This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Multi-Target Embodied Question Answering

Licheng Yu[1], Xinlei Chen[3], Georgia Gkioxari[3], Mohit Bansal[1],
Tamara L. Berg[1,3], Dhruv Batra[2,3]
[1]University of North Carolina at Chapel Hill [2]Georgia Tech [3]Facebook AI

Abstract

Embodied Question Answering (EQA) is a relatively new task where an agent is asked to answer questions about its environment from egocentric perception. EQA as introduced in [8] makes the fundamental assumption that every question, e.g. *"what color is the car?", has exactly* ***one*** *target ("car") being inquired about. This assumption puts a direct limitation on the abilities of the agent.*

We present a generalization of EQA – Multi-Target EQA (MT-EQA). Specifically, we study questions that have ***multiple*** *targets in them, such as "Is the dresser in the bedroom bigger than the oven in the kitchen?", where the agent has to navigate to multiple locations ("dresser in bedroom", "oven in kitchen") and perform comparative reasoning ("dresser" bigger than "oven") before it can answer a question. Such questions require the development of entirely new modules or components in the agent. To address this, we propose a modular architecture composed of a program generator, a controller, a navigator, and a VQA module. The program generator converts the given question into sequential executable sub-programs; the navigator guides the agent to multiple locations pertinent to the navigation-related sub-programs; and the controller learns to select relevant observations along its path. These observations are then fed to the VQA module to predict the answer. We perform detailed analysis for each of the model components and show that our joint model can outperform previous methods and strong baselines by a significant margin. Project page: https://embodiedqa.org.*

1. Introduction

One of the grand challenges of AI is to build intelligent agents that visually perceive their surroundings, communicate with humans via natural language, and act in their environments to accomplish tasks. In the vision, language, and AI communities, we are witnessing a shift in focus from *internet vision* to *embodied AI* – with the creation of new tasks and benchmarks [7, 2, 14, 35], instantiated on new simulation platforms [21, 28, 32, 33, 20, 6].

The focus of this paper is one such embodied AI task, Embodied Question Answering (EQA) [8], which tests an agent's overall ability to jointly perceive its surrounding, communicate with humans, and act in a physical environment. Specifically, in EQA, an agent is spawned in a random location within an environment and is asked a question about something in that environment, for example *"What color is the lamp?"*. In order to answer the question correctly, the agent needs to parse and understand the question, navigate to a good location (looking at the "lamp") based on its first-person perception of the environment and predict the right answer (*e.g.* "blue").

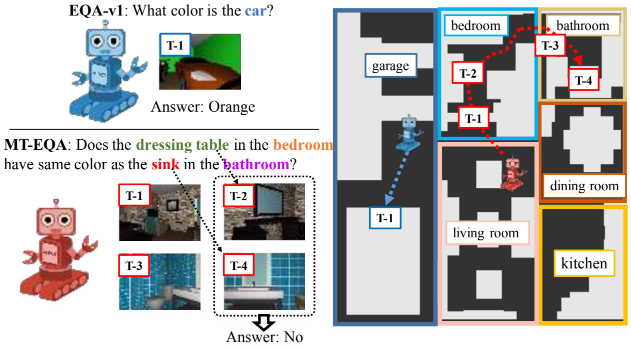

EQA-v1: What color is the car?
T-1
Answer: Orange
MT-EQA: Does the dressing table in the bedroom have same color as the sink in the bathroom?
T-1
T-2
T-3
T-4
Answer: No
garage
T-1
bedroom
T-2
T-1
bathroom
T-3
T-4
dining room
living room
kitchen

Figure 1: Difference between EQA-v1 and MT-EQA. While EQA-v1's question asks about a single target "car", MT-EQA's question involves multiple targets (e.g., bedroom, dressing table, bathroom, sink) to be navigated, and attribute comparison between multiple targets (e.g., dressing table and sink).

However, there is still much left to be done in EQA. In its original version, the EQA-v1 dataset only consists of single-target question-answer pairs, such as *"What color is the car?"*. The agent just needs to find the car then check its color based on its last observed frames. However, the single target constraint places a direct limitation on the possible set of tasks that the AI agent can tackle. For example, consider the question *"Is the kitchen larger than the bedroom?"* in EQA-v1; the agent would not be able to answer this question because it involves navigating to multiple targets –"kitchen" *and* "bedroom" – and the answer requires comparative reasoning between the two rooms, where all of these skills are not part of the original EQA task.

In this work, we present a generalization of EQA –

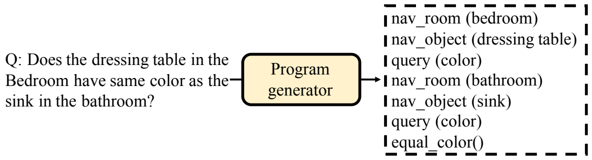

Q: Does the dressing table in the Bedroom have same color as the sink in the bathroom?
Program generator
nav_room (bedroom)
nav_object (dressing table)
query (color)
nav_room (bathroom)
nav_object (sink)
query (color)
equal_color()

Figure 2: Program Generator.

multi-target EQA (MT-EQA). Specifically, we study questions that have multiple implicit targets in them, such as *"Is the dresser in the bedroom bigger than the oven in the kitchen?"*. At a high-level, our work is inspired by the visual reasoning work of Neural Modular Networks [4] and CLEVR [18]. These works study compositional and modular reasoning in a fully-observable environment (an image). Our work may be viewed as embodied visual reasoning, where an agent is asked a question involving multiple modules and needs to gather information before it can execute them. In MT-EQA, we propose 6 types of compositional questions which compare attribute properties (color, size, distance) between multiple targets (objects/rooms). Fig. 1 shows an example from the MT-EQA dataset and contrasts it to the original EQA-v1 dataset.

