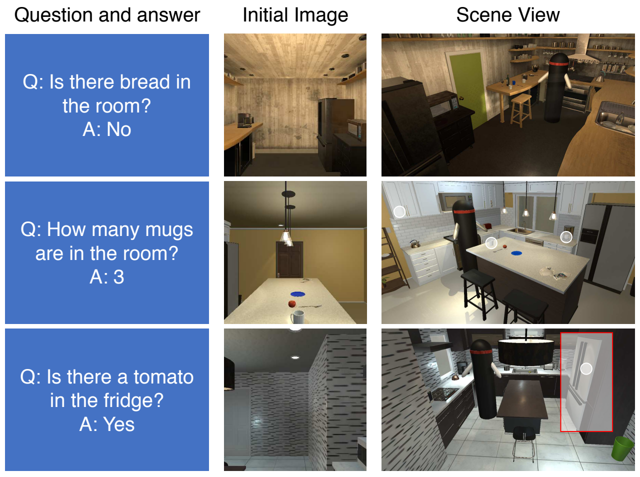
Question and answer
Initial Image
Scene View
Q: Is there bread in the room?
A: No
Q: How many mugs are in the room?
A: 3
Q: Is there a tomato in the fridge?
A: Yes

Figure 1. Samples from IQUAD V1: Each row shows a question paired with the agent's initial view and a scene view of the environment (which is not provided to the agent). In the scene view, the agent is shown in black, and the locations of the objects of interest for each question are outlined. Note that none of the questions can be answered accurately given only the initial image.

We introduce **Interactive Question Answering** (IQA), the task of answering questions that require the agent to interact with a dynamic environment. IQA poses several key challenges in addition to the ones posed by VQA. **First**, the agent must be able to navigate through the environment. **Second**, it must acquire an understanding of its environment including objects, actions, and affordances. **Third**, the agent must be able to interact with objects in the environment (such as opening the microwave, picking up books, etc.). **Fourth**, the agent must be able to plan and execute a series of actions in the environment conditioned on the questions asked of it.

To address these challenges, we propose HIMN (Hierarchical Interactive Memory Network). Figure 2 provides an overview of HIMN. Akin to past works on hierarchical reinforcement learning, HIMN is factorized into a hierarchy

IQA: Visual Question Answering in Interactive Environments

Daniel Gordon[1] Aniruddha Kembhavi[2] Mohammad Rastegari[2,4]
Joseph Redmon[1] Dieter Fox[1,3] Ali Farhadi[1,2]
[1]Paul G. Allen School of Computer Science, University of Washington
[2]Allen Institute for Artificial Intelligence
[3]Nvidia [4]Xnor.ai

Abstract

*We introduce Interactive Question Answering (*IQA*), the task of answering questions that require an autonomous agent to interact with a dynamic visual environment.* IQA *presents the agent with a scene and a question, like: "Are there any apples in the fridge?" The agent must navigate around the scene, acquire visual understanding of scene elements, interact with objects (e.g. open refrigerators) and plan for a series of actions conditioned on the question. Popular reinforcement learning approaches with a single controller perform poorly on* IQA *owing to the large and diverse state space. We propose the Hierarchical Interactive Memory Network (*HIMN*), consisting of a factorized set of controllers, allowing the system to operate at multiple levels of temporal abstraction. To evaluate* HIMN*, we introduce* IQUAD V1*, a new dataset built upon AI2-THOR [35], a simulated photo-realistic environment of configurable indoor scenes with interactive objects.* IQUAD V1 *has 75,000 questions, each paired with a unique scene configuration. Our experiments show that our proposed model outperforms popular single controller based methods on* IQUAD V1*. For sample questions and results, please view our video:* `https://youtu.be/pXd3C-1jr98`*.*

1. Introduction

A longstanding goal of the artificial intelligence community has been to create agents that can perform manual tasks in the real world and can communicate with humans via natural language. For instance, a household robot might be posed the following questions: *Do we need to buy more milk?* which would require it to navigate to the kitchen, open the fridge and check to see if there is sufficient milk in the milk jug, or *How many boxes of cookies do we have?* which would require the agent to navigate to the cabinets, open several of them and count the number of cookie boxes. Towards this goal, Visual Question Answering (VQA), the problem of answering questions about visual content, has received significant attention from the computer vision and natural language processing communities. While there has been a lot of progress on VQA, research by and large focuses on answering questions passively about visual content, i.e. without the ability to interact with the environment generating the content. An agent that is only able to answer questions passively is limited in its capacity to aid humans in their tasks.

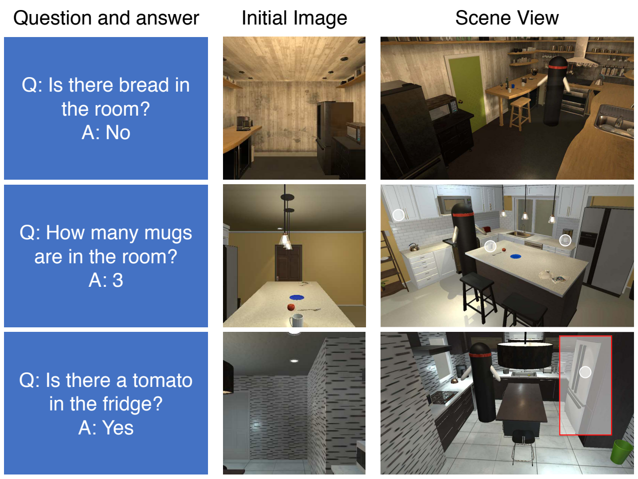
Question and answer
Initial Image
Scene View
Q: Is there bread in the room? A: No
Q: How many mugs are in the room? A: 3
Q: Is there a tomato in the fridge? A: Yes

Figure 1. Samples from IQUAD V1: Each row shows a question paired with the agent's initial view and a scene view of the environment (which is not provided to the agent). In the scene view, the agent is shown in black, and the locations of the objects of interest for each question are outlined. Note that none of the questions can be answered accurately given only the initial image.

We introduce **Interactive Question Answering** (IQA), the task of answering questions that require the agent to interact with a dynamic environment. IQA poses several key challenges in addition to the ones posed by VQA. **First**, the agent must be able to navigate through the environment. **Second**, it must acquire an understanding of its environment including objects, actions, and affordances. **Third**, the agent must be able to interact with objects in the environment (such as opening the microwave, picking up books, etc.). **Fourth**, the agent must be able to plan and execute a series of actions in the environment conditioned on the questions asked of it.

To address these challenges, we propose HIMN (Hierarchical Interactive Memory Network). Figure 2 provides an overview of HIMN. Akin to past works on hierarchical reinforcement learning, HIMN is factorized into a hierarchy

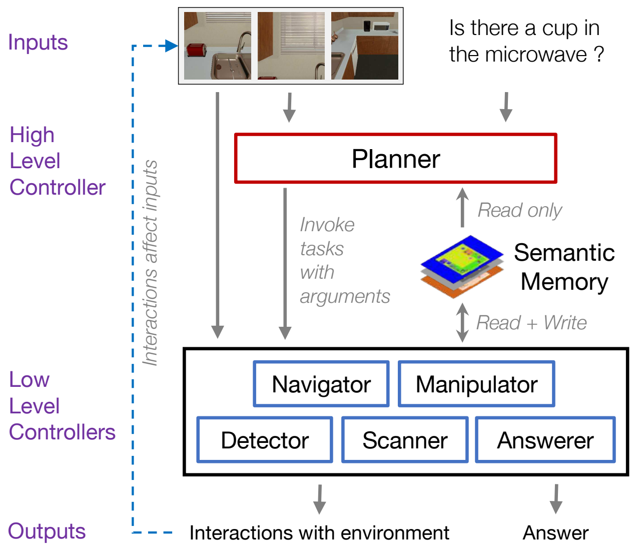

Figure 2. An overview of the Hierarchical Interactive Memory Network (HIMN)

of controllers, allowing the system to operate, learn, and reason across multiple time scales while simultaneously reducing the complexity of each individual subtask. A high level controller, referred to as the *Planner* chooses the task to be performed (for example, navigation / manipulation / answering / etc.) and generates a command for the chosen task. Tasks specified by the *Planner* are executed by a set of low level controllers (*Navigator*, *Manipulator*, *Detector*, *Scanner* and *Answerer*) which return control to the *Planner* when a task termination state is reached. Since these subtasks are fairly independent, we can pretrain each controller independently, while assuming oracle versions of the remaining controllers. Our experiments show that this factorization enables higher accuracy and generalization to unseen environments.

