Supplementary Material to Efficient Attention Mechanism for Visual Dialog that can Handle All the Interactions between Multiple Inputs

Van-Quang Nguyen[1], Masanori Suganuma[2,1], and Takayuki Okatani[1,2]

[1] Graduate School of Information Sciences, Tohoku University
[2] RIKEN Center for AIP
{quang, suganuma, okatani}@vision.is.tohoku.ac.jp

A Representations of Utilities

A.1 Image Utility

The image utility is represented by the standard method employed in many recent studies. It is based on the bottom-up mechanism [1], which extracts region-level image features using the Faster-RCNN pre-trained on the Visual Genome dataset [10]. For each input image, we select the top K objects, and represent each of them by a visual feature $v_i^r \in \mathbb{R}^{2048}$ and a bounding box expressed by $(x_{i,1}, x_{i,2})$ and $(x_{i,3}, x_{i,4})$ (the coordinates of the upper-left and lower-right corners.)

The feature vector v_i^r is then converted into another vector $v_i^f \in \mathbb{R}^d$ as follows. We introduce the following notation to express a single FC layer with ReLU, to which dropout regularization is applied:

$$\underset{k\to d}{\mathrm{MLP}}(x) \equiv \mathrm{Dropout}(\mathrm{ReLU}(W^\top x + b)), \tag{1}$$

where $x \in \mathbb{R}^k$ is the input and $W \in \mathbb{R}^{k\times d}$ and $b \in \mathbb{R}^d$ are the weights and biases. Then, v_i^f is obtained by

$$v_i^f = \mathrm{LayerNorm}(\underset{2048\to d}{\mathrm{MLP}}(v_i^r)), \tag{2}$$

where LayerNorm is the layer normalization [2] applied to the output.

The bounding box geometry is converted into $v_i^b \in \mathbb{R}^d$ in the following way. First, the image is resized to 600×600 pixels and the bounding box geometry is transformed accordingly. Then, representing each of the four coordinates by a one-hot vector of size 600, we convert them into the embedding vectors $\hat{x}_{i,1}, \ldots, \hat{x}_{i,4} (\in \mathbb{R}^d)$ using four different embedding layers. Then, we obtain v_i^b as below

$$v_i^b = \sum_{j=1}^{4} \mathrm{LayerNorm}(\underset{d\to d}{\mathrm{MLP}}(\hat{x}_{i,j})). \tag{3}$$

Finally, v_i^f encoding the visual feature and v_i^b encoding the spatial feature are aggregated by adding and normalizing as

$$v_i = \text{LayerNorm}(v_i^f + v_i^b). \tag{4}$$

The resulting v_i's for the K objects ($i = 1, \ldots, K$) comprise a matrix $V = [v_1, v_2, \cdots, v_K]^\top \in \mathbb{R}^{K \times d}$, which gives the representation of the visual utility.

Optional Image Feature Enrichment. In the experiment of comparing ensembles on the test split of Visdial v1.0, we enrich the image features for further improvement. To be specific, for each object, we also obtain a class label with highest probability (e.g. *'cat', 'hair', and 'car'*) and the top 20 attributes for each class label (e.g., 'curly', 'blond', 'long', and so on, for the label 'hair'). These can be extracted from the Faster-RCNN along with the above CNN features and bounding box geometry. We incorporate these into the image utility representation in the following way.

The class label for the i-th object is first encoded into an embedding vector $e_i^c \in \mathbb{R}^{300}$ using the same embedding layer as the question. Then, we convert e_i^c into a d-dimensional vector v_i^c by

$$v_i^c = \text{LayerNorm}(\underset{300 \to d}{\text{MLP}}(e_i^c)). \tag{5}$$

Similarly, for the top 20 attributes of each object i, we encode them into embedding vectors of size 300, i.e. $e_{i,1}^a, \ldots, e_{i,20}^a$, and then convert them further into $v_i^a \in \mathbb{R}^d$ as

$$v_i^a = \sum_{j=1}^{20} \text{LayerNorm}(\underset{300 \to d}{\text{MLP}}(e_{i,j}^a) w_{i,j}^a, \tag{6}$$

where $w_{i,j}^a$ is the confidence score extracted from the Faster-RCNN for attribute j of the i-th object. Then, the visual feature v_i^f, the spatial feature v_i^b, the class feature v_i^c, and the attribute feature v_i^a are aggregated by their addition followed by normalization as

$$v_i = \text{LayerNorm}(v_i^f + v_i^b + v_i^c + v_i^a). \tag{7}$$

We then use these vectors to form the matrix V instead of Eq.(4).