The assumption in EQA-v1 of decoupling navigation from question-answering not only makes the task simpler but is also reflected in the model used – the EQA-v1 model simply consists of an LSTM navigator which after stopping, hands over frames to a VQA module. In contrast, MT-EQA introduces new modeling challenges that we address in this work. Consider the MT-EQA question in Fig. 1 – *"Does the table in the bedroom have same color as the sink in the bathroom?"*. From this example, it is clear that not only is it necessary to have a tighter integration between navigator and VQA, but we also need to develop fundamentally new modules. An EQA-v1 [8] agent would navigate to the final target location and run the VQA module based on its last sequence of frames along the path. In this case, only the "sink" would be observed from the final frames but dressing table would be lost. Instead, we propose a new model that consists of 4 components: (a) a program generator, (b) a navigator, (c) a controller and (d) a VQA module. The program generator converts the given question into sequential executable sub-programs, as shown in Fig. 2. The controller executes these sub-programs sequentially and gives control to the navigator when the navigation sub-programs are invoked (*e.g.* `nav_room(bedroom)`). During navigation, the controller processes the first-person views observed by the agent and predicts whether the target of the sub-program (*e.g.* bedroom) has been reached. In addition, the controller extracts cues pertinent to the questioned property of the sub-target, *e.g.* `query(color)`. Finally, these cues are fed into the VQA module which deals with the comparison of different attributes, *e.g.* executing `equal_color()` by comparing the color of dressing table and sink (Fig. 1).

Empirically, we show results for our joint model and analyze the performance of each of our components. Our full model outperforms the baselines under almost every navigation and QA metric by a large margin. We also report performance for the navigator, the controller, and the VQA module, when executed separately in an effort to isolate and better understand the effectiveness of these components. Our ablation studies show that our full model is better at all sub-tasks, including room navigation, object navigation and final EQA accuracy. Additionally, we find quantitative evidence that MT-EQA questions on closer targets are relatively easier to solve as they require shorter navigation, while questions for farther targets are harder.

2. Related Work

Our work relates to research in embodied perception and modular predictive models for program execution.

Embodied Perception. Visual recognition from images has witnessed tremendous success in recent years with the advent of deep convolutional neural networks (CNNs) [22, 31, 15] and large-scale datasets, such as ImageNet [26] and COCO [24]. More recently, we are beginning to witness a resurgence of *active vision*. For example, end-to-end learning methods successfully predict robotic actions from raw pixel data [23]. Gupta *et al.* [14] learn to navigate via mapping and planning. Sadeghi & Levine [27] teach an agent to fly in simulation and show its performance in the real world. Gandhi *et al.* [11] train self-supervised agents to fly from examples of drones crashing.

At the intersection of active perception and language understanding, several tasks have been proposed, including instruction-based navigation [7, 2], target-driven navigation [36, 14], embodied question answering [8], interactive question answering [13], and task planning [35]. While these tasks are driven by different goals, they all require training agents that can perceive their surroundings, understand the goal – either presented visually or in language instructions – and act in a virtual environment. Furthermore, the agents need to show strong generalization ability when deployed in novel unseen environments [14, 32].

Environments. There is an overbearing cost to developing real-world interactive benchmarks. Undoubtedly, this cost has hindered progress in studying embodied tasks. On the contrary, virtual environments that offer rich, efficient simulations of real-world dynamics, have emerged as promising alternatives to potentially overcome many of the challenges faced in real-world settings.

Recently there has been an explosion of simulated 3D environments in the AI community, all tailored towards different skill sets. Examples include ViZDoom [20], TorchCraft [30] and DeepMind Lab [5]. Just in the last year, simulated environments of semantically complex, realistic 3D scenes have been introduced, such as

HoME [6], House3D [32], MINOS [28], Gibson [33] and AI2THOR [21]. In this work, we use House3D, following the original EQA task [8]. House3D is a rich, interactive 3D environment based on human-designed indoor scenes sourced from SUNCG [29].

Modular Models. Neural module networks were originally introduced for visual question answering [4]. These networks decompose a question into several components and dynamically assemble a network to compute the answer, dealing with variable compositional linguistic structures. Since their introduction, modular networks have been applied to several other tasks: visual reasoning [16, 19], relationship modeling [17], embodied question answering [9], multitask reinforcement learning [3], language grounding on images [34] and video understanding [12]. Inspired by [10, 19], we cast EQA as a partially observable version of CLEVR and extend the modular idea to this task, which we believe requires an increasingly modular model design to address visual reasoning within a 3D environment.

