

Visual Curiosity: Learning to Ask Questions to Learn Visual Recognition

Jianwei Yang^{1*} Jiasen Lu^{1*} Stefan Lee¹ Dhruv Batra^{1,2} Devi Parikh^{1,2}

¹Georgia Institute of Technology ²Facebook AI Research

Abstract: In an open-world setting, it is inevitable that an intelligent agent (*e.g.*, a robot) will encounter visual objects, attributes or relationships it does not recognize. In this work, we develop an agent empowered with *visual curiosity*, *i.e.* the ability to ask questions to an Oracle (*e.g.*, human) about the contents in images (*e.g.*, ‘*What is the object on the left side of the red cube?*’) and build visual recognition model based on the answers received (*e.g.*, ‘*Cylinder*’). In order to do this, the agent must (1) understand what it recognizes and what it does not, (2) formulate a valid, unambiguous and informative ‘language’ query (a question) to ask the Oracle, (3) derive the parameters of visual classifiers from the Oracle response and (4) leverage the updated visual classifiers to ask more clarified questions.

Specifically, we propose a novel framework and formulate the learning of *visual curiosity* as a reinforcement learning problem. In this framework, all components of our agent – visual recognition module (to see), question generation policy (to ask), answer digestion module (to understand) and graph memory module (to memorize) – are learned entirely end-to-end to maximize the reward derived from the scene graph obtained by the agent as a consequence of the dialog with the Oracle.

Importantly, the question generation policy is disentangled from the visual recognition system and specifics of the ‘environment’ (scenes). Consequently, we demonstrate a sort of ‘double’ generalization – our question generation policy generalizes to new environments *and* a new pair of eyes, *i.e.*, new visual system. Specifically, an agent trained on one set of environments (scenes) and with one particular visual recognition system is able to ask intelligent questions about new scenes when paired with a new visual recognition system.

Trained on a synthetic dataset, our results show that our agent learns new visual concepts significantly faster than several heuristic baselines – even when tested on synthetic environments with novel objects, as well as in a realistic environment.

Keywords: Visual Curiosity, Learn to Ask, Visual Recognition, Dialog

1 Introduction

As the various artificial intelligence sub-fields (vision, language, reasoning) mature, we are beginning to see ambitious multi-disciplinary tasks being undertaken – at the intersection of vision-and-language (*e.g.* image captioning [1, 2, 3], visual question answering [4, 5], visual dialog [6]), vision-and-navigation [7, 8], and vision-language-and-navigation [9, 10, 11]. These tasks (and others) implicitly rely on the assumption that agent’s visual recognition system is mature enough (*i.e.* can recognize scenes, objects, their attributes, relationships, *etc.*) to support these higher-level AI tasks.

However, in an open world, it is inevitable that the agent will encounter some new visual content (new scenes, objects, attributes) that it has never seen before. In such cases, it is natural to consider whether the agent can simply ‘ask’ a human or an Oracle to identify the novel content and build visual classifiers on the fly. Note that this is a challenging task since the agent must (1) understand what it recognizes and what it does not, (2) formulate a valid, unambiguous and informative ‘language’ query (a question) to ask the Oracle, (3) derive the parameters of visual classifiers from the Oracle response and (4) leverage the updated visual classifiers to ask more clarified questions.

Towards this goal, we develop an agent with the ability to ask questions about an image to an Oracle and build visual classifiers based on the answers received. We call this ability – *visual curiosity*. Fig. 1 left illustrates this setup. Given an image, the agent’s visual system generates object proposals (or candidate bounding boxes). The agent is confident about labels of some candidate boxes

*Equal contribution

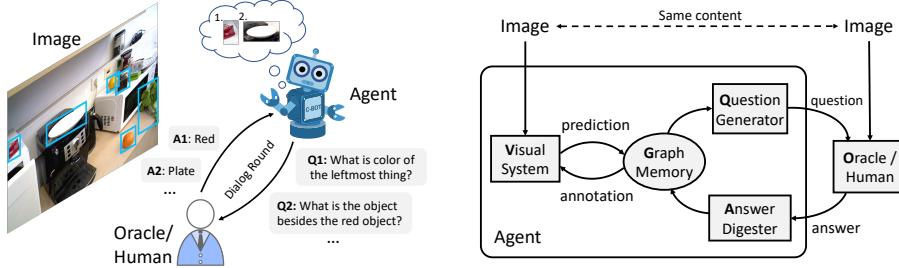


Figure 1: Left: an example scenario where the agent learns to recognize objects through a dialog with an Oracle. Right: the proposed framework contains a visual recognition module (to see), question generation policy (to ask), answer digester (to understand) and graph memory module (to memorize).

(‘orange_fruit’, ‘lettuce’), but does not recognize the content in others. It generates a question ‘*What is the color of the leftmost object?*’. The Oracle responds with the answer ‘red’, which the agent uses to update its ‘red’ classifier. Furthermore, the agent uses the ‘red object’ as a referent in future rounds of dialog to acquire labels of other objects (‘*What is the object besides the red object?*’).

One immediate question at this point may be – what is the relationship of this setup to active learning [12, 13, 14, 15]? A full discussion is available in Section 3, but in short, our approach lies at the intersection of active learning and meta-learning – *i.e.*, instead of using a pre-specified active learning protocol, we *learn to actively learn* [16, 17, 18]. Specifically, we formulate this task as a reinforcement learning problem and *learn a policy* to ask questions to learn visual recognition. All components of our agent (illustrated in Fig. 1 right) – visual recognition module (to see), question generation policy (to ask), answer digester (to understand) and graph memory module (to memorize) – are learned entirely end-to-end to maximize the reward derived from the scene graph generated by the agent as a consequence of the dialog with the Oracle.

Importantly, the question generation policy is disentangled from the visual recognition system and specifics of the environment (scenes). Consequently, we demonstrate a sort of ‘double’ generalization – our question generation policy generalizes to new environments *and* a new pair of eyes. Specifically, an agent trained on one set of environments (scenes) and with one particular visual recognition system is able to ask intelligent questions about new scenes when paired with a new visual recognition system (which may or may not recognize the same set of entities as the visual system during training).

Our results show that our agent – trained in a synthetic environment with a certain set of objects – learns new visual concepts significantly faster than several heuristic baselines when deployed in a synthetic environment with novel objects as well as in a more realistic environment.

In order to make progress on this challenging problem, we make a number of simplifying assumptions that are described in detail in Section 2 but highlighted here for completeness and full disclosure – we use templated questions with slots that are filled by the agent, and model only simple geometric relationships between object proposals (right, left, front, behind) that are trivial for the agent to extract from bounding box coordinates. Also, we assume the agent can localize objects in an image precisely. However, we believe the ideas and components of our work may generalize to more challenging scenarios in the future.

2 Learning to Ask Questions

As illustrated in Fig. 1(b), there are four major components in our framework:

- 1) **Visual System** \mathcal{V} that localizes image regions with high ‘objectness’ (*i.e.* generates object proposals) and predicts their categories and attributes.
- 2) **Question Generator** \mathcal{Q} that identifies an object proposal to inquire about and generates a question based on graph memory to ask the Oracle about its category or attribute;
- 3) **Answer Digester** \mathcal{D} that uses the Oracle’s answer to update the graph memory for training visual system \mathcal{V} to recognize the contents for future images;
- 4) **Graph Memory** that is a semantic graph representation connecting the other three components.