A.2 Question Utility

The question utility is also obtained by the standard method but with one exception, the employment of positional embedding used in NLP studies. Note that we examine its effects in an ablation test shown in the main paper. A given question sentence is first fit into a sequence of N words; zero-padding is applied if necessary. Each word w_i ($i = 1, \ldots, N$) is embedded into a vector e_i of a fixed size using an embedding layer initialized with pretrained GloVe vectors [15].

They are then inputted into two-layer Bi-LSTM, obtaining two d-dimensional vectors $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ as their higher-layer hidden state:

$$\begin{aligned}\overrightarrow{h_i} &= \mathrm{LSTM}(e_i, \overrightarrow{h_{i-1}}),\\ \overleftarrow{h_i} &= \mathrm{LSTM}(e_i, \overleftarrow{h_{i+1}}).\end{aligned} \tag{8}$$

Their concatenation, $h_i = [\overrightarrow{h_i}^\top, \overleftarrow{h_i}^\top]^\top$, is then projected back to a d-dimensional space using a linear transformation, yielding a vector q_i^f. Positional embedding q_i^p from the paper [18] is added to get the final representation $q_i \in \mathbb{R}^d$ of w_i as

$$q_i = \mathrm{LayerNorm}(q_i^f + q_i^p). \tag{9}$$

The representation of the question utility is given as $Q = [q_1, \ldots, q_N]^\top \in \mathbb{R}^{N\times d}$.

A.3 Dialog History Utility

In this study, we choose to represent the dialog history as a single utility. Each of its entities represents the question-answer pair at one round. As with previous studies, the caption is treated as the first round of $2N$-word which is padded or truncated if necessary. For each round $t > 1$, the word sequences of the question and the answer at the round is concatenated into $2N$-word sequence with zero padding if necessary. As with the question utility, after embedding each word into a GloVe vector, the resulting sequence of $2N$ embedded vectors is inputted to two-layer Bi-LSTM, from which only their last (higher-layer) hidden states are extracted to construct $2d$-dimensional vector $[\overrightarrow{h_0}^\top, \overleftarrow{h_{2N}}^\top]^\top$. We then project it with a linear transform to a d-dimensional space, yielding $r_t^f \in \mathbb{R}^d$. For the linear projection, we use different learnable weights from the question utility. As in Eq.(9), we add positional embedding, which represents the order of rounds, and then apply layer normalization, yielding a feature vector of the round t question-answer pair. The history utility is then given by $R = [r_1, \ldots, r_T]^\top \in \mathbb{R}^{T\times d}$.

B Design of Decoders

B.1 Discriminative Decoder

A discriminative decoder outputs the likelihood score for each of 100 candidate answers for the current question at round T in the following way. We use a similar architecture to the one used to extract question features in Sec. A.2 to convert each candidate answer (indexed by $i(= 1, \ldots, 100)$) to a feature vector $a_i \in \mathbb{R}^d$. Specifically, it is two-layer Bi-LSTM receiving a candidate answer at its input, on top of which there is a linear projection layer followed by layer normalization. Using the resulting vectors, the score p_i for i-th candidate answer is computed by

$$p_i = \mathrm{logsoftmax}_i(a_1^\top c, \ldots, a_{100}^\top c). \tag{10}$$

In the test phase, we sort the candidate answers using these scores. In the training phase, the cross-entropy loss $\mathcal{L}_D$ between $p = [p_1, \ldots, p_{100}]^\top$ and the ground truth label encoded by a one-hot vector y is minimized:

$$\mathcal{L}_D = -\sum_{i=1}^{100} y_i p_i. \tag{11}$$

When relevance scores $s = [s_1, \ldots, s_{100}]^\top$ over the answer candidates are available (called dense annotation in the VisDial dataset) rather than a single ground truth answer, we can use them by setting $y_i = s_i$ for all i's and minimize the above loss. We employ dropout with rate of 0.1 for the LSTM.