3. Multi-Target EQA Dataset

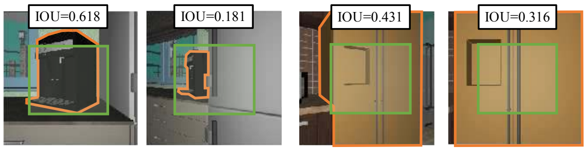
We now describe our proposed Multi-Target Embodied Question Answering (MT-EQA) task and associated dataset, contrasting it against EQA-v1. In v1 [8], the authors select 750 (out of about 45,000) environments for the EQA task. Four types of questions are proposed, each questioning a property (color, location, preposition) of a single target (room, object), as shown at the top of Table. 1. Our proposed MT-EQA task generalizes EQA-v1 and involves comparisons of various attributes (color, size, distance) between multiple targets, shown at the bottom of Table. 1. Next, we describe in detail the generation process, as well as useful statistics of MT-EQA.

3.1. Multi-Target EQA Generation

We generate question-answer pairs using the annotations available on SUNCG. We use the same number of rooms and objects as EQA-v1 (see Figure 2 in [8]). Each question in MT-EQA is represented as a series of functional programs, which can be executed on the environment to yield a ground-truth answer. The functional programs consist of some elementary operations, e.g., select(), unique(), object_color_pair(), query(), *etc*., that operate on the room and object annotations.

Each question type is associated with a question template and a sequence of operations. For example, consider the question type in MT-EQA object_color_compare, whose template is *"Does <OBJ1> share same color as <OBJ2> in <ROOM>?"*. Its sequence of elementary operations is: select(rooms) → unique(rooms) → select(objects) → unique(objects) → pair(objects) → query(color_compare).

The first function, select(rooms), returns all rooms in the environment. The second function, unique(rooms), selects a single unique room from the list to avoid ambiguity. Similarly, the third function, select(objects), and fourth function, unique(objects), return unique objects in the selected room. The fifth function, pair(objects), pairs the objects. The final function, query(color_compare), compares their colors.



(a) coffee machine (b) refrigerator

Figure 3: IOU between the target's mask and the centered rectangle mask. Higher IOU is achieved when the target has larger portion in the center of the view.

We design 6 types of questions comparing different attributes between objects (inside same room/across different rooms), distance comparison, and room size comparison. All question types and templates are shown in Table 2.

In some cases, a question instantiation returned from the corresponding program, as shown above, might not be executable, as rooms might be disconnected or not reachable. To check if a question is feasible, we execute the corresponding `nav_room()` and `nav_object()` programs and compute shortest paths connecting the targets in the question. If there is no path[1], it means the agent would not be able to look at all targets starting from its given spawn location. We filter out such impossible questions.

For computing the shortest path connecting the targets, we need to find the position (x, y, z, yaw) that best views each target. In order to do so, we first sample 100 positions near the target. For each position, we pick the yaw angle that looks at the target with the highest Intersection-Over-Union (IOU), computed using the target's mask[2] and a centered rectangular mask. Fig. 3 shows 4 IOU scores of *coffee machine* and *refrigerator* from different positions. We sort the 100 positions and pick the one with highest IOU as the best-view position of the target, which is used to connect the shortest-path. For each object, its highest IOU value IOU_{best} is recorded for evaluation purposes (as a reference of the target's best-view).

To minimize the bias in MT-EQA, we perform entropy-filtering, similar to [8]. Specifically for each unique question, we compute its answer distribution across the whole dataset. We exclude questions whose normalized answer distribution entropy is below 0.9[3]. This prevents the agent from memorizing easy question-answer pairs without looking at the environment. For example, the answer to *"is the*

[1] This is a result of noisy annotations in SUNCG and inaccurate occupancy maps due to the axis-aligned assumption returned by House3D.

[2] House3D returns the the ground-truth semantic segmentation for each first-person view.

[3] Rather than 0.5 in [8], we set the normalized entropy threshold as 0.9 (maximum is 1) since all of our questions have binary answers.

	Question Type	Template
EQA-v1	location	*“What room is the <OBJ> located in?”*
	color	*“What color is the <OBJ>?”*
	color_room	*“What color is the <OBJ> in the <ROOM>?”*
	preposition	*“What is <on/above/below/next-to> the <OBJ> in the <ROOM>?”*
MT-EQA	object_color_compare_inroom	*“Does <OBJ1> share same color as <OBJ2> in <ROOM>?”*
	object_color_compare_xroom	*“Does <OBJ1> in <ROOM1> share same color as <OBJ2> in <ROOM2>?”*
	object_size_compare_inroom	*“Is <OBJ1> bigger/smaller than <OBJ2> in <ROOM>?”*
	object_size_compare_xroom	*“Is <OBJ1> in <ROOM1> bigger/smaller than <OBJ2> in <ROOM2>?”*
	object_dist_compare	*“Is <OBJ1> closer than/farther from <OBJ2> than <OBJ3> in <ROOM>?”*
	room_size_compare	*“Is <ROOM1> bigger/smaller than <ROOM2> in the house?”*

Table 1: Question types and the associated templates used in EQA-v1 and MT-EQA.