Graph Representation. In our work, the agent’s graph memory is the underlying data-structure connecting all other components; thus, we describe it first. It captures information about the image that the agent has gathered from the Oracle. For an image I , $G = (V, E)$ denotes a directed graph where the nodes V correspond to the object proposals (with $|V| = K$), and edges E correspond to the relationships between proposals. Let \mathcal{A} denote a set of visual attributes (*e.g.*, object category,

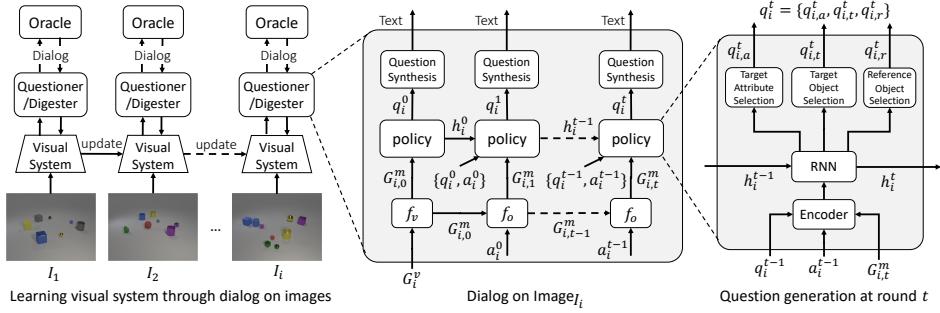


Figure 2: We simulate an agent observes a sequence of images, and interacts with the Oracle through dialogs to update its visual system (left). On each image, the agent asks a number of questions and gets responses from the Oracle (middle). For each question, the agent takes the history and the current graph memory as inputs and fills the question templates recurrently to compose a question (right).

object shape) on object proposals. For an attribute concept $a \in \mathcal{A}$, n_a denotes the number of states for the concept a (e.g. concept *color* can be ‘red’, ‘blue’, ‘green’, etc). Let Δ^n be a n -simplex. Then, $p^a \in \Delta^{n_a}$ denotes a probability distribution over attribute states for attribute concept a . Similarly, n_r denotes the number of spatial relationships and $p^r \in \Delta^{n_r}$ denotes a probability distribution over these relationships. Besides these distributions, each object proposal has a spatial location \mathbf{l} . We can then write the nodes of the graph as $V = \{(\mathbf{l}_k, p_k^{a_1}, \dots, p_k^{a_{|\mathcal{A}|}})\}_{k=1}^K$ and the edges as $E = \{p_{i \rightarrow j}^r | i, j \in [K], i \neq j\}$. In this work, the spatial relationships (left, right, behind, front) between object proposals are trivially recognizable from bounding box coordinates such that $p_{i \rightarrow j}^r$ are always delta functions. As such, we drop them from the graph notation for simplicity – writing $G^m = \{(\mathbf{l}_k, p_k^{a_1}, \dots, p_k^{a_{|\mathcal{A}|}})\}_{k=1}^K$. Besides the agent’s graph memory, the agent also predicts a scene graph from an image using the visual recognition module \mathcal{V} . We denote this graph as $G^v = \{(\mathbf{l}_k, v_k^{a_1}, \dots, v_k^{a_{|\mathcal{A}|}})\}_{k=1}^K$ where $v^a \in \Delta^{n_a}$. Likewise, the oracle \mathcal{O} has an oracle graph $G^o = \{(\mathbf{l}_k, o_k^{a_1}, \dots, o_k^{a_{|\mathcal{A}|}})\}_{k=1}^{K^*}$ corresponding to the ground-truth scene graph with K^* objects. We use G_i^m , G_i^v and G_i^o to denote these graph representations for image I_i .

Environment Setup. To mimic the scenario of an agent traversing a novel environment while being instructed by a human about the world around it, we formalize our learning setup as a Markov Decision Process (MDP) over a series of image grounded dialogs. Specifically, an episode consists of multi-round dialogs about a sequence of n images $I_1, \dots, I_n \in \mathcal{I}$. Our goal is to learn a good policy to discuss with the Oracle in turn one by one on these n images so as to learn a good visual system to recognize objects and attributes. If successful, each of these dialogs with the Oracle produces valuable annotations on which to train the visual system which in turn produces a stronger foundation for subsequent dialogs.

Rollout Process. This environmental setup can be represented by a recurrent process as depicted in Fig. 2 – the agent initializes the graph memory using predictions from its visual system, the agent holds a dialog with the oracle to update this memory, and then the information gained over the dialog is used to update the visual system before this process is repeated for the next image. More formally, assume we have access to a question generation policy π_q , and a visual system \mathcal{V} . Presented with the image I_i , the agent first extracts the visual graph G_i^v from the image with \mathcal{V} . Before beginning the dialog with the Oracle, the agent updates its initial graph memory $G_{i,0}^m$ based on G_i^v through a bottom-up update function $f_v(G_{i,0}^m, G_i^v)$. Then, the agent engages in a T round dialog with the oracle and maintains a sequence of graph memories $\{G_{i,0}^m, \dots, G_{i,T}^m\}$ corresponding to its beliefs about the image I_i at each round. At round t , the agent proposes a question q_i^t using the policy π_q based on the whole dialog history $\mathcal{H}_i^t = \{G_{i,0}^m, q_i^1, a_i^1, G_{i,1}^m, \dots, q_i^{t-1}, a_i^{t-1}, G_{i,t-1}^m\}$. The Oracle receives the question q_i^t and generates an answer a_i^t based on oracle graph G_i^o . Upon receiving the answer, the agent updates its graph memory using the top-down update function $f_o(G_{i,t-1}^m, a_i^t)$.

At the end of dialog on I_i , the agent uses the final graph memory $G_{i,T}^m$ along with the accumulated graph memories $\{G_{1,T}^m, \dots, G_{i-1,T}^m\}$ to update the visual system \mathcal{V} before going to the next image I_{i+1} . This recurrent procedure on n images is outlined in Alg. 1. At the end of this process, the agent produces a trained visual system that can recognize the objects and attributes in images. We will elaborate the detail of each component in following section.

Algorithm 1 $\text{Rollout}(\{I_1, \dots, I_n\}, T)$: MDP rollout process on n images.

Inputs: Image sequence $\{I_1, \dots, I_n\}$; Dialog budget T ; Question generation policy π_q
Outputs: Visual system \mathcal{V} ; Rewards $\{r_1^{1,T}, \dots, r_n^{1,T}\}$

```

1: Initialize  $G_{\{1,\dots,n\}}^o$  with ground truth,  $G_{\{1,\dots,n\}}^m$  with uniform distribution
2: for  $i \in [1 \dots n]$  do
3:    $G_i^v \leftarrow \mathcal{V}(I_i)$                                  $\triangleright$  Extract visual graph from  $I_i$ 
4:    $G_{i,0}^m \leftarrow f_v(G_{i,0}^m, G_i^v)$                  $\triangleright$  Initialize graph memory with visual graph
5:   for  $t \in [1 \dots T]$  do
6:      $q_i^t \leftarrow \pi_q(\mathcal{H}_i^t)$                        $\triangleright$  Generate question
7:      $a_i^t \leftarrow \mathcal{O}(q_i^t, G_i^o)$                    $\triangleright$  Oracle answers the question
8:      $G_{i,t}^m \leftarrow f_o(G_{i,t-1}^m, a_i^t)$              $\triangleright$  Update graph memory with answers
9:   end for
10:  Train  $\mathcal{V}$  with  $[G_{1,T}^m \dots G_{i,T}^m]$            $\triangleright$  Train visual system with graph memories
11: end for

```

2.1 Model

We elaborate on each of the main components of our model in this section.

Question Generator \mathcal{Q} . In order to produce queries to Oracle that are informative, the agent selects from a set of template questions – filling in information from the graph memory. Inspired by [19], each template is associated with a functional program that operates on the oracle scene graph to get the Oracle answer. For example, the question ‘What is the color of the metal object?’ has a corresponding program: ‘*query_color(unique(filter_material(metal, scene))*’.

Using these templates, the question generation is equivalent to selecting the objects and attributes about which to inquire. Specifically, the policy needs to determine which object attribute to ask about (*i.e.*, target attribute), which object to ask about (*i.e.*, target object), and if applicable which object to refer to (*i.e.*, reference object). For instance, for the image in Fig. 1, the generated question may be “What is the $\langle \text{white} \rangle \langle \text{object} \rangle$ besides the $\langle \text{red object} \rangle$ ”. The target object and attribute is $\langle \text{object} \rangle$ and $\langle \text{white} \rangle$ respectively. The reference object is $\langle \text{red object} \rangle$.

We implement the question generation policy π_q using a recurrent neural network (RNN). The memory provided by a recurrent policy is essential for the agent to know which questions have already been asked and whether they were meaningful or not according to the responses from Oracle (*i.e.* referring to valid objects). As illustrated on the right side of Fig. 2, at each recurrent time step through the dialog, this policy takes as input the previous hidden state h_i^{t-1} , the previous round question q_i^{t-1} , its corresponding answer a_i^{t-1} , and the current graph memory $G_{i,t}^m$. Then, it outputs actions to select target attribute, target object, and reference objects. The selection of reference objects is handled by another recurrent process and may specify either none or one reference object.

Oracle \mathcal{O} . Given the question from the agent, the Oracle answers the question by executing the functional program on the oracle graph G_i^o . However, the execution can fail in some cases. First, the question might be ambiguous. For example, the agent may ask ‘What is the color of the sphere?’ when there are multiple spheres in the image. Second, the question might be invalid. For example, it is invalid if the agent asks the same question as above when there are no spheres in the image. As a result, the Oracle has three types of responses to the agent: 1) the answer to the question, 2) ‘ambiguous_question’ and 3) ‘invalid_question’. If the Oracle responds with an answer, *e.g.*, ‘red’ to the agent, the agent’s graph memory will be updated, otherwise it will stay the same.