B.2 Generative Decoder

Following [3], we also consider a generative decoder to score the candidate answers using the log-likelihood scores. The generative decoder consists of a two-layer LSTM to generate an answer using the context vector c as the initial hidden state. In the training phase, we predict the next token based on the current token from the ground truth answer. In details, we first append the special token "SOS" at the beginning of the ground truth answer, then embedding all the sentence into the embedding vectors $a_{gt} = [w_0, w_1, \ldots, w_N]$ where w_0 is the embedding vector of "SOS" token. The hidden state $h_n \in \mathbb{R}^d$ at the n-th timestep (extracted from the higher-layer LSTM) is computed given w_{n-1} and h_{n-1} as follows:

$$h_n = \text{LSTM}(w_{n-1}, h_{n-1}), \tag{12}$$

where h_0 is initialized by c. Thus, we compute p_n, the log-likelihood of n-th word as

$$p_n = \text{logsoftmax}_j(W_n^\top h_n + b_n), \tag{13}$$

where $W_n \in \mathbb{R}^{d \times |V|}$ and $p_n \in \mathbb{R}^{|V|}$, where $|V|$ is the vocabulary size; and j is the index of n-th word in the vocabulary.

In the training phase, we minimize $\mathcal{L}_G$, the summation of the negative log-likelihood defined by

$$\mathcal{L}_G = -\sum_{n=1}^{N} p_n. \tag{14}$$

In the validation and test phase, for each candidate answer $A_{T,i}$, we compute $s_i = \sum_{n=1}^{N} p_n^{(A_{T,i})}$ where $p_n^{(A_{T,i})}$ is the log-likelihood of the n-th word in the candidate answer $A_{T,i}$ which is computed similarly as in Eq.(13). Then, the rankings of the candidate answers are derived as $\text{softmax}_i(s_1, \ldots, s_{100})$. We employ dropout with rate of 0.1 for the LSTM.

C Implementation Details

When computing $\bar{\mathcal{A}}_Y(X)$, we perform the following form of computation

$$\mathcal{A}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V,$$

where we compute a matrix product $QK^\top$ as above. In the computation of $\bar{\mathcal{A}}_X(Y)$, we need another matrix product, but it is merely the transposed matrix $KQ^\top$ due to the symmetry between X and Y. For the computational efficiency, we perform computation of $\bar{\mathcal{A}}_Y(X)$ and $\bar{\mathcal{A}}_X(Y)$ simultaneously; see `MultiHeadAttention`(X,Y) in our code. Further, following [12], we also pad X and Y with two d-dimensional vectors that are randomly initialized with He normal initialization. This implements "no-where-to-attend" features in the computation of $\bar{\mathcal{A}}_Y(X)$ and $\bar{\mathcal{A}}_X(Y)$.

Table 1: Hyperparamters used in the training procedure.

Hyperparameter	Value
Warm-up learning rate	1e−5
Warm-up factor	0.2
Initial learning rate after the 1st epoch	1e−3
β_1 in Adam	0.9
β_2 in Adam	0.997
ϵ in Adam	1e−9
Weight decay	1e−5
Number of workers	8
Batch size	32

Table 1 shows the hyperparameters used in our experiments, which are selected following the previous studies. We perform all the experiments on a GPU server that has four Tesla V100-SXM2 of 16GB memory with CUDA version 10.0 and Driver version 410.104. It has Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz of 80 cores with the RAM of 376GB memory. We use Pytorch version 1.2 [14] as the deep learning framework.