Question Type	Functional Form
object_color_compare	select(rooms) → unique(rooms) → select(objects) → unique(objects) → pair(objects) → query(color_compare)
object_size_compare	select(rooms) → unique(rooms) → select(objects) → unique(objects) → pair(objects) →query(size_compare)
object_dist_compare	select(rooms) → unique(rooms) → select(objects) → unique(objects) → triplet(objects) →query(dist_compare)
room_size_compare	select(rooms) → unique(rooms) → pair(rooms) → query(size_compare)

Table 2: Functional forms of all question types in the MT-EQA dataset. Note that for each object color/size comparison question type, there exists two modes: inroom and xroom, depending on whether the two objects are in the same room or not. For example, object_color_compare_xroom compares the color of two objects in two different rooms.

	random	q-LSTM	q-NN	q-BoW	“no”
Test Acc. (%)	49.44	48.24	53.74	49.22	53.28

Table 3: EQA (test) accuracy using questions and priors.

	Houses	Unique questions	Total questions
train	486	2,030	14,495
val	50	938	1,954
test	52	1,246	2,838

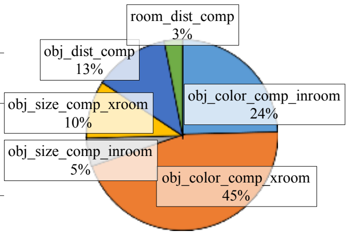


Figure 4: Overview of MT-EQA dataset including split statistics and question type distribution.

bed in the living room bigger than the cup in the kitchen?” is always *Yes*. Such questions are excluded from our dataset. After the two filtering stages, the MT-EQA questions are both balanced and feasible.

In addition, we check if MT-EQA is easily addressed by question-only or prior-only baselines. For this, we evaluate four question-based models: (a) an LSTM-based question-to-answer model, (b) a nearest neighbor (NN) baseline that finds the NN question from the training set and uses its most frequent answer as the prediction, (c) a bag-of-words (BoW) model that encodes a question followed by a learned linear classifier to predict the answer and (d) a naive “no” only answer model, since “no” is the most frequent answer by a slight margin. Table. 3 shows the results. There exists very little bias on the “yes/no” distribution (53.28%), and all question-based models make close to random predictions. In comparison, and as we empirically show in Sec. 5, our results are far better than these baselines, indicating the necessity to explore the environment in order to answer the question. Besides, the results also address the concern in [1] where language-only models (BoW and NN) already form competitive baselines for EQA-v1. In MT-EQA, these baselines perform close to chance as a result of the balanced binary question-answer pairs in MT-EQA.

Overall, our MT-EQA dataset consists of 19,287 questions across 588 environments[4], referring to a total of 61 unique object types in 8 unique room types. Fig. 4 shows the question type distribution. Approximately 32 questions are asked for each house on average, 209 at most and 1 at fewest. There are relatively fewer object_size_compare and room_size_compare questions as many frequently occurring comparisons are too easy to guess without exploring the environment and thus fail the entropy filtering. We will release the MT-EQA dataset and the generation pipeline.

4. Model

Our model is composed of 4 modules: the question-to-program generator, the navigator, the controller, and the VQA module. We describe these modules in detail.

4.1. Program Generator

The program generator takes the question as input and generates sequential programs for execution. We define

[4] The 588 environments are subset of EQA-v1’s. Some environments are discarded due to entropy filtering and unavailable paths.

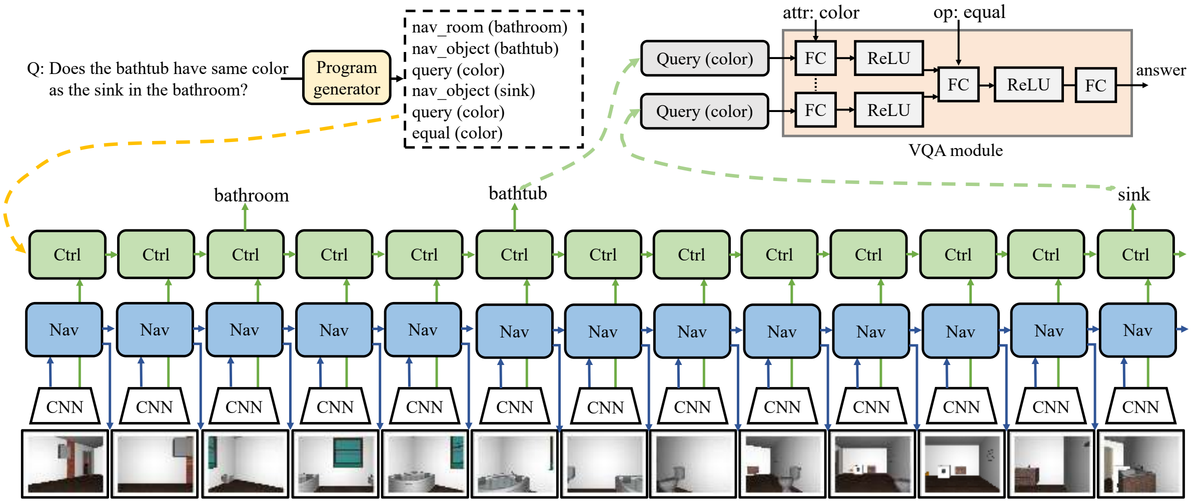
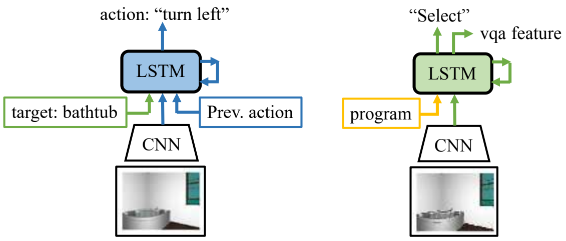


Figure 5: Model architecture: our model is composed of a program generator, a navigator, a controller, and a VQA module.