Several question types require the agent to remember where it has been and what it has seen. For example, *How many pillows are in this house?* requires an agent to navigate around the rooms, open closets and keep track of the number of pillows it encounters. For sufficiently complex spaces, the agent needs to hold this information in memory for a long time. This motivates the need for an explicit external memory representation that is filled by the agent as it interacts with its environment. This memory must be both spatial and semantic so it can represent *what* is *where*. We propose a new recurrent layer formulation: Egocentric Spatial GRU (esGRU) to represent this memory (Sec 4.1).

Training and evaluating interactive agents in the real world is currently prohibitive from the standpoint of operating costs, scale and research reproducibility. A far more viable alternative is to train and evaluate such agents in realistic simulated environments. Towards this end, we present the Interactive Question Answering Dataset (IQUAD V1) built upon AI2-THOR [35], a photo-realistic customizable simulation environment for indoor scenes integrated with the Unity [1] physics engine. IQUAD V1 consists of over 75,000 multiple choice questions, each question accompanied by a unique scene configuration.

We evaluate HIMN on IQUAD V1 using a question answering accuracy metric and show that it outperforms a baseline based on a common architecture for reinforcement learning used in past work. We evaluate in both familiar and unfamiliar environments to show that our semantic model generalizes well across scenes.

In summary, our contributions include: (a) proposing Interactive Question Answering, the task of answering questions that require the agent to interact with a dynamic environment, (b) presenting the Hierarchical Interactive Memory Network, a question answering model factorized into a high level *Planner*, a set of low level controllers and a rich semantic spatial memory, (c) the Egocentric Spatial GRU, a new recurrent layer to represent this memory and (d) a new dataset IQUAD V1 towards the task of IQA.

2. Related Work

Visual Question Answering (VQA): VQA has seen significant progress over the past few years, owing to the design of deep architectures suited for this task and the creation of large VQA datasets to train these models [71]. These include datasets of natural images [2, 36, 41, 68], synthetic images [2, 4, 24, 27, 28], natural videos [23, 65], synthetic videos [32, 49] and multimodal contexts [29]. Some of these use questions written by humans [2, 28, 29, 36] and others use questions that are generated automatically [4, 24, 27]. IQUAD V1 is set in a photo-realistic simulation environment and uses automatically generated questions. In contrast to the aforementioned datasets that only require the agent to observe the content passively, IQUAD V1 requires the agent to interact with a dynamic environment.

The first deep architectures designed for VQA involved using an RNN to encode the question, using a CNN to encode the image and combining them using fully connected layers to yield the answer [2, 42]. More recently, modular networks [4, 20, 25] that construct an explicit representation of the reasoning process by exploiting the compositional nature of language have been proposed. Similar architectures have also been applied to the video domain with extensions such as spatiotemporal attention [23, 49]. Our proposed approach to question answering allows the agent to interact with its environment and is thus fundamentally different to past QA approaches. However, we note that approaches such as visual attention and modularity can easily be combined with our model to provide further improvements.

Reinforcement Learning (RL): RL algorithms have been employed in a wide range of problems including locomotion [33], obstacle detection [44] and autonomous flight [34, 59]. Of particular relevance to our approach is the area of hierarchical reinforcement learning (HRL), which consists of a high level controller and one or more low level controllers. The high-level controller selects a subtask to be executed and invokes one of the low level controllers. The advantage of HRL is that it allows the model to operate at multiple levels of temporal abstraction. Early works proposing HRL algorithms include [11, 54, 64]. More recent approaches include [37] who propose hierarchical-DQN with an intrinsically motivated RL algorithm, [66] who use HRL to create a lifelong learning system that has the ability to reuse and transfer knowledge from one task to another, and [51] who use HRL to enable zero shot task generalization by learning subtask embeddings that capture correspondences between similar subtasks. Our use of HRL primarily lets us learn at multiple time scales and its integration with the semantic memory lets us divide the complex task of IQA into more concrete tasks of navigation, detection, planning etc. that are easier to train.

RL techniques have also recently been applied to QA tasks, most notably by [25] to train a program generator that constructs an explicit representation of the reasoning process to be performed and an execution engine that executes the program to predict the answer.

Visual Navigation: The majority of visual navigation techniques fall into three categories: offline map-based, online map-based, and map-less approaches. Offline map-based techniques [6, 7, 31, 52] require the complete map of the environment to make any decisions about their actions, which limits their use in unseen environments. Online map-based methods [10, 13, 50, 61, 67, 69] often construct the map while exploring the environment. The majority of these approaches use the computed map for navigation only, whereas our model constructs a rich semantic map which is used for navigation as well as planning and question answering. Map-less approaches [17, 22, 40, 60, 74] which use techniques such as obstacle avoidance and feature matching, depend upon implicit representations of the world to perform navigation, and lack long-term memory capabilities. Recently Gupta *et al*. [16] proposed a joint architecture for a *mapper* that produces a spatial memory and a *planner* that can plan paths. The similarities between our works lie in the usage of a hierarchical system and a spatial memory. In contrast to their work, navigation is not the end goal of our system, but a subtask towards question answering, and our action space is more diverse as it includes interaction and question answering.

Visual Planning: To answer questions such as *Do I need to buy milk?* an agent needs to plan a sequence of actions to explore and interact with the environment. A large body of research on planning algorithms [12, 14, 26, 62, 63] use high-level formal languages. These techniques are designed to handle low-dimensional state spaces but do not scale well to high-dimensional state spaces such as natural images.

Other relevant work includes visual navigation [74] and visual semantic planning [73] which both use the AI2-THOR environment [35]. The former tackles navigation, and the latter focuses on high level planning and assumes an ideal low level task executor; in contrast, our model trains low level and high level controllers jointly. Also, both these approaches do not generalize well to unseen scenes, whereas our experiments show that we do not overfit to previously encountered environments. Finally, these methods lack any sort of explicit map, whereas we construct a semantic map which helps us navigate and answer questions.

Recently Chaplot *et al*. [8] and Hill *et al*. [19] have proposed models to complete navigation tasks specified via language (e.g. *Go to the red keycard*) and trained their systems in simulated 3D environments. These models show the ability to generalize to unseen instructions of seen concepts. In contrast, we tackle several question types that require a variety of navigation behaviours and interaction, and the environment we use is significantly more photo-realistic. In our experiments, we compare our proposed HIMN model to a baseline system (A3C in Section 5) that very closely resembles the model architectures proposed in [8] and [19].

Visual Learning by Simulation: There has been an increased use of simulated environments and game platforms to train computer vision systems to perform tasks such as learning the dynamics of the world [47, 48, 70], semantic segmentation [18], pedestrian detection [43], pose estimation [53] and urban driving [3, 9, 57, 58]. Several of these are also interactive making them suitable to learn control, including [5, 30, 35, 39, 72]. We choose to use AI2-THOR [35] in our work since it provides a photo-realistic and interactive environment of real world scenes, making it very suitable to train IQA systems that might be transferable to the real world.

3. Learning Framework

3.1. Actionable Environment

Training and evaluating interactive agents in the real world is currently prohibitive from the standpoint of operating costs, scale, time, and research reproducibility. A far more viable alternative is to use simulated environments. However, the framework should be visually realistic, allow interactions with objects, and have a detailed model of the physics of the scene so that agent movements and object interactions are properly represented. Hence, we adopt the AI2-THOR environment [35] for our purposes. AI2-THOR is a photo-realistic simulation environment of 120 rooms in indoor settings, tightly integrated with a physics engine. Each scene consists of a variety of objects, from furniture such as couches, appliances such as microwaves and smaller objects such as crockery, cutlery, books, fruit, etc. Many of these objects are actionable such as fridges which can be opened, cups which can be picked up and put down, and stoves which can be turned on and off.

3.2. Interactive Question Answering Dataset

IQUAD V1 is a question answering dataset built upon AI2-THOR [35]. It consists of over 75,000 multiple choice questions for three different question types (table 1 shows more detailed statistics). Each question is accompanied by a scene identifier and a unique arrangement of movable objects in the scene. Figure 1 shows three such examples. The wide variety of configurations in IQUAD V1 prevent models from memorizing simple rules like “apples are always in the fridge” and render this dataset challenging. IQUAD V1 consists of several question types including: Existence ques-

Interactive Question Answering Dataset Statistics		
	Train	Test
Existence	25,600	640
Counting	25,600	640
Spatial Relationships	25,600	640
Rooms	25	5
Total scene configurations (s.c.)	76,800	1,920
Avg # objects per (s.c.)	46	41
Avg # interactable objects (s.c.)	21	16
Vocabulary Size	70	70

Table 1. This table shows the statistics of our proposed dataset in a variety of question types, objects and scene configurations.

tions (*Is there an apple in the kitchen?*), Counting questions (*How many forks are present in the scene?*), and Spatial Relationships questions (*Is there lettuce in the fridge? / Is there a cup on the counter-top?*). Questions, ground truth answers, and answer choices are generated automatically. Since natural language understanding is not a focus of this dataset, questions are generated using a set of templates written down a priori. Extending IQUAD V1 to include more diverse questions generated by humans is future work. IQUAD V1 is a balanced dataset that prevents models from obtaining high accuracies by simply exploiting trivial language and scene configuration biases. Similar to past balanced VQA datasets [15], each question is associated with multiple scene configurations that result in different answers to the question. We split the 30 kitchen rooms into 25 train and 5 test, and have 1024 unique (question, scene configuration) pairs for each (room, question type) pair in train, and 128 in test. An episode is finished when the *Answerer* is invoked. We evaluate different methods using Top-1 accuracy.