D Analysis on Visdial v1.0 Validation Split

D.1 Analyzing Inconsistency between NDCG and Other Metrics

As mentioned in the main paper, we observed the inconsistency in the performance of models evaluated by NDCG and other metrics, such as MRR. The same is also reported in recent studies. We show an analysis on this.

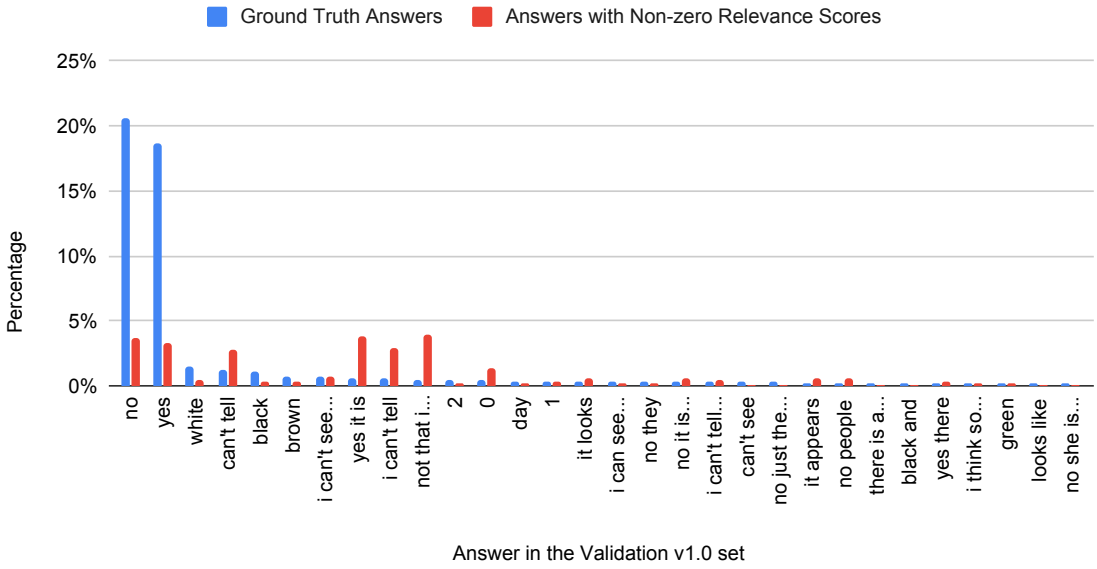

Fig. 1: The distribution of the 30 most popular answers in the Validation v1.0 set.

We first recap how the Visdial v1.0 dataset was collected [3]. A live chat between two workers, i.e., a questioner and an answerer, was conducted on Amazon Mechanical Turk (AMT). For an image provided with a caption, the questioner raised a question based on the caption without seeing the image. The answerer responded to the question by looking at the image, which are used as ground truth answers.

To cope with the difficulty of evaluating answers generated by a model in the form of free texts, Das et al. [3] proposed a method that discriminatively evaluates the performance of visual dialog systems by using a set of 100 candidate answers, to each of which a relevance score is given. It makes a system under evaluation return the rankings of all the candidate answers and then calculates the scores of metrics, e.g. NDCG, MRR, etc. based on the returned rankings. To create a set of 100 candidate answers for each question, they collected from all the answers given by the answerers, the **plausible** answers of the 50 most similar questions to the ground truth answer including itself, the 30 most **popular** answers, and 20 **random** answers. Each of these candidate answers was then given a relevance score with a consensus of several AMT workers.

Now we make a few observations on the dataset. First, the ground truth answers provided by the answerers are not always high-quality. As shown in Table 2, 33.6% of the ground truth answers have relevance scores lower than 0.5. Assuming the AMT workers giving the relevance scores to be accurate, this

Table 2: The performance of the training strategies, i.e. based on the MRR or NDCG early stopping, categorized by questions of corresponding relevance score of ground truth answers.