1) `nav_object(phrase)` 2) `nav_room(phrase)`
3) `query(color / size / room_size)`
4) `equal_color()`
5) `object_size_compare(bigger / smaller)`
6) `object_dist_compare(farther / closer)`
7) `room_size_compare(bigger / smaller)`

Table 4: MT-EQA executable programs.



(a) Navigator (b) Controller

Figure 6: Navigator and Controller.

7 types of executable programs for the MT-EQA task in Table. 4. For example, *"Is the bathtub the same color as the sink in the bathroom?"* is decomposed into a series of sequential sub-programs: `nav_room(bathroom)` → `nav_object(bathtub)` → `query_color()` → `nav_object(sink)` → `query_color()` → `equal_color()`. Similar to CLEVR [18], the question programs are automatically generated in a templated manner (Table. 2), making sub-component decomposition (converting questions back to programs) simple (Table. 4). We use template-based rules by selecting and filling in the arguments in Table. 4 to generate the programs (which is always accurate). While a neural model could also be applied, a learned program generator is not the focus of our work.

4.2. Navigator

The navigator executes the `nav_room()` and `nav_object()` programs. As shown in Fig. 6(a), we use an LSTM as our core component. At each time step, the LSTM takes as inputs the current egocentric (first-person view) image, an encoding of the target phrase (*e.g.* "bathtub" if the program is `nav_object(bathtub)`), and the previous action, in order to predict the next action.

The navigator uses a CNN feature extractor that takes a 224x224 RGB image returned from the House3D renderer, and transforms it into a visual feature, which is then fed into the LSTM. Similar to [8], the CNN is pre-trained under a multi-task framework consisting of three tasks: RGB-value reconstruction, semantic segmentation, and depth estimation. Thus, the extracted feature contains rich information about the scene's appearance, content, and geometry (objects, color, texture, shape, and depth). In addition to the visual feature, the LSTM is presented with two additional inputs. The first is the target embedding, where we use the average embedding of GloVE vectors [25] over words describing the target. The second is previous action, which is in the form of a look-up from an action embedding matrix.

We want to note the different perceptual skills required for room and object navigation: Room navigation relies on understanding the overall scene and finding cross-room paths (entry/exit), while object navigation requires localizing the target object within a room and finding a path to reach it. To capture the difference, we implement two separate navigation modules, `nav_room()` and `nav_object()` respectively. These two modules share same architecture but are trained separately for different targets.

In MT-EQA, the action space for navigation consists of 3 action types: turning left (30 degrees), turning right (30 degrees), and moving forward. This is almost the same as

EQA-v1 [8], except we use larger turning angles – as our navigation paths are much longer due to the multi-target setting. We find that this change reduces the number of actions required for navigation, leading to easier training.

4.3. Controller

The controller is the central module in our model, as it connects all of the other modules by: 1) creating a plan from the program generator, 2) collecting the necessary observations from the navigator, and 3) invoking the VQA module.

Fig. 6 (b) shows the controller, whose key component is another LSTM. Consider the question *"Does the bathtub have same color as the sink in the bathroom?"* with part of its program as example – `nav_room(bathroom)` → `nav_object(bathtub)`. The controller starts by calling the room navigator to look for "bathroom". During navigation, the controller keeps track of the first-person views, looking for the target. Particularly, it extracts the features via CNN which are then fused with the target embedding as input to the LSTM. The controller predicts SELECT if the target is found, stopping the current navigator, in our example `nav_room(bathroom)`, and starting execution of the next program, `nav_object(bathtub)`.

Finally, after the object target "bathtub" has been found, the next program – `query_color()`, is executed. The controller extracts attribute features from the first-person view containing the target. In all, there are three attribute types in MT-EQA - object's color, object's size, and room's size. Again, we treat object and room differently in our model. For object-specific attributes, we use the hidden state of the controller at the location where SELECT was predicted. This state should contain semantic information for the target, as it is where the controller is confident the target is located. For room-specific attributes, the controller collects a panorama by asking the navigator to rotate 360 degrees (by performing 12 turning-right actions) at the location where SELECT is predicted. The CNN features from this panorama view are concatenated as the representation.

During program execution by the controller, the extracted cues for all the targets are stored, and in the end they are used by the VQA module to predict the final answer.