Updating the Graph Memory. The graph memory is updated from the bottom-up (via visual system \mathcal{V}) and top-down (via answer digester \mathcal{D}) with update functions f_v and f_o , respectively:

- **Bottom-Up f_v :** For object k and attribute a , its probability p_k^a is updated to a one-hot vector by setting its $\arg \max(v_k^a)$ -th entry to 1 and others to 0, if $\max(v_k^a) > \tau_i$, where τ_i is a threshold that is annealed during the recurrent process, $\tau_i = \max(0.9, \exp(-i/n))$.
- **Top-Down f_o :** Suppose the agent asks about attribute a for object k , and the answer is the l -th category for that attribute concept, then the agent will update its graph memory by setting the corresponding l -th entry in p_k^a to 1, and others to 0.

Reward. A good question generator is one that asks meaningful questions to acquire knowledge about images from the Oracle. So we define the reward at each dialog round as:

$$r_i^t = R(G_{i,t-1}^m, G_{i,t}^m, G_i^o) = S(G_{i,t}^m, G_i^o) - S(G_{i,t-1}^m, G_i^o) \quad (1)$$

where $S(\cdot)$ measures the similarity between the graph memory and the oracle graph. The reward is the difference in similarities between the current time step and the previous one. The purpose is

Algorithm 2 Learning to Ask Question to Learn Visual Recognition.

Inputs: Image sequence length n ; Dialog budget T

```
1: Initialize parameters  $\theta_\pi$  and  $\theta_v$  for policy and visual system, respectively
2: while True do
3:   Initialize parameters  $\theta_v$ 
4:    $\mathcal{I} \leftarrow \{I_1, \dots, I_n\} \sim \mathcal{E}$                                  $\triangleright$  Reset visual system at the beginning of episode
5:    $\mathcal{V}, \{r_1^{1,T}, \dots, r_n^{1,T}\} \leftarrow \text{Rollout}(\mathcal{I}, T)$            $\triangleright$  Sample  $n$  images from an environment
6:   Update  $\theta_\pi$  based on  $\{r_1^{1,T}, \dots, r_n^{1,T}\}$  using Eq. (3)            $\triangleright$  Rollout on  $n$  images with Algorithm 1
7: end while
```

to learn an agent that asks meaningful questions at *each* time step so that it can recover as much information as possible within a budget of T questions.

2.2 Learning

The overall learning algorithm for the agent is summarized in Alg. 2. The policy π_q is updated at the end of each episode while the visual system is updated multiple times during the inner *Rollout*. Recall that our goal is to learn a strong question generation policy that can ask useful questions across varied environments and differently skilled visual systems. To achieve this, we decouple the question policy from the visual system during training through two strategies: first, we introduce the semantic graph representation as an intermediate between perception and question generation; secondly, we reset the visual system to a random initialization at the beginning of each episode.

The visual system is trained inside the rollout process. At the end of dialogs on each image I_i , we append the graph memory to the history and use both for training the visual system. This training is a supervised learning task and the objective is:

$$\theta_v^* = \arg \min_{\theta_v} \sum_{I_i \in \mathcal{I}} \sum_{k=1}^{K_i} \sum_{a \in \mathcal{A}} -\mathbf{p}_k^a \cdot \log(\mathbf{v}_k^a) \quad (2)$$

where K_i is the number of object proposals in I_i ; $[\cdot]$ denotes the inner product operator between two vectors. The above objective is targeted to minimize the cross entropy between the prediction of visual system \mathbf{v}_k^a and the graph memory \mathbf{p}_k^a , which is a one-hot vector as mentioned before. We use standard gradient descent methods to optimize this objective.

We train the question generation policy π_q to recover as much information as possible from the Oracle in a limited budget, say T dialog rounds. To succeed, the agent must ask valid, unambiguous questions about uncertain object attributes. To train the policy π_q parametrized by θ_π , we consider maximizing the expected reward gained by the policy over episodes under environment \mathcal{E} ,

$$\theta_\pi^* = \arg \max_{\theta_\pi} J(\theta_\pi) = \arg \max_{\theta_\pi} \mathbb{E}_{\mathcal{I} \sim \mathcal{E}} \mathbb{E}_{\pi_q} \left[\sum_{i=1}^n \sum_{t=1}^T r_i^t (q_i^t \sim \pi_q(\mathcal{H}_i^t)) \right]. \quad (3)$$

In practice, we take a Monte Carlo estimate of this expected reward – sampling a sequence of images and questions throughout our dialogs – and use advantage actor-critic [20] (A2C) to train our agent. Details of the visual system, question templates, question generator can be found in Appendix.

3 Related Work

Active Learning addresses the problem of selecting samples from an unlabeled set to be labeled by some oracle [12, 13, 14, 15]. Common selection criteria rely on heuristics, including entropy [21], expected model change [22], and boosting classifier margin [23]. Unlike traditional active learning, querying the oracle in our setting is not guaranteed to succeed; to gain a new label, agents must correctly refer to target objects when issuing queries to the Oracle. Further, our approach learns to collect labels efficiently from end-to-end training rather than with predefined measures.

Meta Active Learning. Other recent work has also followed this *learning-to-active-learn* strategy [16, 17, 18], training meta-learning models to select sets of instances to be labeled in order to maximize performance of some target model trained on the selected set. As before, these models have direct access to the oracle labels. Further, these meta-learners are tightly coupled with their corresponding target model; being trained based on target model performance. In contrast, our approach is agnostic to the specific perception model.

Learning by Asking Questions. Mirsa *et al.* [24] present a learning-by-asking (LBA) framework for visual question answering (VQA). The main differences between our setting and LBA are two-fold:

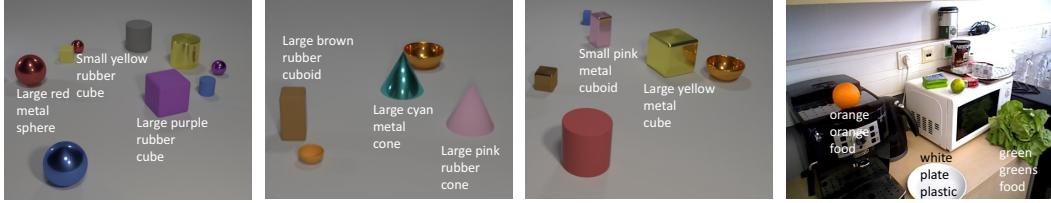


Figure 3: We use two types of datasets in our experiments. One is synthesized (left three columns) and one is a realistic dataset (right most). The synthesized one is further split to three sets, *standard*, *novel* and *mixed*.

1) We focus on learning a better visual system, not a better VQA model. Essentially, LBA is active learning (via language) for VQA, while our work is active learning (via language) to learn to see. 2) Our model decouples the visual system and question generation, which makes the learned question generator agnostic to different environments.

Teaching Robots via Language Interactions. Previous work in human-robot interaction focus on agents learning new concepts from speaking with human operators [25, 26, 27, 28, 29, 30]. Tellex *et al.* [30] present a generalized grounding graph framework based on the linguistic coreference. However, there question generation policy and vision system are not designed to learn. Lütkebohle *et al.* [28] propose use language-interaction to solve ambiguity in the object references and for grasping commands. Thomason *et al.* [25] learn an active learning dialog policy for natural language grounding. Both [26] and [27] generate questions to continuous learn objects and visual properties. However, our goal is to learn a question generation policy that is distangled from the visual recognition system and specifics of the scenes, which enables both active learning and meta learning.

4 Experiments

Recall that our goal is to learn visual curiosity, *i.e.*, a question generation policy that can intelligently ask questions to an Oracle and in doing so acquire meaningful information to train a visual recognition system. A successful agent should work well not only in the setting it was trained, but also in new environments that contain partially or entirely novel attributes and with different visual systems ranging in levels of competency. Moreover, as the visual system is decoupled from question generation, the agent should generalize well to new visual domains, e.g. from synthetic environments to realistic images. We evaluate our method for these qualities in the following experiments.