Rel Score	Percentage	MRR-favored		NDCG-favored	
		MRR	NDCG	MRR	NDCG
0.0	9.0%	**58.26**	44.06	56.12	**48.00**
0.2	11.0%	**58.26**	44.06	57.70	**54.46**
0.4	13.6%	**61.07**	56.94	60.69	**58.94**
0.6	16.0%	**65.38**	59.40	62.35	**62.37**
0.8	19.2%	**67.34**	62.90	64.45	**66.79**
1.0	31.2%	**67.48**	65.13	65.37	**69.21**

implies some of the "ground truth" answers provided by the answerers are simply wrong. Second, the answerers tend to more frequently use short, general answers, such as '*no*' and '*yes*'. This is illustrated in Fig. 1 that shows the frequencies of the most popular ones in the ground truth answers and also those having non-zero relevance scores. It is clearly seen that the short and less informative answers (i.e., '*no*' and '*yes*') are less frequently considered to be relevant.

Recall that NDCG metric is measured based on the rankings of all the candidate answers, whereas MRR and other metrics are based on the ranking of the ground truth answers. Based on the above observations, we can say that the NDCG is more appropriate as an evaluation metric, following the other recent studies. This claim is also supported by Table 2, the results of the experiments examining how evaluated performances vary depending on when to stop the training of the proposed model. It is seen from the table that the model at epoch 5, which is noted as 'MRR-favored' as it corresponds to early stopping based on validation on MRR, yields high MRR and low NDCG scores over all questions. It tends to give higher scores on the safe and popular answers that appear more frequently in the ground truth answers. When we continue to train the model until 12 epochs, it (noted as NDCG-favored) generates better rankings for all possible answers rather than only the ground truth answers, yielding large improvements in NDCG scores. However, it yields lower MRR scores, since the model does not give high ranks to some of the "ground truth" answers; they are indeed very likely to be bad answers. It is also seen from the table that the both models yield better scores on the both MRR and NDCG metrics for the questions having the ground truth answers with high relevance scores.

D.2 Question-Type Analysis

Following [4], we perform a question-type analysis of the NDCG scores achieved by different decoders from the model mentioned in our main paper. The ques-

Table 3: The performance comparison of discriminative and generative decoders evaluated on question types evaluated on the NDCG metric.

Question Type		Yes/No	Number	Color	Others
Percentage		75%	3%	11%	11%
Decoder	Model				
Generative	ReDAN [4]	63.49	41.09	52.16	51.45
	Ours	**66.24**	**46.35**	**55.77**	**57.25**
Discriminative	ReDAN [4]	60.89	44.47	58.13	52.68
	Ours	**64.08**	**49.86**	**60.95**	**58.16**

Table 4: Retrieval performance of compared methods and ours on the val v0.9 split reported with a single model.

Model	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓
SAN [20]	57.64	43.44	74.26	83.72	5.88
LF [3]	58.07	43.82	74.68	84.07	5.78
HRE [3]	58.46	44.67	74.5	84.22	5.72
HREA [3]	58.68	44.82	74.81	84.36	5.66
MN [3]	59.65	45.55	76.22	85.37	5.46
NMN [6]	61.60	48.28	77.54	86.75	4.98
HCIAE [11]	62.22	48.48	78.75	87.59	4.81
AMEM [17]	62.27	48.53	78.66	87.43	4.86
SF [7]	62.42	48.55	78.75	87.75	4.47
GNN [21]	62.85	48.95	79.65	88.36	4.57
CoAtt [19]	63.98	50.29	80.71	88.81	4.47
CoefNMN [9]	64.10	50.92	80.18	88.81	4.45
FGA [16]	65.25	51.43	82.08	89.56	4.35
RvA [13]	66.34	52.71	82.97	90.73	3.93
DAN [8]	66.38	53.33	82.42	90.38	4.04
Ours	**67.94**	**55.05**	**83.98**	**91.58**	**3.69**

tions are classified into four categories: Yes/No, Number, Color, and Others. As shown in Table 3, the Yes/No questions account for the majority whereas there is only 3% of the Number questions. Therefore, the performance on the Yes/No questions translates into the overall performance of any models. Similar to ReDAN [4], the performance of ours on the Number questions is the lowest among the other question types, reflecting the hardness of the counting task. Another similarity observed on our models and ReDAN is that generative decoders show better performance on the Yes/No questions while discriminative decoders yield higher NDCG scores on the other questions. It is because generative decoders favor the short answers that are relevant more often in the Yes/No

questions. It is also seen that our model consistently shows better performance over all question types, i.e. about 3pp on the Yes/No and Color questions, 5pp on the Number questions, and 6pp on the other questions.