3.3. Agent and Objects

The agent in our environments has a single RGB camera mounted at a fixed height. An agent can perform one of five navigation actions (move ahead 25 cm, rotate 90 degrees left or right, look up or down 30 degrees). We assume a grid-world floor plan that ensures that the agent always moves along the edges of a grid and comes to a stop on a node in this grid. The agent can perform two interaction actions (open and close) to manipulate objects. A wide variety of objects (fridges, cabinets, drawers, microwaves, etc.) can be interacted with. If there are multiple items in the current viewpoint which can be opened or closed, the environment chooses the one nearest to the center of the current image. The success of each action depends on the current state of the environment as well as the agent's current location. For instance, the agent cannot open a cabinet that is more than 1 meter away or is not in view, or is already open, and it cannot walk through a table or a wall.

4. Model

We propose HIMN (Hierarchical Interactive Memory Network), consisting of a hierarchy of controllers that operate at multiple levels of temporal abstraction and a rich semantic memory that aids in navigation, interaction, and question answering. Figure 2 provides an overview of HIMN. We now describe each of HIMN's components in greater detail.

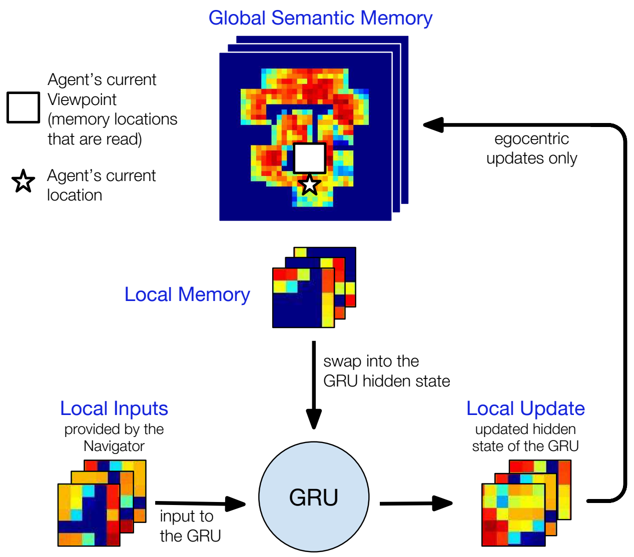
Global Semantic Memory
Agent's current Viewpoint (memory locations that are read)
Agent's current location
egocentric updates only
Local Memory
swap into the GRU hidden state
Local Inputs
provided by the Navigator
Local Update
updated hidden state of the GRU
GRU
input to the GRU

Figure 3. An overview of the Egocentric Spatial GRU (esGRU): The esGRU only allows writing to a local window within the memory, dependent on the agent's current location and viewpoint.

4.1. Spatial Memory

Several question types require the agent to keep track of objects that it has seen in the past along with their locations. For complex scenes with several locations and interactable objects, the agent needs to hold this information in memory for a long duration. This motivates the need for an explicit external memory representation that is filled by the agent on the fly and can be accessed at any time. To address this, HIMN uses a rich semantic spatial memory that encodes a semantic representation of each location in the scene. Each location in this memory consists of a feature vector encoding object detection probabilities, free space probability (a 2D occupancy grid), coverage (has the agent inspected this location before), and navigation intent (has the agent attempted to visit this location before). We propose a new recurrent layer formulation: Egocentric Spatial GRU (esGRU) to represent this memory, illustrated in Figure 3. The esGRU maintains an external global spatial memory represented as a 3D tensor. At each time step, the esGRU swaps in local egocentric copies of this memory into the hidden state of the GRU, performs computations using current inputs, and then swaps out the resulting hidden state into the global memory at the predetermined location. This speeds up computations and prevents corrupting the memory at locations far away from the agent's current viewpoint. When navigating and answering questions, the agent can access the full memory, enabling long-term recall from observations seen hundreds of states prior. Furthermore, only

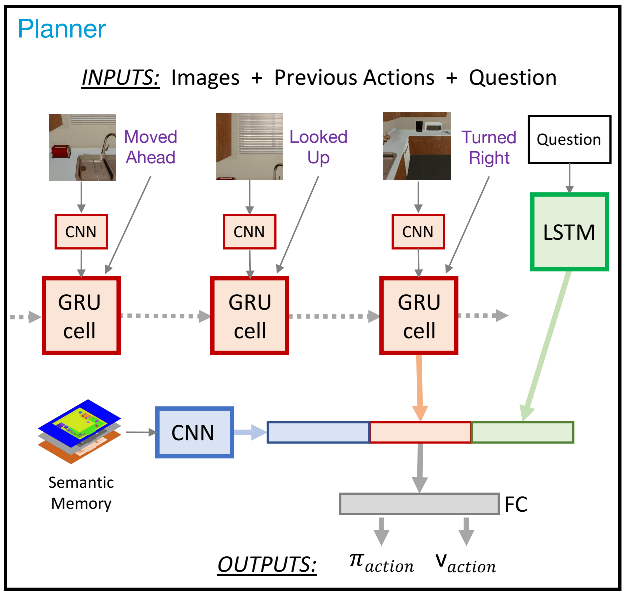

Figure 4. Schematic representation of the *Planner*

low level controllers have read-write access to this memory. Since the *Planner* only makes high level decisions, without interacting with the world at a lower level, it only has read access to the memory.

4.2. Planner

The high level *Planner* invokes low level controllers in order to explore the environment, gather knowledge needed to answer the given question, and answer the question. We frame this as a reinforcement learning problem where the agent must issue the fewest possible commands that result in a correct answer. The agent must learn to explore relevant areas of the scene based on learned knowledge (e.g. apples are often in the fridge, cabinets are openable, etc.), the current memory state (e.g. the fridge is to the left), current observations (e.g. the fridge is closed) and the question. At every timestep, the *Planner* chooses to either invoke the *Navigator* providing a relative location in a 5x5 grid in front of the agent, invoke the *Scanner* with a direction such as up or left, invoke the *Manipulator* with open/close commands on a nearby object, or invoke the *Answerer* for a total of 32 discrete actions. It does this by producing a policy π consisting of probabilities π_i for each action, and a value v for the current state. π and v are learned using the A3C algorithm [46]. Figure 4 shows a schematic of the *Planner*. It consists of a GRU which accepts at each time step the current viewpoint (encoded by a CNN) and the previous action. The *Planner* has read only access to the semantic memory centered around the agent's current location. The output of this GRU is combined with the question embedding and an embedding of the nearby semantic spatial memory to predict π and v. The agent receives a fixed reward/penalty based on answering correctly/incorrectly. It is also provided a constant time penalty to encourage efficient explorations of the environment and quick answering, as well as a penalty for attempting to perform invalid actions. The agent is also given intermediate rewards for increasing the "coverage" of the environment, effectively training the network to maximize the amount of the room it has explored as quickly as possible. Finally, at each time step, the *Planner* also predicts which high level actions are viable given the current world state. In many locations in the scenes, certain navigation destinations are unreachable or there are no objects to interact with. Predicting possible/impossible actions at each time step, allows gradients to propagate through all actions rather than just the chosen action. This leads to higher accuracies and faster convergence (see section 5.2 for more details).