4.1 Dataset

We evaluate our question generation policy in both synthesized and realistic environments. Exemplar images are shown in Fig. 3. We generate the synthetic datasets using the same API as [19]. Each image contains 5 to 10 objects each with four different attribute types (*shape*, *color*, *material*, and *size*). We construct three different datasets to test generalization; specifically, we generate:

- *Standard* composed of objects from 3 shapes (cube, sphere, cylinder), 6 colors (gray, red, blue, green, yellow, purple), 2 materials (rubber, metal), and 2 sizes (large, small).
- *Novel* consisting of objects from 3 novel shapes (cuboid, bowl, cone) and 4 new colors (pink, brown, cyan, orange) not present in *Standard*; however, materials and sizes are the same. The goal is to check generalization to novel attribute values.
- *Mixed* which contains objects from all 6 shapes, 10 colors, 2 materials and 2 sizes from both *standard* and *novel* splits. This is used to test the generalization ability on complex scenes where some attributes are known and others are not.

We synthesize 1800 images which we split 900/300/600 for train, val, and test respectively. The *standard* train and val sets are used to train the agent policy, and the *standard*, *novel*, and *mixed* test sets are used for evaluation. For the realistic dataset, we use the images and bounding boxes from the Autonomous Robot Indoor Dataset (ARID) [31]. It contains 153 objects from 51 categories. We further annotated each object with one of 6 different materials and one of 11 colors. The agent trained on the synthetic *standard* split is also evaluated on this dataset.

4.2 Metrics and Baselines

For evaluation, we split each test set into 12 folds, each containing 50 images (*i.e.* a single episode sequence). We run the learned agent on each fold and evaluate two metrics:

- **Graph Recovery.** We measure the correctness of the agent’s graph memory. This measures how informative the agent’s questions were. We compute the graph memory’s recall with respect to the

Table 1: Graph recovery performance (i.e., quality of questions asked) on the *Standard*, *Novel*, and *Mixed* test sets for agents trained on *Standard*.

Model	Standard				Novel				Mixed			
	R@10	R@20	R@50	AUC	R@10	R@20	R@50	AUC	R@10	R@20	R@50	AUC
Random	28.3	36.5	59.4	0.41	23.4	31.2	54.0	0.36	27.0	37.3	63.2	0.43
Entropy	29.5	39.1	65.5	0.44	28.3	36.3	61.9	0.42	29.9	40.7	70.7	0.47
Entropy+Context	38.0	52.5	67.1	0.52	35.2	46.5	59.5	0.46	38.5	49.9	66.4	0.52
Our model	42.1	59.1	89.3	0.63	43.3	58.4	88.9	0.64	42.9	60.1	90.3	0.64
Our model w/o \mathcal{V}	25.8	50.6	84.1	0.55	25.5	50.0	85.2	0.55	26.8	51.7	87.2	0.57

ground truth as the percentage of correctly predicted attributes. We report the average graph recall across testing folds at dialog round K as R@K. We also report the area under this curve as AUC.

- **Visual Recognition.** To evaluate if a better question generator leads to a better visual system, we measure how well a visual system performs after being trained through the agent’s interactions with the Oracle on the test fold. To do so, we report the average graph recall of the visual system predictions on the remaining folds.

We compare our proposed approach with three baselines:

- **Random.** This agent randomly samples question to ask i.e it selects the target attribute, target object, and reference objects uniformly at random.
- **Entropy.** An object/attribute with higher entropy (in the graph memory) is more likely to be chosen as the target. Likewise, objects with lower entropy are more likely to be references.
- **Entropy+Context.** The agent prefers to select uncertain (high entropy) object/attribute with reliable (low entropy) neighbors as reference objects. This way, the model prefers to ground questions on objects with low ambiguity.

For comparisons, we make no changes other than replacing our approach with the above baselines.

5 Results

Recall that we train on the *standard* train set and evaluate on the *standard*, *novel* and *mixed* test sets.

Questioner Graph Memory. We first compare graph memory recovery for different models. As seen in Table. 1, our approach consistently outperforms the baseline models by a significant margin across all three test settings. Further, the performance between *standard* and *novel/mixed* is similar, suggesting that our approach generalizes well to novel settings. The **Random** and **Entropy** baselines both struggle to propose unambiguous questions without the use of spatial context. The **Entropy+Context** model fairs better, but falls off later when the hand-crafted strategy fails to find unambiguous reference objects. Our model steadily improves over the entire dialog and has apparently found a much better question asking strategy that generalizes well across different environments.

Static Vision Ablation. We also evaluate an ablated version of our model (**Ours w/o \mathcal{V}**) which never updates its visual system \mathcal{V} . This model must ask questions essentially from ‘scratch’ without any bottom-up visual information. As shown in Table 1, the agent starts dialogs with significantly lower graph similarity scores than our full model; however, as the dialog proceeds, this agent performs similarly. This highlights that the agent has learned to ask informative questions and not to simply rely on steadily improving the visual system.

Visual System Performance. We report the visual recognition accuracies in Fig. 4(a-c). We take visual system checkpoints throughout the agent dialogs and evaluate them on the held out folds – tracking the evolution of the visual system through the agent’s interactions with the Oracle. We find our approach outperforms the baselines significantly in all settings. This is somewhat unsurprising as question generation and visual system learning are naturally synergistic – with improvement of either leading to easier improvement in the other.

5.1 Transferring to Realistic Environment

Here, we apply the policy learned on the synthesized standard dataset to the realistic dataset. The episode length is also set to 50. In both situations, we observe significantly higher graph recalls for our model (86.2 R@50) than the baselines (56.7, 66.1, 55.5, for random, entropy, and entropy+context respectively). More details are in Appendix. We also show the visual recognition curves in Fig. 4(d). As we can see, the learned question generator can flexibly adapt to the realistic dataset and the learned visual system outperforms other baselines by a large margin. These results imply that our model

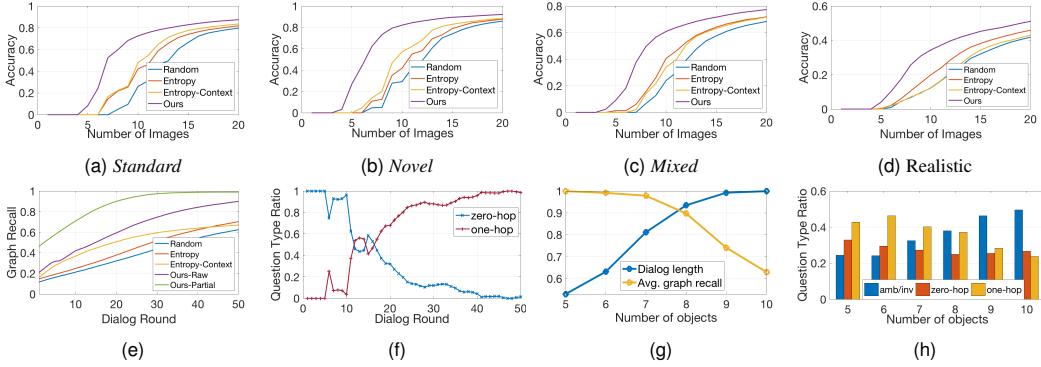


Figure 4: Top Row: visual recognition accuracy curves against dialog round on different test sets. Bottom Row: Inspecting different aspects of question generation.

could perhaps be deployed on a real embodied agent and learn a visual system from traversing in an environment with a guide.

5.2 Inspecting the Question Generator

Questioner starting with partially learned visual system. We investigate how the question generator behaves on the *Mixture* set if its initial visual system can already recognize some of the attributes, *i.e.*, those in the *standard* set. In this case, the agent needs to ask about the remaining unknown attributes. In Fig. 4e, we show average graph recalls on the *Mixture* test set. We find the average graph recall for this ablation (Ours-Partial) starts from a much higher graph recall, and continues to increase to almost 1 – indicating that the learned policy can also generalize well to partially trained visual systems. This is a promising result showing that the agent can leverage known visual concepts when learning about new ones and integrate additional visual systems seamlessly.

Question type against dialog round. We run our question generator on all *Standard* test images individually without updating the visual system. As shown in Fig. 4f, it proposes more zero-hop questions at the beginning and then transitions to one-hop questions. This demonstrates that our model has learned an efficient strategy that asks questions about directly referable objects (e.g. leftmost) first and then objects that require referring to other (known) objects.

Questioner behavior with varying number of objects. We explore how the number of objects in images affect the question generation behavior. We separately evaluate the learned policy on images with varying number of objects. As shown in Fig. 4g, the average graph recall on images decreases and the relative dialog length (divided by maximal length 50, so can be plotted from 0 to 1) increases when there are more objects in the images. In Fig. 4h, we can see there are fewer unambiguous/valid questions when the number of objects increases – implying that greater numbers of objects increases the difficulty for the questioner. However, our model can still perform well. As shown in Fig. 4(g), our approach still achieves an average graph recall of 89.75 with 8 objects present.