E Results on the Visdial v0.9 dataset

Following the previous studies, we report the performance of our method (specifically, the discriminative decoder) on the VisDial v0.9 dataset. The v0.9 dataset consists of the train v0.9 split (82,783 images) and the val v0.9 split (40,504 images). Note that all the hyperparameter settings are the same as those on the Visdial v1.0 dataset except that we train the model with only five epochs.

Table 4 shows the results on the validation set along with performances of other methods. It shows that our model consistently outperforms all the methods across all metrics: MRR, R@1, R@5, R@10 and Mean.

F Qualitative Results

We provide additional examples of the results obtained by our method in Figs. 2-11. They are divided into two groups, results for which the top-1 prediction coincides with the ground truth answer (Figs. 2-6) and those for which they do not coincide (Figs. 7-11). For each result, we show the attention maps created on the input image and question, respectively.

G Experiments on AVSD

To test the generality of the proposed method on other tasks as well as its performance on a greater number of utilities, we additionally apply it to the Audio Visual Scene-aware Dialog (AVSD) task [5]. This task requires a system to generate an answer to a question about events seen in a video given with a previous dialog. AVSD provides more utilities than Visual Dialog, i.e., audio features and video features, such as VGG or I3D features (I3D RGB sequence and I3D flow sequence). We build a network by simply replacing the multimodal attention mechanism in the baseline model of [5] with a simple extension of the proposed attention mechanism. Details are given below.

G.1 Network Design

Following the baselines [5], we extract the question utility Q using a two-layer LSTM. We separate the caption from the dialog history and feed it into another two-layer LSTM to obtain the caption utility C. Similar to [5], the dialog history consisting of previous question-answer pairs is inputted into a hierarchical LSTM network; specifically, we encode each question-answer pair with one LSTM and summarize the obtained encodings with another LSTM, yielding a final vector

Table 5: Comparison of response generation evaluation results with objective measures.

Model	Video Feat.	CIDEr	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROUGE_L
Baseline [5]	VGG	0.618	0.231	0.141	0.095	0.067	0.102	0.259
Ours	VGG	**0.841**	**0.266**	**0.172**	**0.118**	**0.086**	**0.117**	**0.296**
Baseline [5]	I3D	0.727	0.256	0.161	0.109	0.078	0.113	0.277
Ours	I3D	**0.851**	**0.277**	**0.178**	**0.122**	**0.088**	**0.119**	**0.302**

representation c_r. All LSTMs used for language encoding have d units. We convert words into vectors with a shared embedding layer initialized with GLoVe vectors.

The video provides two sources of features, i.e., video features and audio features. We use the audio features extracted from the pretrained VGGish model [5], which are fed to a projection layer, providing the audio utility A; it is represented as a collection of d-dimensional vectors. For video processing, following [5], we consider two models with different features: i) VGG features extracted from four uniformly sampled frames in the video, giving the video utility V, and ii) I3D features extracted by the I3D network pretrained on an action recognition task, which are forwarded to projection layers to obtain an I3D-rgb utility and an I3D-flow utility denoted by V and F.

To compute the multimodal attention between U utilities, we add a stack of U proposed attention blocks; $U = 4$ for the model (i) and $U = 5$ for (ii). To make the designs of two models (i) and (ii) similar, we use only A utility to attend language utilities; and only Q and C are allowed to attend audio and video utilities. After obtaining the updated representations of all utilities, we summarize each utility into a single vector by the self-attention mechanism, in which the summarized vector of question utility is denoted by c_q. We concatenate all these vectors together with c_r, projecting it into a d-dimensional vector of context representation c.