4.3. Low level controllers

Navigator The *Navigator* is invoked by the *Planner* which also provides it with the relative coordinates of the target location. Given a destination specified by the *Planner* and the current estimate of the room's occupancy grid, the *Navigator* runs A* search to find the shortest path to the goal. As the *Navigator* moves through the environment, it uses the esGRU to produce a local (5x5) occupancy grid given the current visual observation. This updates the global occupancy estimate, and prompts a new shortest-path computation. This is a fully supervised problem and can be trained with the standard sigmoid-cross-entropy. The *Navigator* also invokes the *Scanner* to obtain a wide angle view of the environment. Given that the requested destination may be outside the bounds of the room or otherwise impossible (e.g. at a wall or other obstacle), the *Navigator*'s network also predicts a termination signal, and returns control to the *Planner* when the prediction passes a certain threshold.
Scanner The *Scanner* is a simple controller which captures images by rotating the camera up, down, left, or right while maintaining the agent's current location. The *Scanner* calls the *Detector* on every new image.
Detector Object detection is a critical component of HIMN given that all questions in IQUAD V1 involve one or more objects in the room. We use YOLOv3 [56] fine-tuned on the AI2-THOR training scenes as an object detector. We estimate the depth of an object using the FRCN depth estimation network [38] and project the probabilities of the detected objects onto the ground plane. Both of these networks operate at real-time speeds, which is necessary since they are invoked on every new image. The detection probabilities are incorporated into the spatial memory using a moving average update rule. We also perform experiments where we substitute the trained detector and depth estimator with oracle detections. Detections provided by the environment still requires the network to learn affordances. For instance, the network must learn that *microwaves can be opened*, *apples can be in fridges*, etc.
Manipulator The *Manipulator* is invoked by the *Planner* to manipulate the current state of an object. For example, opening and closing the microwave. This leads to a change in the visual appearance of the scene. If the object is too far away or out of view, the action will fail.
Answerer The *Answerer* is invoked by the *Planner* to an-

	Existence		Counting		Spatial Relationships	
Model	Accuracy	Length	Accuracy	Length	Accuracy	Length
Most Likely Answer Per Q-type (MLA)	50	-	25	-	50	-
A3C with ground truth (GT) detections	48.59	332.41	24.53	998.32	49.84	578.71
HIMN with YOLO [56] detections	**68.47**	318.33	**30.43**	926.11	**58.67**	516.23
Human (small sample)	90	58.40	80	81.90	90	43.00

Table 2. This tables compares the test accuracy and episode lengths of question answering across different models and question types.

swer the question. It uses the current image, the full spatial memory, and the question embedding vector to predict answer probabilities a_i for each possible answer to the question. The question vector is tiled to create a tensor with the same width and height as the spatial memory. These are depthwise concatenated with the spatial memory and passed through 4 convolution and max pool layers followed by a sum over the spatial layers. This output vector is fed through two fully connected layers and a softmax over possible answer choices. After the *Answerer* is invoked, the episode ends, whether the answer was correct or not.

4.4. Training

The full system is trained jointly. However, since the individual tasks of the controllers are mostly independent, we are able to pretrain them separately. Our initial analysis showed that this leads to faster convergence and better accuracy than training end-to-end from scratch. We outline our training procedures below.

Planner: To pretrain the *Planner*, we assume a perfect *Navigator* and *Detector* by using the ground truth shortest path for the *Navigator* and the ground truth object information for the *Detector*.

Navigator: We pretrain the *Navigator* by providing pairs of random starting points and goal locations.

Answerer: The *Answerer* is pretrained by using ground-truth partial semantic maps which contain enough information to answer the current question correctly.

Detector: The *Detector* is pretrained by fine-tuning YOLOv3 [56] on the AI2-THOR training scenes. It is trained to identify small object instances which may repeat in multiple scenes (apples, forks, etc.) as well as large object instances which are unique to each scene (e.g. each fridge model will only exist in one scene).

Scanner and Manipulator: There are no trainable parameters for these controllers in our current setup. Their behavior is predefined by the AI2-THOR environment.

Joint Training After all trainable controllers are pretrained, we update the model end-to-end.

5. Experiments

We evaluate HIMN on the IQUAD V1 dataset, using Top-1 question answering accuracy. An initial baseline of Most likely answer per Question-Type (MLA) shows that the dataset is exactly balanced. Additionally, because we construct the data such that each generated question has a scene configuration for each answer possibility, there is no possible language bias in the dataset. The learned baseline that we compare to (A3C), is based on a common architecture for reinforcement learning used in past works including for visual semantic planning [73] and task oriented language grounding [8, 19]. We extend this for the purpose of question answering. Since HIMN has access to object detections provided by either the environment or YOLO [56], we also provide detections to the baseline. For the baseline model, at each time-step, the raw RGB image observed by the agent is concatenated depth wise with object detections (one channel per object class). This tensor is passed through convolutional layers and fed into a GRU. The question is passed through an LSTM. The output of the LSTM and GRU are concatenated, and passed through two fully connected layers to produce probabilities π_i for each action and a value v. The output of the first fully connected layer is also passed to an answering module that consists of two more fully connected layers with a softmax on the space of all possible answers. The model is trained using the A3C algorithm for action probabilities and a supervised loss on the answers. We also provide a human baseline of random questions on each question type.

Table 2 shows the test accuracies and the average episode lengths for the proposed HIMN model and baselines for each question type. HIMN significantly outperforms the baselines on all question types, both with YOLO object detections as well as ground truth object detections. Surprisingly, the A3C baseline performs slightly worse than random chance even with ground truth detections. We conjecture that this is because there is no explicit signal for when to answer, and no persistence of object detections. The A3C model is not able to associate object detections with the question, and thus has no reason to remember detections for long periods of time. Because HIMN does not overwrite its entire spatial memory at each timestep, object detections persist for much longer, and the *Answerer* can better learn the associations between the questions and the objects. HIMN further benefits from a spatial memory in counting and spatial relationship questions because these require much more spatial reasoning than existence questions. Additionally, because A3C does not learn to answer questions, it also does not learn to efficiently explore the environments, as most of the reward comes from answering questions correctly. HIMN, on the other hand, traverses much more of the environment, only answering when it is confident that it has sufficiently explored the room. This indicates that HIMN (which uses an explicit semantic spatial memory with egocentric up-

	Existence		Counting		Spatial Relationships	
Model	Accuracy	Length	Accuracy	Length	Accuracy	Length
HIMN with YOLO [56] detections	68.47	318.33	30.43	926.11	58.67	516.23
HIMN with GT detection	86.56	679.70	35.31	604.79	70.94	311.03
HIMN with GT detection and oracle navigator (HIMN-GT)	**88.60**	618.63	**48.44**	871.12	**72.50**	475.55
HIMN-GT Question not given to planner	50.00	150.60	24.50	293.33	50.25	118.09
HIMN-GT No loss on invalid actions	49.84	659.28	24.84	911.46	50.00	613.50

Table 3. Ablation experiments on the HIMN model.

dates) is more effective than A3C (which uses a standard fully-connected GRU) at (a) Estimating when the environment has been sufficiently explored, given the question (b) Keeping track of past observations for much longer durations, which is important in determining answers for questions that require a thorough search of the environment, and (c) Keeping track of multiple object instances in the scene, which may be observed several time steps apart (which is crucial for answering counting questions).

5.1. Ablation Analysis

We perform four ablative experiments on our network structure and inputs, shown in table 3. First, we use the ground truth object detections and depth instead of YOLO [56] and FRCN depth [38]. This adds a dramatic improvement to our model owing primarily to the fact that without detection mistakes, the *Answerer* can be more accurate and confident. Secondly, we substitute our learned navigation controller with an oracle *Navigator* that takes the shortest path in the environment. When the optimal *Navigator* is provided, HIMN further improves. This is because the *Planner* can more accurately direct the agent through the environment, allowing it to be more efficient and more thorough at exploring the environment. It also takes fewer invalid actions (as seen in table 4), indicating that it is less likely to get stuck in parts of the room. In our third ablative experiment, we remove the question vector from the input of the *Planner*, only providing it to the *Answerer*, which results in random performance. This shows that the *Planner* utilizes the question to direct the agent towards different parts of the room to gather information required to answer the question. For instance any question about an object in the fridge requires the planner to know the fridge needs to be opened. If the planner is not told the question, it has no reason to open the fridge, and instead will likely choose to continue exploring the room as exploration often gives more reward than opening an object. Also, some questions can be answered soon after an object is observed (*e.g.* Existence), whereas others require longer explorations (*e.g.* Counting). Having access to the questions can clearly help the *Planner* in these scenarios. Tables 3 shows that HIMN does in fact explore the environment for longer durations for Counting questions than for Existence and Spatial Relationship questions. In our final ablation experiment, we remove the loss on invalid actions. If we do not apply any loss on these actions and only propagate gradients through the chosen action, the agent suffers from the difficulty of exploring a large action space and again performs at random chance.

Percentage of invalid actions			
Model	Existence	Counting	Spatial Relationships
A3C with GT detections	32.75	34.55	32.63
HIMN No loss on invalid actions	56.27	53.43	51.93
HIMN with YOLO [56] detections	6.07	5.95	6.68
HIMN with GT detections	6.49	5.71	5.66
HIMN-GT	**1.79**	**2.02**	**1.27**
Human	5.99	6.47	3.49

Table 4. This tables compares the percentage of invalid actions across different models on test. Lower is better.