4.4. VQA Module

The final task requires comparative reasoning, *e.g.*, `object_size_compare(bigger)`, `equal_color()`, *etc*. When the controller has gathered all of the targets for comparison, it invokes the VQA module. As shown in top-right of Fig. 5, the VQA module embeds the stored features of multiple targets into the question-attribute space, using a FC layer followed by ReLU. The transformed features are then concatenated and fed into another FC+ReLU which is conditioned on the comparison operator (equal, bigger than, smaller than, *etc*.). The output is a binary prediction (yes/no) for that attribute comparison. We call it compositional VQA (cVQA). The cVQA module in Fig. 5 depicts a two-input comparison as an example, but our cVQA module also extends to three inputs, for questions like *"Is the refrigerator closer to the coffee machine than the microwave?"*.

4.5. Training

Training follows a two-stage approach: First, the full model is trained using Imitation Learning (IL); Second, the navigator is further fine-tuned with Reinforcement Learning (RL) using policy gradients.

First, we jointly train our full model using imitation learning. For imitation learning, we treat the shortest paths and the key positions containing the targets as our ground-truth labels for navigation and for the controller's SELECT classifier, respectively. The objective function consists of a navigation objective and a controller objective at every time step t, and a VQA objective at the final step. For the i-th question, let $P^{nav}_{i,t,a}$ be action a's probability at time t, $P^{sel}_{i,t}$ be the controller's SELECT probability at time t, and P^{vqa}_i be the answer probability from VQA, then we minimize the combined loss:

$$
\begin{aligned}
L &= L_{nav} + \alpha L_{ctrl} + \beta L_{vqa} \\
&= \underbrace{-\sum_i \sum_t \sum_a y^n_{i,t,a} \log P^{nav}_{i,t,a}}_{\text{Cross-entropy on navigator action}} \\
&\quad \underbrace{\text{-}\alpha \sum_i \sum_t (y^c_{i,t} \log P^{sel}_{i,t} + (1 - y^c_{i,t}) \log(1 - P^{sel}_{i,t}))}_{\text{Binary cross-entropy on controller's SELECT}} \\
&\quad \underbrace{\text{-}\beta \sum_i (y^v_i \log P^{vqa}_i + (1 - y^v_i) \log(1 - P^{vqa}_i)).}_{\text{Binary cross-entropy on VQA's answer}}
\end{aligned}
$$

Subsequently, we use RL to fine-tune the room and object navigators.

We provide two types of reward signals to the navigators. The first is a dense reward, corresponding to the agent's progress toward the goal (positive if moving closer to the target and negative if moving away). This reward is measured by the distance change in the 2D bird-view distance space, clipped to lie within $[-1.0, 1.0]$. The second is a sparse reward that quantifies whether the agent is looking at the target object when the episode is terminated. For object targets, we compute IOU_T between the target's mask and the centered rectangle mask at termination. We use the best IOU score of the target IOU_{best} as reference and compute the ratio $\frac{\text{IOU}_T}{\text{IOU}_{best}}$. If the ratio is greater than 0.5, we set the reward to 1.0 otherwise -1.0. For room targets, we assign reward 0.2 to the agent if it is inside the target room at termination, otherwise -0.2.

		Object Navigation					Room Navigation		EQA				
		d_T	d_Δ	h_T	IOU_T^r	$\%\mathrm{stop}_o$	$\%\mathrm{r}_T$	$\%\mathrm{stop}_r$	ep_len	$\%_{easy}$	$\%_{medium}$	$\%_{hard}$	$\%_{overall}$
1	Nav+cVQA	5.41	-0.64	0.19	0.15	36	34	60	153.13	58.42	53.29	51.46	53.24
2	Nav(RL)+cVQA	3.80	0.10	0.33	**0.30**	46	40	62	144.80	67.57	55.91	53.28	57.40
3	Nav+Ctrl+cVQA	5.25	-0.56	0.20	0.18	36	37	70	145.20	59.73	53.48	49.04	54.44
4	Nav(RL)+Ctrl+cVQA	**3.60**	**0.16**	**0.33**	0.29	**48**	**43**	**72**	**127.71**	**72.22**	**59.97**	**54.92**	**61.45**

Table 5: Quantitative evaluation of object/room navigation and EQA accuracy for different approaches.

		object_color_compare		object_size_compare		object_dist_compare	room_size_compare	$\%_{overall}$
		inroom	xroom	inroom	xroom	inroom	xroom	
1	Nav+cVQA	64.15	52.47	57.85	55.68	49.38	48.37	53.24
2	Nav(RL)+cVQA	71.24	53.92	74.38	60.81	51.23	46.66	57.40
3	Nav+Ctrl+cVQA	66.41	52.65	57.85	53.48	49.38	48.37	54.44
4	Nav(RL)+Ctrl+cVQA	**72.68**	**58.19**	**76.86**	**63.37**	**54.94**	**55.57**	**61.45**

Table 6: EQA accuracy on each question type for different approaches.

		object_color_compare		object_size_compare		object_dist_compare	room_size_compare	$\%_{overall}$
		inroom	xroom	inroom	xroom	inroom	xroom	
1	[BestView] + attn-VQA (cnn)	71.16	59.56	65.29	65.93	58.64	49.74	60.50
2	[BestView] + cVQA (cnn)	82.92	72.70	80.99	83.88	**69.75**	64.32	74.14
3	[ShortestPath+BestView] + Ctrl + cVQA	**90.70**	**85.49**	**82.64**	**88.64**	68.52	**71.87**	**82.88**
4	[ShortestPath] + seq-VQA	53.32	54.44	51.24	50.55	47.53	49.74	52.36
5	[ShortestPath] + Ctrl + cVQA	76.09	69.11	75.21	79.49	64.20	61.23	69.77

Table 7: EQA accuracy of different approaches on each question type in oracle setting (given shortest path or best-view images).