The decoder architecture is similar to the generative decoder described in Sec. B.2 except that the input of the decoder at the i-th step is the concatenation of w_{i-1}, c_q, and c_r. At the time of inference, we use the beam search technique to efficiently find the most likely hypothesis generated by the decoder.

G.2 Experimental Setup

Following [5], we perform the experiment on the AVSD prototype which is split into training, validation, and test sets with 6172, 732, and 733 videos, respectively. Each video is collected from the Charades dataset, annotated with a caption and 10 dialog rounds. The hidden size d is set to 512; the GLoVe vectors are 300-dimensional. We train the models in 15 epochs using the Adam optimizer with initial learning rate 1×10^{-3} in all the experiments. The dropout with rate of 0.2 is applied for the LSTMs.

G.3 Experimental Results

Table 5 shows the results, which include evaluation on a number of metrics to measure the quality of generated answers, i.e. CIDEr, BLEU, METEOR, ROUGE_L. It is seen that our models outperform the baselines presented in [5] over all the metrics; specifically, it improves the CIDEr score by 22.3% (from 0.618 to 0.841) with VGG features and by 12.4% (from 0.727 to 0.851) with I3D features.

References

1. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6077–6086 (2018)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
3. Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J.M., Parikh, D., Batra, D.: Visual dialog. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 326–335 (2017)
4. Gan, Z., Cheng, Y., Kholy, A.E., Li, L., Liu, J., Gao, J.: Multi-step reasoning via recurrent dual attention for visual dialog. In: Proceedings of the Conference of the Association for Computational Linguistics. pp. 6463–6474 (2019)
5. Hori, C., Alamri, H., Wang, J., Wichern, G., Hori, T., Cherian, A., Marks, T.K., Cartillier, V., Lopes, R.G., Das, A., et al.: End-to-end audio visual scene-aware dialog using multimodal attention-based video features. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2352–2356 (2019)
6. Hu, R., Andreas, J., Rohrbach, M., Darrell, T., Saenko, K.: Learning to reason: End-to-end module networks for visual question answering. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 804–813 (2017)
7. Jain, U., Lazebnik, S., Schwing, A.G.: Two can play this game: visual dialog with discriminative question generation and answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5754–5763 (2018)
8. Kang, G.C., Lim, J., Zhang, B.T.: Dual attention networks for visual reference resolution in visual dialog. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 2024–2033 (2019)
9. Kottur, S., Moura, J.M., Parikh, D., Batra, D., Rohrbach, M.: Visual coreference resolution in visual dialog using neural module networks. In: Proceedings of the European Conference on Computer Vision. pp. 153–169 (2018)
10. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision **123**(1), 32–73 (2017)
11. Lu, J., Kannan, A., Yang, J., Parikh, D., Batra, D.: Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In: Advances in Neural Information Processing Systems. pp. 314–324 (2017)

12. Nguyen, D.K., Okatani, T.: Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6087–6096 (2018)
13. Niu, Y., Zhang, H., Zhang, M., Zhang, J., Lu, Z., Wen, J.R.: Recursive visual attention in visual dialog. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6679–6688 (2019)
14. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
15. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543 (2014)
16. Schwartz, I., Yu, S., Hazan, T., Schwing, A.G.: Factor graph attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2039–2048 (2019)
17. Seo, P.H., Lehrmann, A., Han, B., Sigal, L.: Visual reference resolution using attention memory for visual dialog. In: Advances in Neural Information Processing Systems. pp. 3719–3729 (2017)
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)
19. Wu, Q., Wang, P., Shen, C., Reid, I., van den Hengel, A.: Are you talking to me? reasoned visual dialog generation through adversarial learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6106–6115 (2018)
20. Yang, Z., He, X., Gao, J., Deng, L., Smola, A.: Stacked attention networks for image question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 21–29 (2016)
21. Zheng, Z., Wang, W., Qi, S., Zhu, S.C.: Reasoning visual dialogs with structural and partial observations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6669–6678 (2019)