5.2. Invalid Actions

Table 4 shows the percentage of invalid actions taken by the different methods. Failed actions are due to navigation failures (failing to see an obstacle) or interaction failures (trying to interact with something too far away or otherwise impossible). There is a clear benefit to including a loss on the invalid actions both in terms of QA accuracy, as can be seen in table 3, as well as in terms of percentage of invalid actions performed, shown in table 4. All models in table 4 are penalized for every invalid action they attempt, but this only provides feedback on a single action at every timestep. With the addition of a supervised loss on all possible actions, the percentage of invalid actions performed is nearly an order of magnitude lower. By directly training our agent to recognize affordances (valid actions), we are able to mitigate the difficulties posed by a large action space, allowing the *Planner* to learn much more quickly. The validity loss also serves as an auxiliary task which has been shown to aid the convergence of RL algorithms [21]. By replacing the learned *Navigator* with an oracle, we observe that the majority of failed actions are due to navigation failures. We believe that with a smaller step size, we would further reduce the navigation errors at the expense of longer trajectories.

	Existence		Counting		Spatial Relationships	
Model	S	U	S	U	S	U
HIMN with YOLO [56] detections	73.68	68.47	36.26	30.43	60.71	58.67
HIMN with GT detections	94.00	86.56	42.38	35.31	73.38	70.94

Table 5. This tables compares the accuracy of question answering across different models on Seen (S) and Unseen (U) environments.

5.3. Generalization in Unseen Environments

One benefit of HIMN over other RL architectures is that encoding semantic information into a spatial map should generalize well in both seen and unseen environments. Thus, in table 5, we compare HIMN's performance on seen and unseen environments. Unseen environments tests the agent with questions that occur in 5 never-before-seen

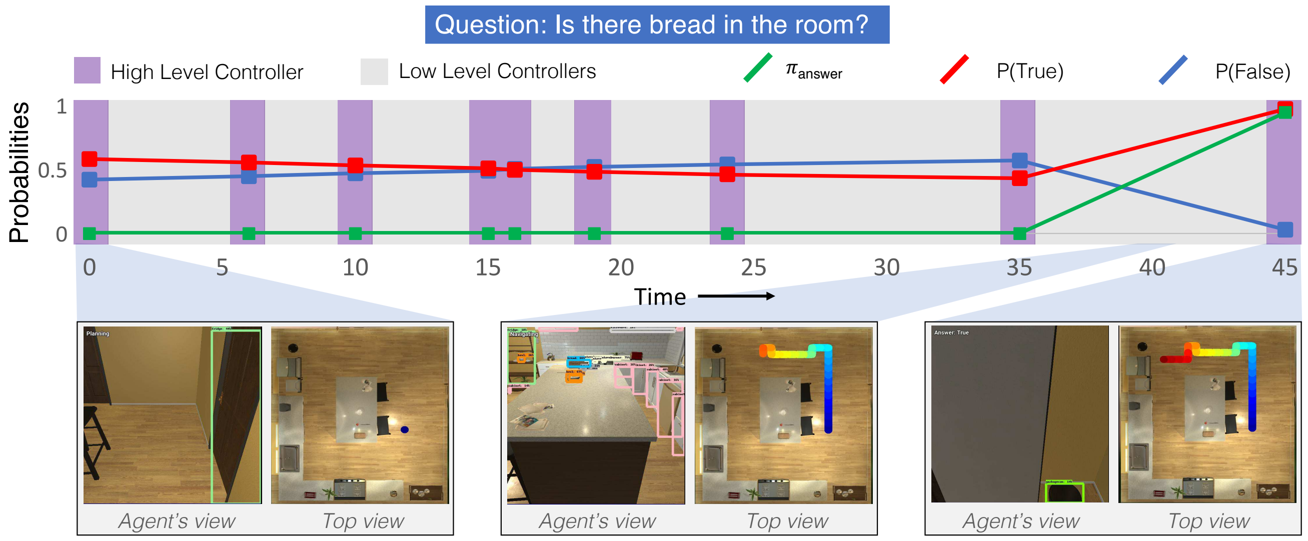

Figure 5. A sample trajectory for answering the existence question: *Is there bread in the room?* The purple sections indicate a *Planner* step, and the gray sections indicate a lower level controller such as the *Navigator* is controlling the agent. For a more detailed explanation, refer to section 5.4.

rooms, whereas the seen environments use the 25 training rooms but never-before-seen object placements and corresponding questions. Despite our relatively small number of training rooms, table 5 shows that our method only loses up to a few percentage points of accuracy when tested on unseen environments. This contrasts with many other end-to-end RL methods which learn deep features that tend to limit their applicability outside a known domain [73, 74]. Note that all experiments in previous sections were only performed on unseen environments.

5.4. Qualitative Results

Figure 5 shows a sample run of HIMN for the question "Is there bread in the room." Initially, $P(True)$ and $P(False)$ both start near 50%. The *Planner* begins searching the room by navigating around the kitchen table. During the initial exploration phase, bread is not detected, and $P(False)$ slowly increases. At timestep 39, the *Navigator* invokes the *Detector*, which sees the bread and incorporates it into the semantic spatial map. However, the *Navigator* does not return control to the *Planner*, as it has not yet reached the desired destination. Upon returning at timestep 45, the *Planner* reads the spatial map, sees the bread, and immediately decides it can answer the question. Thus π_{answer} and $P(True)$ both increase to nearly 100%. For more examples, please see our supplementary video https://youtu.be/pXd3C-1jr98.

5.5. Limitations

Although HIMN performs quite well, it still has several obvious limitations. Due to the 2D nature of the semantic spatial map, HIMN is unable to differentiate between an object being inside a container and being on top of the container. Two obvious extensions of HIMN are storing an explicit height parameter or using multiple 2D slices to construct a 3D map. Secondly, as can be seen in the human experiments in table 2, HIMN is still fairly inefficient at exploring the environment. We plan on investigating more traditional planning algorithms to reduce the time spent exploring previously searched areas. Finally, our templated language model is quite simple, and would not extend to arbitrary questions. We plan on extending IQUAD to include more varied questions, and we will use more expressive language embeddings like [45, 55] in future work.

6. Conclusion

In this work, we pose a new problem of Interactive Question Answering for several question types in interactive environments. We propose the Hierarchical Interactive Memory Network, consisting of a factorized set of controllers, allowing the system to learn from long trajectories. We also introduce the Egocentric Spatial GRU for updating spatial memory maps. The effectiveness of our proposed model is demonstrated on a new benchmark dataset built upon a high-quality simulation environment for this task. This dataset still presents several challenges to our model and baselines and warrants future research.

7. Acknowledgements

This work was funded in part by the National Science Foundation under contract number NSF-NRI-1637479, NSF-IIS-1338054, NSF-1652052, ONR N00014-13-1-0720, the Allen Distinguished Investigator Award, and the Allen Institute for Artificial Intelligence. We would like to thank Xun Huang for initial discussions and dataset prototypes. We would also like to thank NVIDIA for generously providing a DGX used for this research via the UW NVIDIA AI Lab (NVAIL).

References

[1] Unity software. https://unity3d.com. 2

[2] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 2017. 2

[3] J. L. Alireza Shafaei and M. Schmidt. Play and learn: Using video games to train computer vision models. In E. R. H. Richard C. Wilson and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 26.1–26.13. BMVA Press, September 2016. 3

[4] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 2013. 3

[6] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 1989. 3

[7] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 1991. 3

[8] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *CoRR*, abs/1706.07230, 2017. 3, 6

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, 2015. 3

[10] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, 2003. 3

[11] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.*, 2000. 3

[12] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel. Integrating symbolic and geometric planning for mobile manipulation. In *Safety, Security & Rescue Robotics (SSRR), 2009 IEEE International Workshop on*, 2009. 3

[13] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014. 3

[14] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 1971. 3

[15] 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. 4

[16] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017. 3

[17] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In *International Conference on Robotics and Automation*, 1998. 3

[18] A. Handa, V. Ptrucean, V. Badrinarayanan, R. Cipolla, et al. Understanding realworld indoor sceneswith synthetic data. In *CVPR*, 2016. 3

[19] F. Hill, K. M. Hermann, P. Blunsom, and S. Clark. Understanding grounded language learning agents. *CoRR*, abs/1710.09867, 2017. 3, 6

[20] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. *CoRR*, abs/1704.05526, 2017. 2

[21] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017. 7

[22] A. Jaegle, S. Phillips, and K. Daniilidis. Fast, robust, continuous monocular egomotion computation. In *International Conference on Robotics and Automation*, 2016. 3

[23] Y. Jang, Y. Song, Y. Yu, Y. Kim, and G. Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. *CoRR*, abs/1704.04497, 2017. 2