5. Experiments

In this section we describe our experimental results. Since MT-EQA is a complex task and our model is modular, we will show both the final results (QA accuracy) and the intermediate performance (for navigation). Specifically, we first describe our evaluation setup and metrics for MT-EQA. Then, we report the comparison of our model against several strong baselines. And finally, we analyze variants of our model and provide ablation results.

5.1. Evaluation Setup and Metrics

Spawn Location. MT-EQA questions involve multiple targets (rooms/objects) to be found. To prevent the agent from learning biases due to spawn location, we randomly select one of the mentioned targets as reference and spawn our agent 10 actions (typically 1.9 meters) away.

EQA Accuracy. We compute overall accuracy as well as accuracy for each of the 6 types of questions in our dataset. In addition, we also categorize question difficulty level into easy, medium, and hard by binning the ground-truth action length. Easy questions are those with fewer than 25 action steps along the shortest path, medium are those with 25-70 actions, and hard are those with more than 70 actions. We report accuracy for each difficulty, $\%_{easy}$, $\%_{medium}$, $\%_{hard}$, as well as overall, $\%_{overall}$, in Table 5.

Navigation Accuracy. We also measure the navigation accuracy for both objects and rooms in MT-EQA. As each question involves several targets, the order of them being navigated matters. We consider the 'ground truth' ordering of targets for navigation as the order in which they are mentioned in the question, e.g., given *"Does the bathtub have same color as the sink?"*, the agent is trained and evaluated for visiting the "bathtub" first and then the "sink".

For each mentioned target object, we evaluate the agent's navigation performance by computing the distance to the target object at navigation termination, d_T, and change in distance to the target from initial spawned position to terminal position, d_Δ. We also compute the stop ratio $\%stop_o$ as in EQA-v1 [8]. Additionally, we propose two new metrics based on the IOU of the target object at its termination. When the navigation is done, we compute the IOU of the target w.r.t a centered rectangular box (see Fig. 3 as example). The first metric is mean IOU ratio $\mathrm{IOU}_T^r = \frac{1}{N}\sum_i \frac{\mathrm{IOU}_T(o_i)}{\mathrm{IOU}_{best}(o_i)})$ where $\mathrm{IOU}_{best}(o_i)$ is the highest IOU score for object o_i. The second is hit accuracy h_T – we compute the percentage of the ratio $\mathrm{IOU}_T(o_i)/\mathrm{IOU}_{best}(o_i)$ greater than 0.5, i.e., $h_T = \frac{1}{N}\sum_i ||\frac{\mathrm{IOU}_T(o_i)}{\mathrm{IOU}_{best}(o_i)} > 0.5||$. Both metrics measure to what extent the agent is looking at the target at termination.

For each mentioned target room, we evaluate the agent's navigation by recording the percentage of agents terminating inside the target room $\%r_T$ and the stop ratio $\%stop_r$.

For all the above metrics except for d_T, larger is better. Additionally, we report the overall number of action steps (episode length) executed for each question, i.e., ep_len.

5.2. EQA Results

Nav+Ctrl+cVQA is our full model, which is composed of a program generator, a navigator, a controller and a comparative VQA module. Another variant of our model, the REINFORCE fine-tuned model is denoted as Nav(RL)+Ctrl+cVQA. We also train a simplified version of our full model, Nav+cVQA. which does not use a controller. For this model, we let the navigator predict termination whenever a target is detected, then feed its hidden states to the VQA model. The training details are similar to our full model for both IL and RL. We show comparisons of both navigation and EQA accuracy in Table. 5.

RL helps both navigation and EQA accuracies. Both object and room navigation performance are improved after RL finetuning. We notice without finetuning $d_\triangle$ for both models (Row 1 &3) are negative, which means the agent has moved farther away from the target during navigation. After RL finetuning, $d_\triangle$ jumps from -0.56 to 0.16 (Row 3 & 4). The hit accuracy also improves from 20% to 33%, indicating that the RL-finetuned agent is more likely to find the target mentioned in the question. Episode lengths from the stronger navigators are shorter, indicating that better navigators find their target more quickly. And, higher EQA accuracy is also achieved with the help of RL finetuning (from 54.44% to 61.45%). After breaking down the EQA into different types, we observe the same trend in Table. 6 – our full model with RL far outperforms the others.

Controller is important. Comparing our full model (Row 4) to the one without a controller (Row 2), we notice that the former outperforms the latter across almost all the metrics. One possible reason is that the VQA task and navigation task are quite different, such that the features (hidden state) from the navigator cannot help improve the VQA module. On the contrary, our controller decouples the two tasks, letting the navigator and VQA module focus on their own roles.