6 Conclusion

In this paper, we introduce a new setting *learning visual curiosity*, where an agent *learns* to ask questions to learn visual recognition. This is a challenging task where the agent needs to understand what it recognizes in an image and formulate language queries to the Oracle that are both unambiguous and informative. We use a graph memory to decouple the visual system and question generator. As a result, we demonstrate “double” generalization – we show that the learnt policy to ask informative questions generalizes to new environments as well as to a new visual system. We experimentally demonstrate that a policy learnt on a synthetic set of objects generalizes to novel objects, to mixture of novel objects and attributes, as well as to a realistic dataset – significantly better than strong baselines. This ability to learn about new objects and attributes by interacting with an Oracle is key to agents that operate in realistic open world settings.

Acknowledgements This work was supported in part by NSF, AFRL, DARPA, Siemens, Google, Amazon, ONR YIPs and ONR Grants N00014-16-1-{2713,2793}. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government, or any sponsor.

References

- [1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [2] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.
- [3] J. Lu, J. Yang, D. Batra, and D. Parikh. Neural baby talk. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [4] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.
- [5] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7w: Grounded question answering in images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [6] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra. Visual Dialog. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [7] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [8] 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 *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [9] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. v. d. Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [10] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [11] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [12] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2007.
- [13] X. Li and Y. Guo. Adaptive active learning for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [14] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.
- [15] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal of Computer Vision (IJCV)*, 108(1-2):97–114, 2014.
- [16] P. Bachman, A. Sordoni, and A. Trischler. Learning algorithms for active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
- [17] G. Contardo, L. Denoyer, and T. Artieres. A meta-learning approach to one-step active learning. *arXiv preprint arXiv:1706.08334*, 2017.
- [18] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.
- [19] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [20] 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 *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

- [21] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [22] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.
- [23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [24] I. Misra, R. Girshick, R. Fergus, M. Hebert, A. Gupta, and L. van der Maaten. Learning by Asking Questions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [25] J. Thomason, A. Padmakumar, J. Sinapov, J. Hart, P. Stone, and R. J. Mooney. Opportunistic active learning for grounding natural language descriptions. In *Proceedings of the Annual Conference on Robot Learning (CoRL)*, 2017.
- [26] Y. Yu, A. Eshghi, and O. Lemon. Learning how to learn: an adaptive dialogue agent for incrementally learning visually grounded word meanings. In *RoboNLP@ACL*, 2017.
- [27] D. Skočaj, A. Vrečko, M. Mahnič, M. Janíček, G.-J. M. Kruijff, M. Hanheide, N. Hawes, J. L. Wyatt, T. Keller, K. Zhou, et al. An integrated system for interactive continuous learning of categorical knowledge. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(5):823–848, 2016.
- [28] I. Lütkebohle, J. Peltason, L. Schillingmann, C. Elbrechter, B. Wrede, S. Wachsmuth, and R. Haschke. The curious robot-structuring interactive robot learning. In *International Conference on Robotics and Automation*, 2009.
- [29] Y. Sun, L. Bo, and D. Fox. Learning to identify new objects. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [30] S. Tellexll, P. Thakerll, R. Deitsl, D. Simeonovl, T. Kollar, and N. Royl. Toward information theoretic human-robot dialog. *Robotics*, page 409, 2013.
- [31] M. R. Loghmani, B. Caputo, and M. Vincze. Recognizing objects in-the-wild: Where do we stand? In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [32] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [33] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [35] J. Yang, J. Lu, D. Batra, and D. Parikh. A faster pytorch implementation of faster r-cnn. https://github.com/jwyang/faster_rcnn.pytorch, 2017.
- [36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

7 Appendix

7.1 Question Generator

As mentioned in Sec. 2, our question generation policy π_q is implemented using a recurrent neural network (RNN). The memory provided by a recurrent policy is essential for the agent to know which questions have already been asked and whether they were meaningful or not according to the responses from Oracle. As illustrated in Fig. 2 right side, it consists of two components, target selection policy and reference selection policy. The first one determines which object and attribute to ask about. The second one determines whether to use a reference and which one to use if needed. These two policies share the low-level representations. Hence, we first elaborate the representation we use.

Representation. At the t -th dialog round on I_i , the question generator takes the question q_i^{t-1} and answer a_i^{t-1} from last round, and graph memory $G^m = \{(l_k, p_k^{a_1}, \dots, p_k^{a_{|\mathcal{A}|}})\}_{k=1}^K$ as the input. Based on these three inputs, we compute:

- **Entropy of Graph Memory:** For object k and its attribute a , we compute $e_k^a = Entropy(p_k^a)$. For the whole scene graph memory, we obtain the entropy tensor $K \times |\mathcal{A}| \times 1$;
- **Location Embedding.** For each object, we normalize its bounding box location l_k with image size and then use a two-layer MLP (4 – 4 – 2) to embed it to two dimensions. For all K objects, the dimension is $K \times 2$. Afterward, we duplicate it for all attribute concepts, and thus obtain a tensor $K \times |\mathcal{A}| \times 2$;
- **Target at last round:** For each of K objects, we use one-hot tensor to encode which target object and which attribute the agent pointed to at last dialog round. Hence, the dimension is $K \times |\mathcal{A}| \times 1$.
- **Reference at last round:** We use another one-hot vector to encode which reference object the agent pointed, and the dimension is $K \times 1$. We use another one-hot vector $K \times 1$ to record whether the agent use a reference object or not. If the agent does not use reference object, then the vector becomes zero vector. Similarly, We combine them and duplicate it for all attribute concepts to $K \times |\mathcal{A}| \times 2$;
- **Answer at last round:** We use one-hot vector to encode the answer from Oracle at last round. If the answer is valid, then we assign 1 to the target and reference object slots; otherwise 0. As a result, we obtain $K \times |\mathcal{A}| \times 1$ and $K \times 1$ for target and reference, respectively. Afterward, we duplicate $K \times 1$ reference vector to $K \times |\mathcal{A}| \times 1$ and concatenate it with target tensor to obtain $K \times |\mathcal{A}| \times 2$.

Combining all the above signals, the final input to our question generator policy network at t -th dialog round is $x_t \in \mathcal{R}^{K \times |\mathcal{A}| \times 8}$. In our dataset, the number of attribute concept is 4. We replace $|\mathcal{A}|$ with 4 in the following for clarity. Given $x_t \in \mathcal{R}^{K \times 4 \times 8}$, we first reshape it to $K \times 32$, where each row encode the graph memory and history for one object. Then we vectorize x_t to K vectors and feed them as a batch to a LSTM, obtaining new features $x_t^p \in \mathcal{R}^{K \times 64}$ by

$$h_t, c_t = lstm(h_{t-1}, c_{t-1}, x_t); x_t^p = h_t \quad (4)$$

where h_{t-1} and c_{t-1} are the hidden state and cell memory from the lstm network at dialog round $(t-1)$. This x_t^p from the hidden state in lstm will be used in both target and reference policy.

Target policy. It is aimed at pointing the right target object and attribute concept to ask about. This can be completed by directly pointing one of $K \times 4$ slots. To propose meaningful target and reference objects, the context is important. In our work, we exploit graph convolutional layers [32] to pass the context information across different objects. Specifically, the GCN layer has the following typical formulation:

$$\mathbf{z}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W \mathbf{z}_j^{(l)} \right) \quad (5)$$

where $\mathcal{N}(i)$ is the neighbors of node i ; W is a learnable projection matrix; α_{ij} is the affinity between node i and j . In our model, we compute the affinity between two object nodes based on the spatial distance:

$$\alpha_{ij} = \alpha_{ji} = \exp \left(-\frac{d(i, j)}{d_{max}} \right) \quad (6)$$

where d_{max} is the maximal distance in all object pairs. Given this affinity matrix, we first reshape x_t^p to $(K \times 4) \times 16$ and pass the above tensor through two graph convolutional layers to obtain $x_t^{tar} \in \mathcal{R}^{(K \times 4) \times 16}$. Then we pass x_t^{tar} through two-layer MLP (16-16-1) to obtain $(K \times 4)$ scores, and further a softmax layer to obtain a probability distribution $\mathbf{p}_t^{tar} = Softmax(mlp(x_t^{tar}))$. Besides the head for action, we have another head to compute the value. We simply perform average pooling for x_t^{tar} and also pass it to two-layer MLP to obtain the value for each of the object nodes. At end, to select the target object and attribute, we use an epsilon greedy sampling ($\epsilon = 0.1$) strategy to choose one entry during training and choose the maximal one during testing.