[24] J. Johnson, B. Hariharan, L. van der Maaten, F. fei Li, C. L. Zitnick, and R. B. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. 2

[25] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, F. fei Li, C. L. Zitnick, and R. B. Girshick. Inferring and executing programs for visual reasoning. *CoRR*, abs/1705.03633, 2017. 2, 3

[26] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *International Conference on Robotics and Automation*, 2011. 3

[27] S. E. Kahou, A. Atkinson, V. Michalski, Á. Kádár, A. Trischler, and Y. Bengio. Figureqa: An annotated figure dataset for visual reasoning. *CoRR*, abs/1710.07300, 2017. 2

[28] A. Kembhavi, M. Salvato, E. Kolve, M. J. Seo, H. Hajishirzi, and A. Farhadi. A diagram is worth a dozen images. In *ECCV*, 2016. 2

[29] A. Kembhavi, M. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *CVPR*, 2017. 2

[30] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, 2016. 3

[31] D. Kim and R. Nevatia. Symbolic navigation with a generic map. *Autonomous Robots*, 1999. 3

[32] K.-M. Kim, M.-O. Heo, S.-H. Choi, and B.-T. Zhang. Deepstory: Video story qa by deep embedded memory networks. In *IJCAI*, 2017. 2

[33] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *International Conference on Robotics and Automation*, 2004. 2

[34] T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 2008. 2

[35] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1, 2, 3

[36] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and F. fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 2016. 2

[37] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating

temporal abstraction and intrinsic motivation. In *NIPS*, 2016. 3

[38] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, 2016. 5, 7

[39] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning*, pages 430–438, 2016. 3

[40] C. Linegar, W. Churchill, and P. Newman. Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera. In *International Conference on Robotics and Automation*, 2016. 3

[41] M. Malinowski and M. Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*, 2014. 2

[42] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015. 2

[43] J. Marin, D. Vázquez, D. Gerónimo, and A. M. López. Learning appearance in virtual scenarios for pedestrian detection. In *CVPR*, 2010. 3

[44] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *International Conference on Machine Learning*, 2005. 2

[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 8

[46] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016. 5

[47] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *CVPR*, 2016. 3

[48] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi. what happens if... learning to predict the effect of forces in images. In *ECCV*, 2016. 3

[49] J. Mun, P. H. Seo, I. Jung, and B. Han. Marioqa: Answering questions by watching gameplay videos. *CoRR*, abs/1612.01669, 2016. 2

[50] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 2015. 3

[51] J. Oh, S. Singh, H. Lee, and P. Kohli. Communicating hierarchical neural controllers for learning zero-shot task generalization. 2016. 3

[52] G. Oriolo, M. Vendittelli, and G. Ulivi. On-line map building and navigation for autonomous mobile robots. In *International Conference on Robotics and Automation*, 1995. 3

[53] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic rgb-d. In *ICCV*, 2015. 3

[54] R. E. Parr and S. J. Russell. Reinforcement learning with hierarchies of machines. In *NIPS*, 1997. 3

[55] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 8

[56] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. 2018. 5, 6, 7

[57] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 3

[58] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 3

[59] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016. 2

[60] P. Saeedi, P. D. Lawrence, and D. G. Lowe. Vision-based 3-d trajectory tracking for unknown environments. *IEEE transactions on robotics*, 2006. 3

[61] R. Sim and J. J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. In *International Conference on Intelligent Robots and Systems*. IEEE, 2006. 3

[62] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *International Conference on Robotics and Automation*, 2014. 3

[63] S. Srivastava, L. Riano, S. Russell, and P. Abbeel. Using classical planners for tasks with continuous operators in robotics. In *Intl. Conf. on Automated Planning and Scheduling*, 2013. 3

[64] R. S. Sutton, D. Precup, and S. P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 1999. 3

[65] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. Movieqa: Understanding stories in movies through question-answering. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4631–4640, 2016. 2

[66] 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

[67] M. Tomono. 3-d object map building using dense object models with sift-based recognition features. In *International Conference on Intelligent Robots and Systems*, 2006. 3

[68] P. Wang, Q. Wu, C. Shen, A. van den Hengel, and A. R. Dick. Fvqa: Fact-based visual question answering. *IEEE transactions on pattern analysis and machine intelligence*, 2017. 2

[69] D. Wooden. A guide to vision-based map building. *IEEE Robotics & Automation Magazine*, 2006. 3

[70] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*, 2015. 3

[71] Q. Wu, D. Teney, P. Wang, C. Shen, A. R. Dick, and A. van den Hengel. Visual question answering: A survey of methods and datasets. *CoRR*, abs/1607.05910, 2016. 2

[72] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 2000. 3

[73] 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. *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 3, 6, 8

[74] 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 *International Conference on Robotics and Automation*, 2017. 3, 8

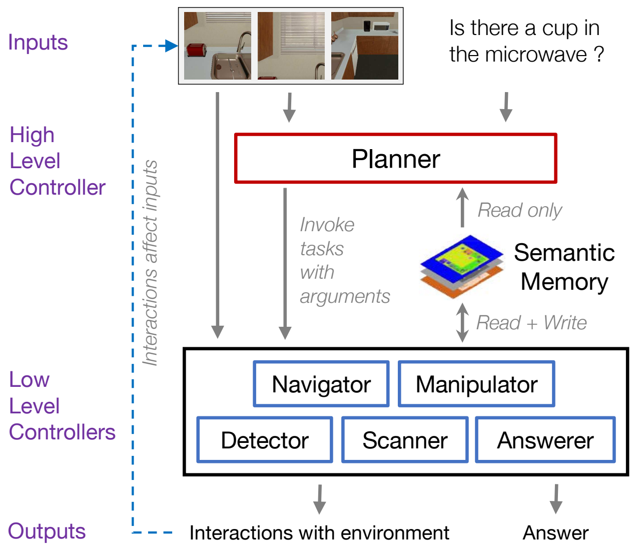

Figure 2. An overview of the Hierarchical Interactive Memory Network (HIMN)

of controllers, allowing the system to operate, learn, and reason across multiple time scales while simultaneously reducing the complexity of each individual subtask. A high level controller, referred to as the *Planner* chooses the task to be performed (for example, navigation / manipulation / answering / etc.) and generates a command for the chosen task. Tasks specified by the *Planner* are executed by a set of low level controllers (*Navigator*, *Manipulator*, *Detector*, *Scanner* and *Answerer*) which return control to the *Planner* when a task termination state is reached. Since these subtasks are fairly independent, we can pretrain each controller independently, while assuming oracle versions of the remaining controllers. Our experiments show that this factorization enables higher accuracy and generalization to unseen environments.

Several question types require the agent to remember where it has been and what it has seen. For example, *How many pillows are in this house?* requires an agent to navigate around the rooms, open closets and keep track of the number of pillows it encounters. For sufficiently complex spaces, the agent needs to hold this information in memory for a long time. This motivates the need for an explicit external memory representation that is filled by the agent as it interacts with its environment. This memory must be both spatial and semantic so it can represent *what* is *where*. We propose a new recurrent layer formulation: Egocentric Spatial GRU (esGRU) to represent this memory (Sec 4.1).

Training and evaluating interactive agents in the real world is currently prohibitive from the standpoint of operating costs, scale and research reproducibility. A far more viable alternative is to train and evaluate such agents in realistic simulated environments. Towards this end, we present the Interactive Question Answering Dataset (IQUAD V1) built upon AI2-THOR [35], a photo-realistic customizable simulation environment for indoor scenes integrated with the Unity [1] physics engine. IQUAD V1 consists of over 75,000 multiple choice questions, each question accompanied by a unique scene configuration.

We evaluate HIMN on IQUAD V1 using a question answering accuracy metric and show that it outperforms a baseline based on a common architecture for reinforcement learning used in past work. We evaluate in both familiar and unfamiliar environments to show that our semantic model generalizes well across scenes.

In summary, our contributions include: (a) proposing Interactive Question Answering, the task of answering questions that require the agent to interact with a dynamic environment, (b) presenting the Hierarchical Interactive Memory Network, a question answering model factorized into a high level *Planner*, a set of low level controllers and a rich semantic spatial memory, (c) the Egocentric Spatial GRU, a new recurrent layer to represent this memory and (d) a new dataset IQUAD V1 towards the task of IQA.

2. Related Work

Visual Question Answering (VQA): VQA has seen significant progress over the past few years, owing to the design of deep architectures suited for this task and the creation of large VQA datasets to train these models [71]. These include datasets of natural images [2, 36, 41, 68], synthetic images [2, 4, 24, 27, 28], natural videos [23, 65], synthetic videos [32, 49] and multimodal contexts [29]. Some of these use questions written by humans [2, 28, 29, 36] and others use questions that are generated automatically [4, 24, 27]. IQUAD V1 is set in a photo-realistic simulation environment and uses automatically generated questions. In contrast to the aforementioned datasets that only require the agent to observe the content passively, IQUAD V1 requires the agent to interact with a dynamic environment.