Questions with shorter ground-truth path are easier. We observe that our agent is far better at dealing with easy questions than hard ones (72.22% over 54.92% in Table. 5 Row 4). One reason is that the targets mentioned in the easy questions, e.g., sink and toilet in *"Does the sink have same color as the toilet in the bathroom?"*, are typically closer to each other, thus are relatively easier to be explored, whereas questions like *"Is the kitchen bigger than the garage?"* requires a very long trajectory and the risk of missing one (kitchen or garage) is increased. The same observation is found in Table. 6, where we get higher accuracy for "in-room" questions than "cross-room" ones.

5.3. Oracle Comparisons

To better understand each module of our model, we run ablation studies. Table. 7 shows EQA accuracy of different approaches given the shortest paths or best-view frames.

Our VQA module helps. We first compare the performance of our VQA module against an attention-based VQA. Given the best view of each target, we can directly feed the features from those images to the VQA module, using the CNN features instead of hidden states from controller side. The attention-based VQA architecture is similar to [8], which uses an LSTM to encode questions and then uses its representation to pool image features with attention. Comparing the two methods in Table. 7, Row 1 & 2, our VQA module achieves 13.64% higher accuracy. The benefit mainly comes from the decomposition of attribute representation and comparison in our VQA module.

Controller's features help. We compare the controller's features to raw CNN features for VQA. When given both shortest path and best-view position, we run our full model with these annotations and feed the hidden states from the controller's LSTM to our VQA model. As shown in Table. 7, Row 2 & 3, the controller's features are far better than raw CNN features, especially for object_color_compare and object_size_compare question types.

Controller's SELECT matters. Our controller predicts SELECT and extracts the features at that moment. One possible question is how important is this moment selection. To demonstrate its advantage, we trained another VQA module which uses a LSTM to encode the whole sequence of frames along the shortest path and uses its final hidden state to predict the answer, denoted as seq-VQA. The hypothesis is that the final hidden state might be able to encode all relevant information, as the LSTM has gone through the whole sequence of frames. Table. 7, Row 4, shows its results, which is nearly random. On the contrary, when controller is used to SELECT frames in Row 5, the results are far better. However, there is still much space for improvement. Comparing Table. 7, Row 3 & 5, the overall accuracy drops 13% when using features from the predicted SELECT instead of oracle moments, and 20% when using additional navigators (comparing Table. 7, Row 3, & Table. 6, Row 4), indicating the necessity of both accurate SELECT and navigation.

6. Conclusion

We proposed MT-EQA, extending the original EQA questions from a limited single-target setting to a more challenging multi-target setting, which requires the agent to perform comparative reasoning before answering questions. We collected a MT-EQA dataset as a test benchmark for the task, and validated its usefulness with simple baselines from just text or prior. We also proposed a new EQA model consisting of four modular components: a program generator, a navigator, a controller, and VQA module for MT-EQA. We experimentally demonstrated that our model significantly outperforms baselines on both question answering and navigation, and conducted detailed ablative analysis for each component in both the embodied and oracle settings.

Acknowledgements: We thank Abhishek Das, Devi Parikh and Marcus Rohrbach for helpful discussions. This work is supported by NSF Awards #1633295, 1562098, 1405822, and Facebook.

References

[1] Ankesh Anand, Eugene Belilovsky, Kyle Kastner, Hugo Larochelle, and 1Aaron Courville. Blindfold baselines for embodied qa. *arXiv preprint arXiv:1811.05013*, 2018. 4

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

[3] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *ICML*, 2017. 3

[4] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016. 2, 3

[5] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. 2

[6] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. Home: A household multimodal environment. *arXiv preprint arXiv:1711.11017*, 2017. 1, 3

[7] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *AAAI*, 2018. 1, 2

[8] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. *CVPR*, 2018. 1, 2, 3, 5, 6, 7, 8

[9] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural Modular Control for Embodied Question Answering. *CoRL*, 2018. 3

[10] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. *arXiv preprint arXiv:1810.11181*, 2018. 3

[11] Abhinav Gupta Dhiraj Gandhi, Lerrel Pinto. Learning to fly by crashing. *IROS*, 2017. 2

[12] Li Fei-Fei and Juan Carlos Niebles. Temporal modular networks for retrieving complex compositional activities in videos. In *ECCV*, 2018. 3

[13] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *CVPR*, 2018. 2

[14] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 1, 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[16] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *ICCV*, 2017. 3

[17] Ronghang Hu, Marcus Rohrbacnh, Jacob Andreas, Trevor Darrell, and Kate Saenko. Modeling relationship in referential expressions with compositional modular networks. In *CVPR*, 2017. 3

[18] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 2, 5

[19] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. *ICCV*, 2017. 3

[20] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, 2016. 1, 2

[21] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 1, 3

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[23] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016. 2

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2

[25] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 5

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. 2015. 2

[27] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. *RSS*, 2017. 2

[28] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 1, 3

[29] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 3

[30] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016. 2

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[32] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *ICLR workshop*, 2018. 1, 2, 3

[33] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 1, 3

[34] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *CVPR*, 2018. 3

[35] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. Visual semantic planning using deep successor representations. In *ICCV*, 2017. 1, 2

[36] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 2