Reference policy. It is aimed at determining whether to use reference object and which one if needed. It also takes x_t^p as input. To select the right reference, this policy needs to know which target object is selected. Suspect the k -th object is selected as the target, we take the corresponding k -th 1×64 vector, and replicate it, which is

then concatenated with the remaining to obtain $x_t^{ref} \in \mathcal{R}^{(K-1) \times 128}$. To determine which object to select as the reference, we also use two graph convolutional layers to update x_t^{ref} to 64 dimension. Then, output is sent to a two-layer MLP (64-32-1) to obtain the $K - 1$ dimensional scores over all candidate reference objects. Similar to the way used in target policy, the reference object is selected based epsilon greedy during training and the entry with maximal score during testing. Meanwhile, x_t^{ref} is fed into another two graph convolutional layers to 64-d, which are then average pooled to obtain a single 64-d vector. This vector is then sent to a two-layer MLP (64-32-1) to predict whether or not to use reference object. For both selecting reference and determining whether to use reference, we compute the value using x_t^{ref} as input.

Based on the above policies, we can deterministically compose a question with the corresponding template and ask it to the Oracle. We will introduce the details on the question template we used in our experiments below.

7.2 Question Templates

In our question templates, we introduce four attribute concepts size ($\langle Z \rangle$), color ($\langle C \rangle$, material ($\langle M \rangle$), shape ($\langle S \rangle$). We use $\langle R \rangle$ to depict the relation between two objects, which could be ‘left’, ‘right’, ‘front’ and ‘behind’. Besides, we introduce the absolute spatial relationship $\langle P \rangle$ to depict the spatial location of one object proposal to the whole image. According to its location, it can be ‘left-most’, ‘right-most’, ‘closest’ ‘farthest’ or ‘None’ otherwise. Further, we use $\langle L \rangle$ to indicate whether the target object proposal is at the extreme location among all proposals that have the relationship $\langle R \rangle$ to its reference. It can be ‘closest’ if it is extreme, and ‘None’ otherwise. For clarification, we show two exemplar question templates below:

- “What shape is the $\langle P \rangle \langle Z \rangle \langle C \rangle \langle M \rangle \langle S \rangle$? ”
- “What size is $\langle L \rangle \langle Z \rangle \langle C \rangle \langle M \rangle \langle S \rangle$ that is $\langle R \rangle \langle Z \rangle \langle C \rangle \langle M \rangle \langle S \rangle$? ”

\mathcal{Q} first points to the target object and reference object (if needed) and fill them into the above templates correspondingly. Based on the locations of target and reference objects, the relationship $\langle R \rangle$, absolute location $\langle P \rangle$ and relative location $\langle L \rangle$ are manually inferred. Thus far, we can compose a unique questions which is then forward to Oracle side. Fig. 7 and Fig. 8 show the zero-hop and one-hop text and program templates on 4 attributes respectively.

7.3 Implementation Details

We elaborate on the implementation details below:

Visual system. We use Faster-RCNN [33] in conjunction with a pre-trained VGG16 [34] as the backbone of the visual system. We use the implemetnation open sourced in [35]. During training, the backbone is fixed, and we only learn the parameters for the four attribute classifiers (shape, color, size, material), which are two layer MLPs. We use the ground-truth bounding boxes on the agent side, since proposing object regions from the images is not our focus. In the future, we will try to use a region proposal network (RPN) to get the object proposals on the agent side. We are effectively assuming the the agent understands what constitutes an object, just not their names or attributes.

Question templates. We use zero-hop and one-hop question templates in our model. This is for two reasons: 1) they are enough to compose informative questions; 2) lower hop questions are more plausible to humans. Two simplified exemplar templates are: 1) Zero-hop: “What shape is the $\langle Some\ Object \rangle$? ”; 2) “What size is $\langle One\ Object \rangle$ that is $\langle Spatial\ Relation \rangle \langle Another\ Object \rangle$? ”

During training, the length of the sampled image sequences in an episode is set to 100 and we train the question generator over 200 episodes. The visual system is updated with 50 gradient descent steps during each update. We start the visual system update once it accumulates $5 \times n_a$ annotations for attribute concept a . We use the Adam optimizer [36] for the whole model. The learning rate starts from 1e-4 and decreases by a factor 0.99 after each image.

7.4 Attribute Annotations for ARID

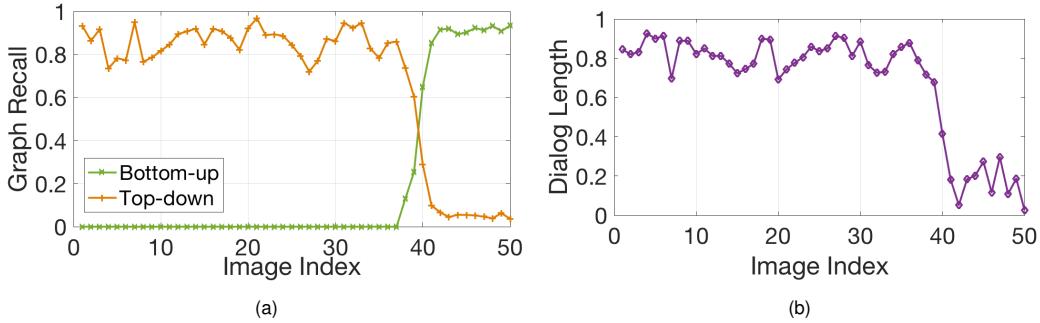
The attributes annotations for ARID [31] is shown in table 2. We annotate three attribute concepts: object, color and material. Though the huge number of object categories than our synthetic dataset, our model trained on synthetic dataset generalizes very well to this new environment. Fig. 9 and Fig. 10 show the example of an image and the associated scene graph on *mixture* synthesized dataset and ARID dataset.

7.5 Graph recall from Bottom-up and Top-down

Recall that the graph memory is updated both by the visual system as well as information from the oracle. We investigate the contributions of these two factors to the graph recovery over the dialogs. As shown in Fig. 5(a), as the dialog proceeds, the agent relies more on its visual system and less on interactions with Oracle. Since the graph memory is either updated bottom-up or top-down, we can easily measure their contributions by counting the number of entries updated by visual system and oracle in the graph memory. Also, as shown in Fig. 5(b), the number of dialog rounds drops. This is a plausible behavior since we would not expect an intelligent agent to keep asking questions repeatedly after multiple interactions with Oracle.

Table 2: Attributes annotations for ARID dataset, contains object, color and material.

Object	lightbulb, apple, bell, calculator, sponge, keyboard, marker, scissors, glue, lime, flashlight, cell, lemon, instant, peach, toothpaste, bowl, rubber, camera, orange, banana, plate, coffee, ball, mushroom, food, pear, pitcher, dry, kleenex, toothbrush binder, notebook, garlic, cereal, pliers, comb, tomato, water, stapler, onion, greens, potato, cap, shampoo, hand, soda
Color	blue, brown, purple, grey, yellow, mixed, pink, green, orange, black, white, silver, red
Materials	cloth, food, metal, plastic, glass, paper

**Figure 5:** Left: graph recall contributed bottom-up and top-down; Right: relative dialog length over time.**Table 3:** Graph recall on realistic test set for policy trained on synthetic *Standard* train set.

Model	Realistic			
	R@10	R@20	R@50	AUC
Random	20.1	27.5	56.7	0.33
Entropy	20.3	32.0	66.1	0.39
Entropy+Context	30.2	39.9	55.5	0.41
Ours model	35.6	53.4	86.2	0.59

7.6 Graph Recall on Realistic Environment

As a supplement to the Section 5.1 in our main paper, in Table 3, we present the graph recalls for different methods on our collected realistic dataset. Clearly, our model outperforms all three baselines significantly. Though not being trained on the realistic environment, our questioner successfully propose meaningful questions to ask and get much higher graph recalls. Moreover, the numbers are comparable to those reported on synthetic test sets. **These numbers indicate that our model have a strong generalization ability across different environments.**

7.7 Qualitative Results

In Fig. 6, we show some qualitative results on both synthetic dataset and realistic dataset. Specifically, the questioner generation policy is trained on *Standard* train set, and then applied to test sets. Here, we display the first 16 rounds of dialog with oracle on three images, which are from *Mixed*, and ARID dataset. Our model learns to begin with zero-hop questions (blue), and followed with one-hop questions (green). Moreover, the learned question generation policy tends to repeatedly query one object until all the attributes are observed. When transferring to realistic environment, our method can successfully generalize to new objects and attributes, and ask meaningful questions. This verifies the effectiveness of our framework on disentangling the question generation policy and visual recognition system. We also find our model sometimes asks the ambiguous questions (red) which can not be answered by Oracle. The ambiguous questions can be either zero-hop question or one-hop question. When looking more closely, we find the ambiguous question is mainly caused by the unspecified target object, *e.g.*, “What size is the thing left of the small cyan shiny cylinder?”, “There is a thing that ...”. However, by taking the current graph memory and histories, the agent can successfully get rid of this soon after a few dialog rounds with the Oracle.