The first deep architectures designed for VQA involved using an RNN to encode the question, using a CNN to encode the image and combining them using fully connected layers to yield the answer [2, 42]. More recently, modular networks [4, 20, 25] that construct an explicit representation of the reasoning process by exploiting the compositional nature of language have been proposed. Similar architectures have also been applied to the video domain with extensions such as spatiotemporal attention [23, 49]. Our proposed approach to question answering allows the agent to interact with its environment and is thus fundamentally different to past QA approaches. However, we note that approaches such as visual attention and modularity can easily be combined with our model to provide further improvements.

Reinforcement Learning (RL): RL algorithms have been employed in a wide range of problems including locomotion [33], obstacle detection [44] and autonomous flight [34, 59]. Of particular relevance to our approach is the area of hierarchical reinforcement learning (HRL), which consists of a high level controller and one or more low level controllers. The high-level controller selects a subtask to be executed and invokes one of the low level controllers. The advantage of HRL is that it allows the model to operate at multiple levels of temporal abstraction. Early works propos-

Interactive Question Answering Dataset Statistics		
	Train	Test
Existence	25,600	640
Counting	25,600	640
Spatial Relationships	25,600	640
Rooms	25	5
Total scene configurations (s.c.)	76,800	1,920
Avg # objects per (s.c.)	46	41
Avg # interactable objects (s.c.)	21	16
Vocabulary Size	70	70

Table 1. This table shows the statistics of our proposed dataset in a variety of question types, objects and scene configurations.

tions (*Is there an apple in the kitchen?*), Counting questions (*How many forks are present in the scene?*), and Spatial Relationship questions (*Is there lettuce in the fridge? / Is there a cup on the counter-top?*). Questions, ground truth answers, and answer choices are generated automatically. Since natural language understanding is not a focus of this dataset, questions are generated using a set of templates written down a priori. Extending IQUAD V1 to include more diverse questions generated by humans is future work. IQUAD V1 is a balanced dataset that prevents models from obtaining high accuracies by simply exploiting trivial language and scene configuration biases. Similar to past balanced VQA datasets [15], each question is associated with multiple scene configurations that result in different answers to the question. We split the 30 kitchen rooms into 25 train and 5 test, and have 1024 unique (question, scene configuration) pairs for each (room, question type) pair in train, and 128 in test. An episode is finished when the *Answerer* is invoked. We evaluate different methods using Top-1 accuracy.

3.3. Agent and Objects

The agent in our environments has a single RGB camera mounted at a fixed height. An agent can perform one of five navigation actions (move ahead 25 cm, rotate 90 degrees left or right, look up or down 30 degrees). We assume a grid-world floor plan that ensures that the agent always moves along the edges of a grid and comes to a stop on a node in this grid. The agent can perform two interaction actions (open and close) to manipulate objects. A wide variety of objects (fridges, cabinets, drawers, microwaves, etc.) can be interacted with. If there are multiple items in the current viewpoint which can be opened or closed, the environment chooses the one nearest to the center of the current image. The success of each action depends on the current state of the environment as well as the agent's current location. For instance, the agent cannot open a cabinet that is more than 1 meter away or is not in view, or is already open, and it cannot walk through a table or a wall.

4. Model

We propose HIMN (Hierarchical Interactive Memory Network), consisting of a hierarchy of controllers that operate at multiple levels of temporal abstraction and a rich semantic memory that aids in navigation, interaction, and question answering. Figure 2 provides an overview of HIMN. We now describe each of HIMN's components in greater detail.

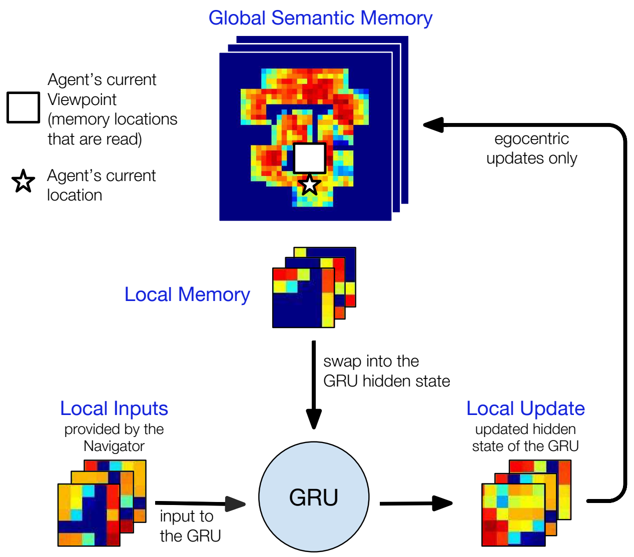
Global Semantic Memory
Agent's current Viewpoint (memory locations that are read)
Agent's current location
egocentric updates only
Local Memory
swap into the GRU hidden state
Local Inputs
provided by the Navigator
Local Update
updated hidden state of the GRU
input to the GRU
GRU

Figure 3. An overview of the Egocentric Spatial GRU (esGRU): The esGRU only allows writing to a local window within the memory, dependent on the agent's current location and viewpoint.

4.1. Spatial Memory

Several question types require the agent to keep track of objects that it has seen in the past along with their locations. For complex scenes with several locations and interactable objects, the agent needs to hold this information in memory for a long duration. This motivates the need for an explicit external memory representation that is filled by the agent on the fly and can be accessed at any time. To address this, HIMN uses a rich semantic spatial memory that encodes a semantic representation of each location in the scene. Each location in this memory consists of a feature vector encoding object detection probabilities, free space probability (a 2D occupancy grid), coverage (has the agent inspected this location before), and navigation intent (has the agent attempted to visit this location before). We propose a new recurrent layer formulation: Egocentric Spatial GRU (esGRU) to represent this memory, illustrated in Figure 3. The esGRU maintains an external global spatial memory represented as a 3D tensor. At each time step, the esGRU swaps in local egocentric copies of this memory into the hidden state of the GRU, performs computations using current inputs, and then swaps out the resulting hidden state into the global memory at the predetermined location. This speeds up computations and prevents corrupting the memory at locations far away from the agent's current viewpoint. When navigating and answering questions, the agent can access the full memory, enabling long-term recall from observations seen hundreds of states prior. Furthermore, only

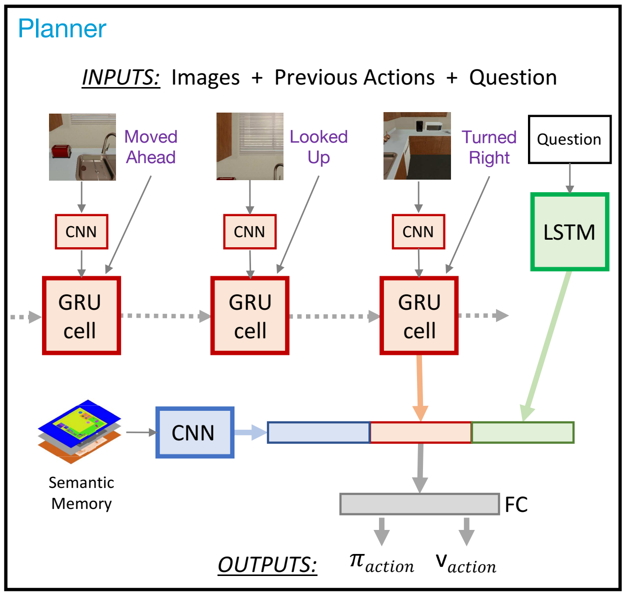
Planner
INPUTS: Images + Previous Actions + Question
Moved Ahead
Looked Up
Turned Right
Question
CNN
LSTM
GRU cell
Semantic Memory
FC
OUTPUTS: π_{action} v_{action}

Figure 4. Schematic representation of the *Planner*

low level controllers have read-write access to this memory. Since the *Planner* only makes high level decisions, without interacting with the world at a lower level, it only has read access to the memory.