7.8 Limitations and Future Directions

As we mentioned early, our work is an initial step towards learning visual curiosity. To start on this challenging task, we've made a number of simplifying assumptions that future work could soften. For instance, extending the model to more complicated visual scenarios where object proposal systems might be error-prone. In this case, the visual contents from the perspective of agent and Oracle are different from each other, which make the questions more ambiguous or confusing to the Oracle. One way to address this is empower the Oracle the visual curiosity ability as well, including answering clarifying questions to the agent. Another extension is considering richer sets of relationships between objects, and enable the agent to learn about relationships as well during the interaction with Oracle. Further, models could be extended to operate on non-templated dialog exchange, *i.e.*, natural questions from agent and natural answers from Oracle. At last, in the current setting, we assume the number of the attribute concepts is given. However, incorporating with incremental learning to grow the attribute space over time would also be an interesting future direction to explore.

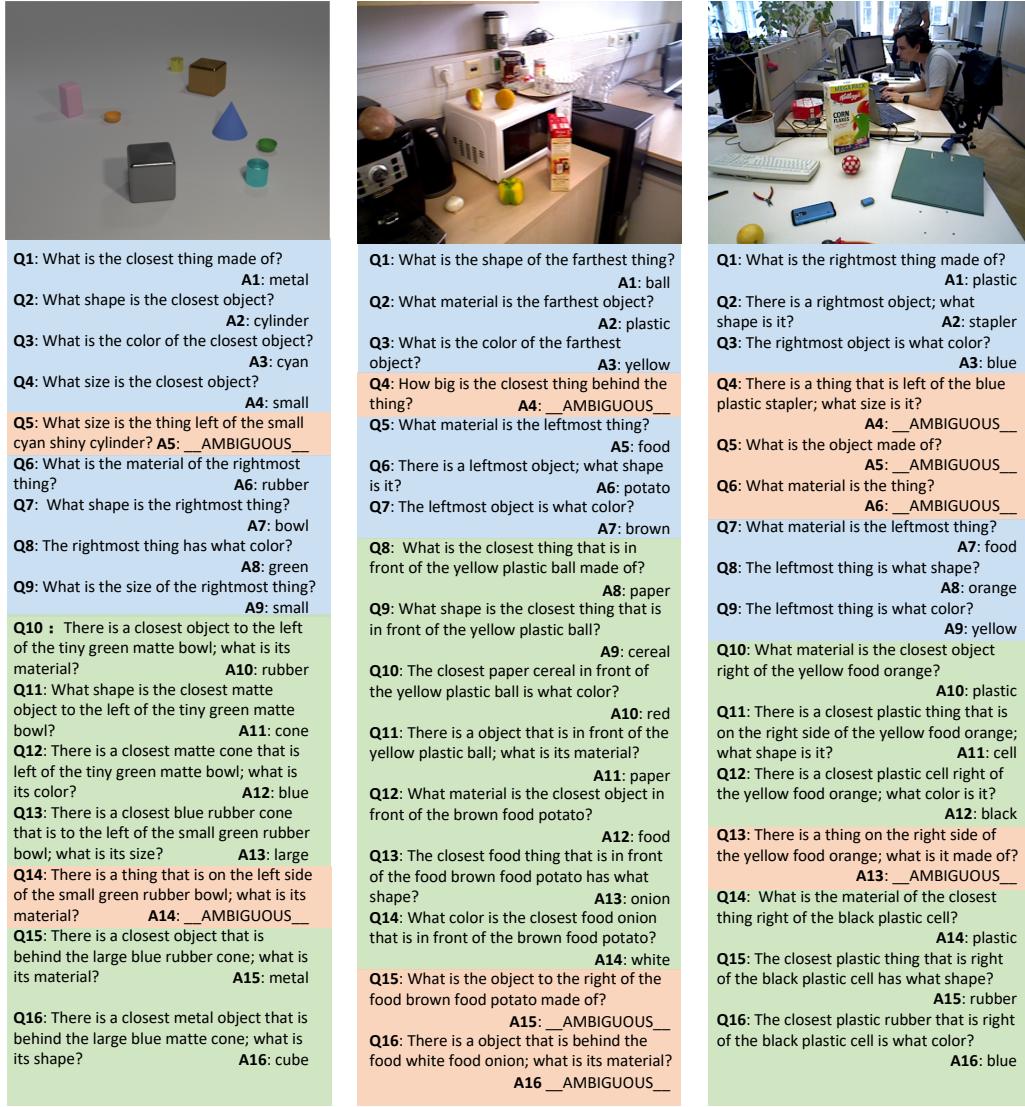


Figure 6: Dialogs with Oracle on *Mixed* synthesized dataset (left) and ARID dataset (middle and right) based on the policy learned on *normal* synthesized dataset. Questions in blue, green and red background corresponds to one-hop, two-hop and ambiguous questions respectively.

<pre> "type": ["size"], "text": ["What size is the <P> <Z> <C> <M> <S>?", "What is the size of the <P> <Z> <C> <M> <S>?", "The <P> <Z> <C> <M> <S> has what size?", "The <P> <Z> <C> <M> <S> is what size?", "How big is the <P> <Z> <C> <M> <S>?"], "nodes": [{ "inputs": [], "type": "scene" }, { "side_inputs": [<Z>, <C>, <M>, <S>], "inputs": [0], "type": "filter" }, { "side_inputs": [<P>], "inputs": [1], "type": "filter_pos" }, { "inputs": [2], "type": "query_size" }], "params": [{ "type": "Position", "name": "<P>" }, { "type": "Size", "name": "<Z>" }, { "type": "Color", "name": "<C>" }, { "type": "Material", "name": "<M>" }, { "type": "Shape", "name": "<S>" }], </pre>	<pre> "type": ["color"], "text": ["What color is the <P> <Z> <C> <M> <S>?", "What is the color of the <P> <Z> <C> <M> <S>?", "The <P> <Z> <C> <M> <S> has what color?", "The <P> <Z> <C> <M> <S> is what color?"], "nodes": [{ "inputs": [], "type": "scene" }, { "side_inputs": [<Z>, <C>, <M>, <S>], "inputs": [0], "type": "filter" }, { "side_inputs": [<P>], "inputs": [1], "type": "filter_pos" }, { "inputs": [2], "type": "query_color" }], "params": [{ "type": "Position", "name": "<P>" }, { "type": "Size", "name": "<Z>" }, { "type": "Color", "name": "<C>" }, { "type": "Material", "name": "<M>" }, { "type": "Shape", "name": "<S>" }], </pre>
<pre> "type": ["material"], "text": ["What material is the <P> <Z> <C> <M> <S>?", "What is the material of the <P> <Z> <C> <M> <S>?", "What is the <P> <Z> <C> <M> <S> made of?"], "nodes": [{ "inputs": [], "type": "scene" }, { "side_inputs": [<Z>, <C>, <M>, <S>], "inputs": [0], "type": "filter" }, { "side_inputs": [<P>], "inputs": [1], "type": "filter_pos" }, { "inputs": [2], "type": "query_material" }], "params": [{ "type": "Position", "name": "<P>" }, { "type": "Size", "name": "<Z>" }, { "type": "Color", "name": "<C>" }, { "type": "Material", "name": "<M>" }, { "type": "Shape", "name": "<S>" }], </pre>	<pre> "type": ["shape"], "text": ["What shape is the <P> <Z> <C> <M> <S>?", "What is the shape of the <P> <Z> <C> <M> <S>?", "The <P> <Z> <C> <M> <S> has what shape?", "What is the shape of the <P> <Z> <C> <M> <S>?", "There is a <P> <Z> <C> <M> <S>; what shape is it?", "The <P> <Z> <C> <M> <S> is what shape?"], "nodes": [{ "inputs": [], "type": "scene" }, { "side_inputs": [<Z>, <C>, <M>, <S>], "inputs": [0], "type": "filter" }, { "side_inputs": [<P>], "inputs": [1], "type": "filter_pos" }, { "inputs": [2], "type": "query_shape" }], "params": [{ "type": "Position", "name": "<P>" }, { "type": "Size", "name": "<Z>" }, { "type": "Color", "name": "<C>" }, { "type": "Material", "name": "<M>" }, { "type": "Shape", "name": "<S>" }], </pre>

Figure 7: Zero-hop text and program templates on 4 attributes concepts (size, color, material, shape)

Figure 8: One-hop text and program templates on 4 attributes concepts (size, color, material, shape)

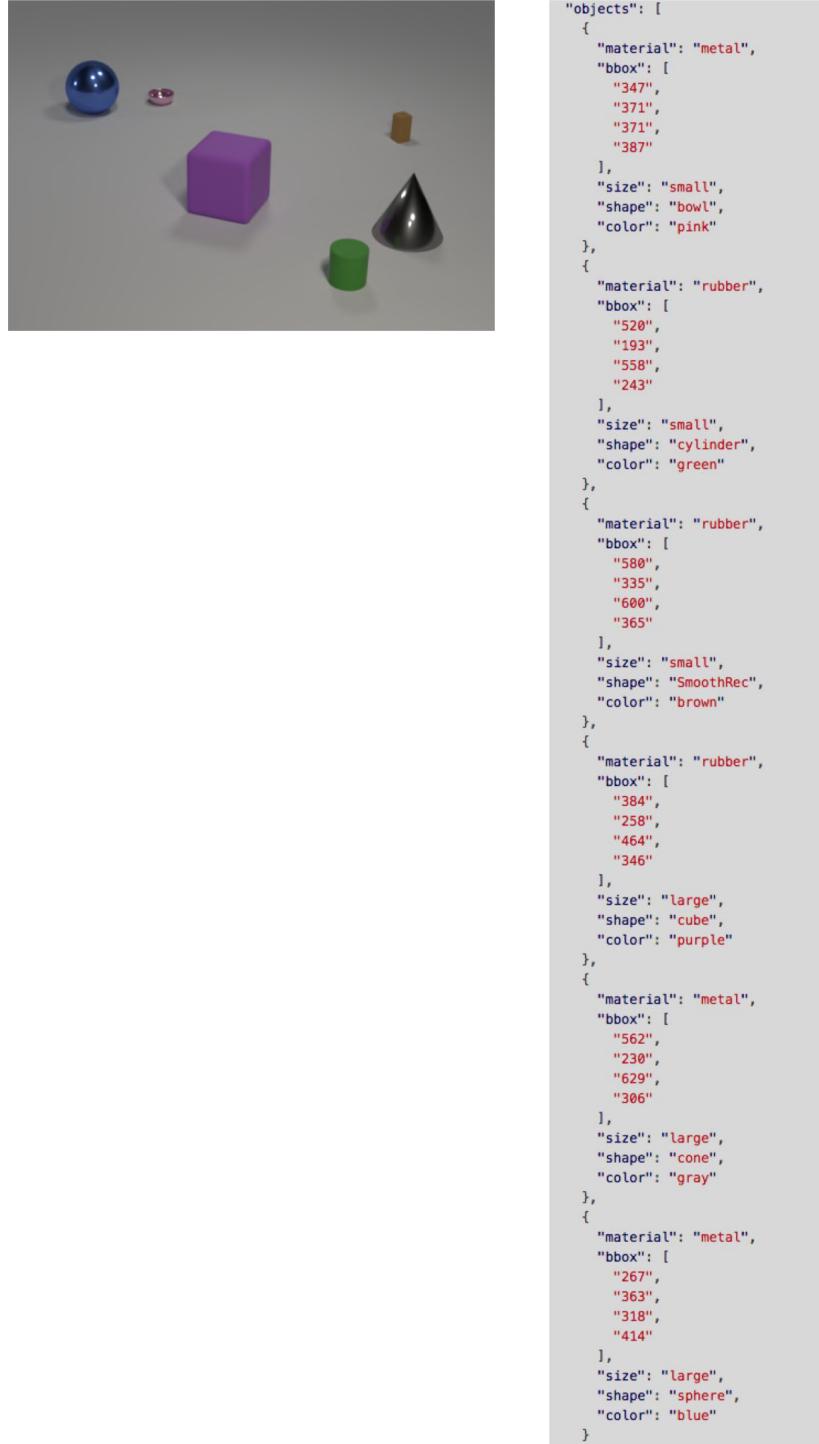


Figure 9: Example of an image and the associated scene graph on *mixture* synthesized dataset.

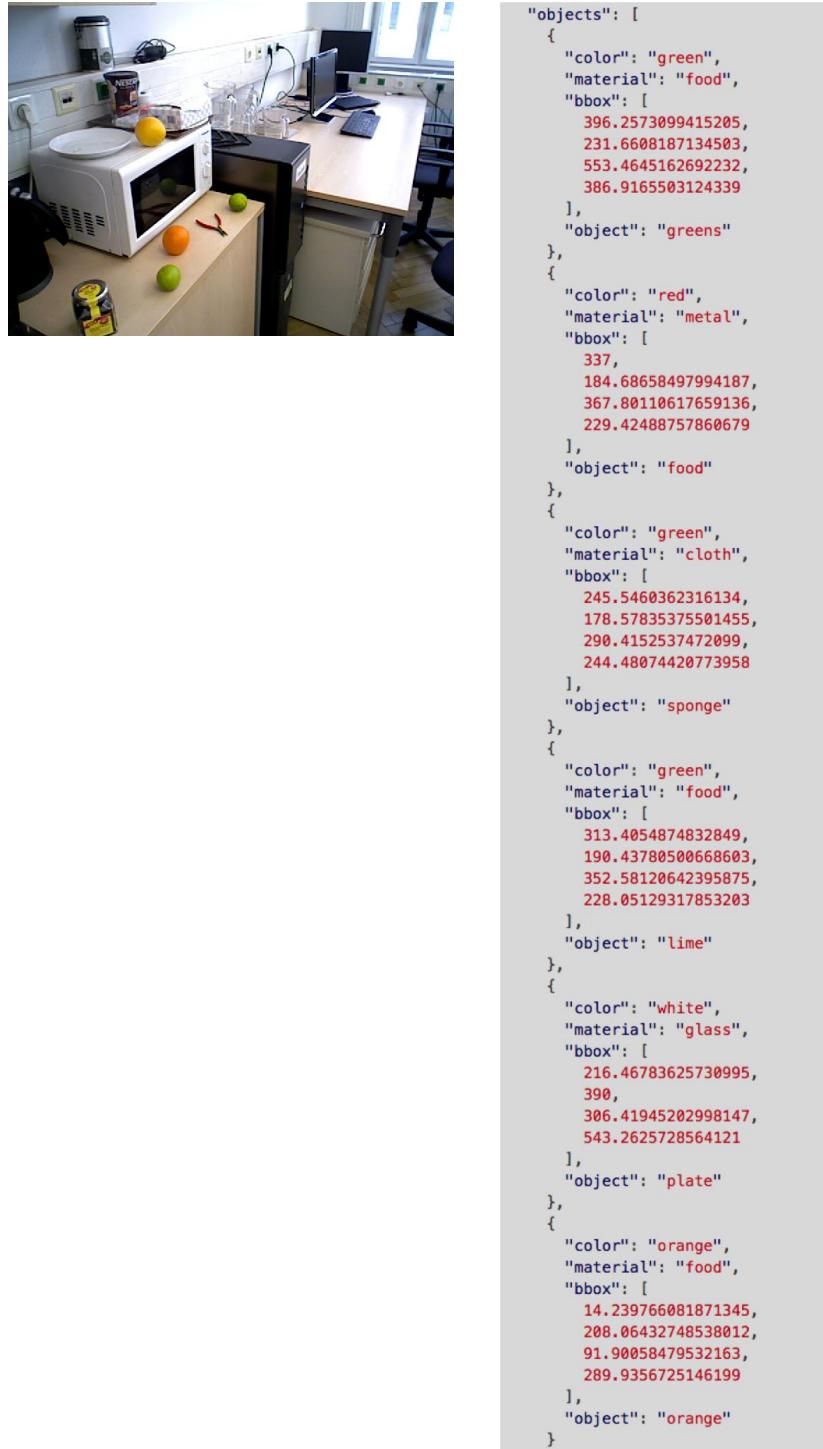


Figure 10: Example of an image and the associated scene graph on ARID dataset.