4.2. Planner

The high level *Planner* invokes low level controllers in order to explore the environment, gather knowledge needed to answer the given question, and answer the question. We frame this as a reinforcement learning problem where the agent must issue the fewest possible commands that result in a correct answer. The agent must learn to explore relevant areas of the scene based on learned knowledge (e.g. apples are often in the fridge, cabinets are openable, etc.), the current memory state (e.g. the fridge is to the left), current observations (e.g. the fridge is closed) and the question. At every timestep, the *Planner* chooses to either invoke the *Navigator* providing a relative location in a 5x5 grid in front of the agent, invoke the *Scanner* with a direction such as up or left, invoke the *Manipulator* with open/close commands on a nearby object, or invoke the *Answerer* for a total of 32 discrete actions. It does this by producing a policy π consisting of probabilities π_i for each action, and a value v for the current state. π and v are learned using the A3C algorithm [46]. Figure 4 shows a schematic of the *Planner*. It consists of a GRU which accepts at each time step the current viewpoint (encoded by a CNN) and the previous action. The *Planner* has read only access to the semantic memory centered around the agent's current location. The output of this GRU is combined with the question embedding and an embedding of the nearby semantic spatial memory to predict π and v. The agent receives a fixed reward/penalty based on answering correctly/incorrectly. It is also provided a constant time penalty to encourage efficient explorations of the environment and quick answering, as well as a penalty for attempting to perform invalid actions. The agent is also given intermediate rewards for increasing the "coverage" of the environment, effectively training the network to maximize the amount of the room it has explored as quickly as possible. Finally, at each time step, the *Planner* also predicts which high level actions are viable given the current world state. In many locations in the scenes, certain navigation destinations are unreachable or there are no objects to interact with. Predicting possible/impossible actions at each time step, allows gradients to propagate through all actions rather than just the chosen action. This leads to higher accuracies and faster convergence (see section 5.2 for more details).

4.3. Low level controllers

Navigator The *Navigator* is invoked by the *Planner* which also provides it with the relative coordinates of the target location. Given a destination specified by the *Planner* and the current estimate of the room's occupancy grid, the *Navigator* runs A* search to find the shortest path to the goal. As the *Navigator* moves through the environment, it uses the esGRU to produce a local (5x5) occupancy grid given the current visual observation. This updates the global occupancy estimate, and prompts a new shortest-path computation. This is a fully supervised problem and can be trained with the standard sigmoid-cross-entropy. The *Navigator* also invokes the *Scanner* to obtain a wide angle view of the environment. Given that the requested destination may be outside the bounds of the room or otherwise impossible (e.g. at a wall or other obstacle), the *Navigator*'s network also predicts a termination signal, and returns control to the *Planner* when the prediction passes a certain threshold.
Scanner The *Scanner* is a simple controller which captures images by rotating the camera up, down, left, or right while maintaining the agent's current location. The *Scanner* calls the *Detector* on every new image.
Detector Object detection is a critical component of HIMN given that all questions in IQUAD V1 involve one or more objects in the room. We use YOLOv3 [56] fine-tuned on the AI2-THOR training scenes as an object detector. We estimate the depth of an object using the FRCN depth estimation network [38] and project the probabilities of the detected objects onto the ground plane. Both of these networks operate at real-time speeds, which is necessary since they are invoked on every new image. The detection probabilities are incorporated into the spatial memory using a moving average update rule. We also perform experiments where we substitute the trained detector and depth estimator with oracle detections. Detections provided by the environment still requires the network to learn affordances. For instance, the network must learn that *microwaves can be opened*, *apples can be in fridges*, etc.
Manipulator The *Manipulator* is invoked by the *Planner* to manipulate the current state of an object. For example, opening and closing the microwave. This leads to a change in the visual appearance of the scene. If the object is too far away or out of view, the action will fail.
Answerer The *Answerer* is invoked by the *Planner* to an-

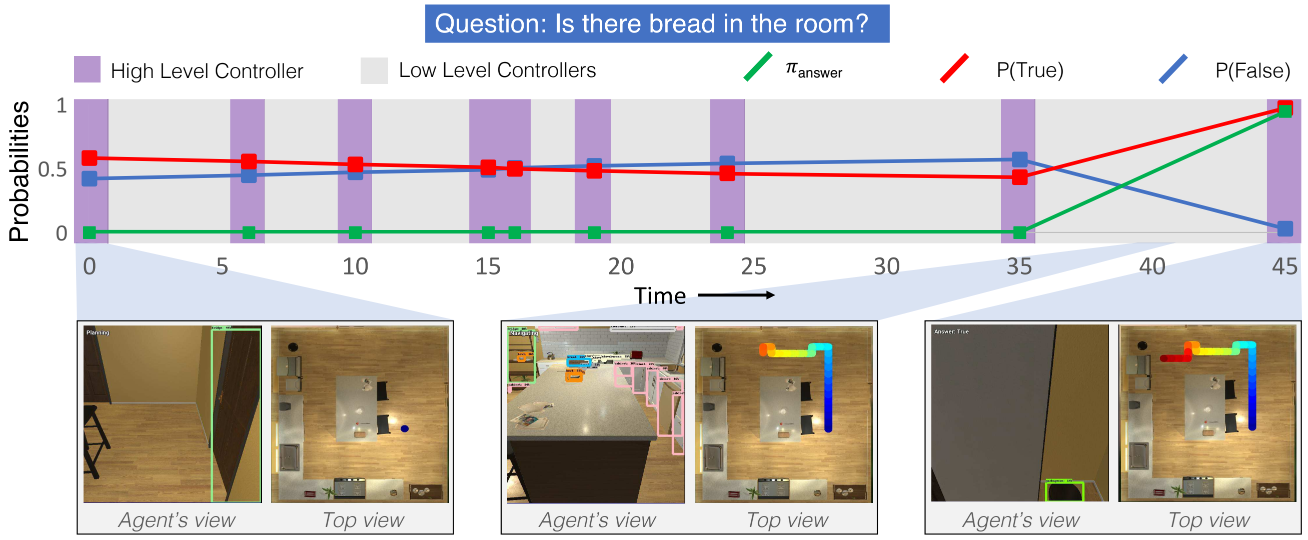

Figure 5. A sample trajectory for answering the existence question: *Is there bread in the room?* The purple sections indicate a *Planner* step, and the gray sections indicate a lower level controller such as the *Navigator* is controlling the agent. For a more detailed explanation, refer to section 5.4.

rooms, whereas the seen environments use the 25 training rooms but never-before-seen object placements and corresponding questions. Despite our relatively small number of training rooms, table 5 shows that our method only loses up to a few percentage points of accuracy when tested on unseen environments. This contrasts with many other end-to-end RL methods which learn deep features that tend to limit their applicability outside a known domain [73, 74]. Note that all experiments in previous sections were only performed on unseen environments.

5.4. Qualitative Results

Figure 5 shows a sample run of HIMN for the question "Is there bread in the room." Initially, $P(True)$ and $P(False)$ both start near 50%. The *Planner* begins searching the room by navigating around the kitchen table. During the initial exploration phase, bread is not detected, and $P(False)$ slowly increases. At timestep 39, the *Navigator* invokes the *Detector*, which sees the bread and incorporates it into the semantic spatial map. However, the *Navigator* does not return control to the *Planner*, as it has not yet reached the desired destination. Upon returning at timestep 45, the *Planner* reads the spatial map, sees the bread, and immediately decides it can answer the question. Thus π_{answer} and $P(True)$ both increase to nearly 100%. For more examples, please see our supplementary video `https://youtu.be/pXd3C-1jr98`.

5.5. Limitations

Although HIMN performs quite well, it still has several obvious limitations. Due to the 2D nature of the semantic spatial map, HIMN is unable to differentiate between an object being inside a container and being on top of the container. Two obvious extensions of HIMN are storing an explicit height parameter or using multiple 2D slices to construct a 3D map. Secondly, as can be seen in the human experiments in table 2, HIMN is still fairly inefficient at exploring the environment. We plan on investigating more traditional planning algorithms to reduce the time spent exploring previously searched areas. Finally, our templated language model is quite simple, and would not extend to arbitrary questions. We plan on extending IQUAD to include more varied questions, and we will use more expressive language embeddings like [45, 55] in future work.

6. Conclusion

In this work, we pose a new problem of Interactive Question Answering for several question types in interactive environments. We propose the Hierarchical Interactive Memory Network, consisting of a factorized set of controllers, allowing the system to learn from long trajectories. We also introduce the Egocentric Spatial GRU for updating spatial memory maps. The effectiveness of our proposed model is demonstrated on a new benchmark dataset built upon a high-quality simulation environment for this task. This dataset still presents several challenges to our model and baselines and warrants future research.

7. Acknowledgements

This work was funded in part by the National Science Foundation under contract number NSF-NRI-1637479, NSF-IIS-1338054, NSF-1652052, ONR N00014-13-1-0720, the Allen Distinguished Investigator Award, and the Allen Institute for Artificial Intelligence. We would like to thank Xun Huang for initial discussions and dataset prototypes. We would also like to thank NVIDIA for generously providing a DGX used for this research via the UW NVIDIA AI Lab (NVAIL).