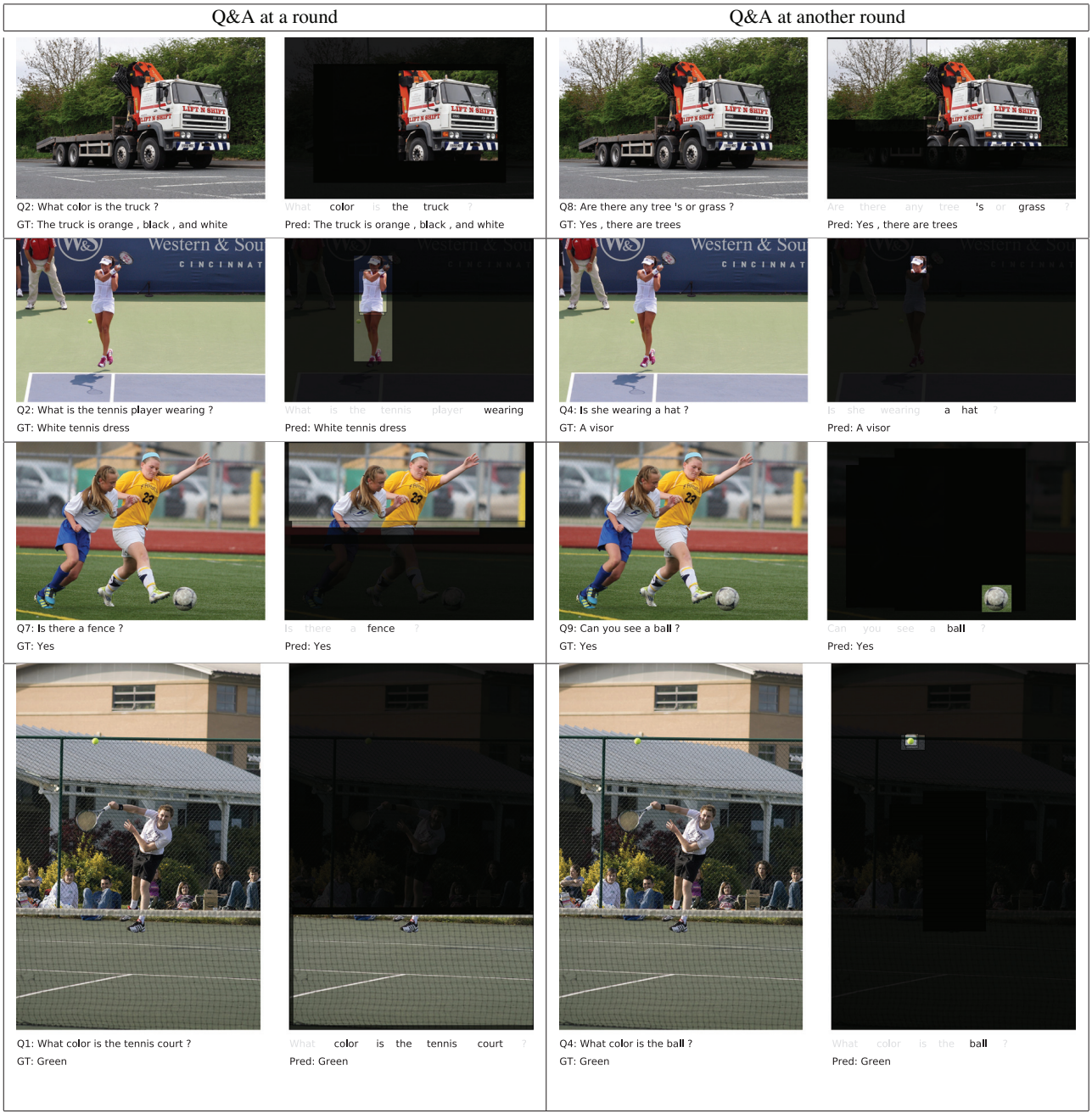

Fig. 2: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

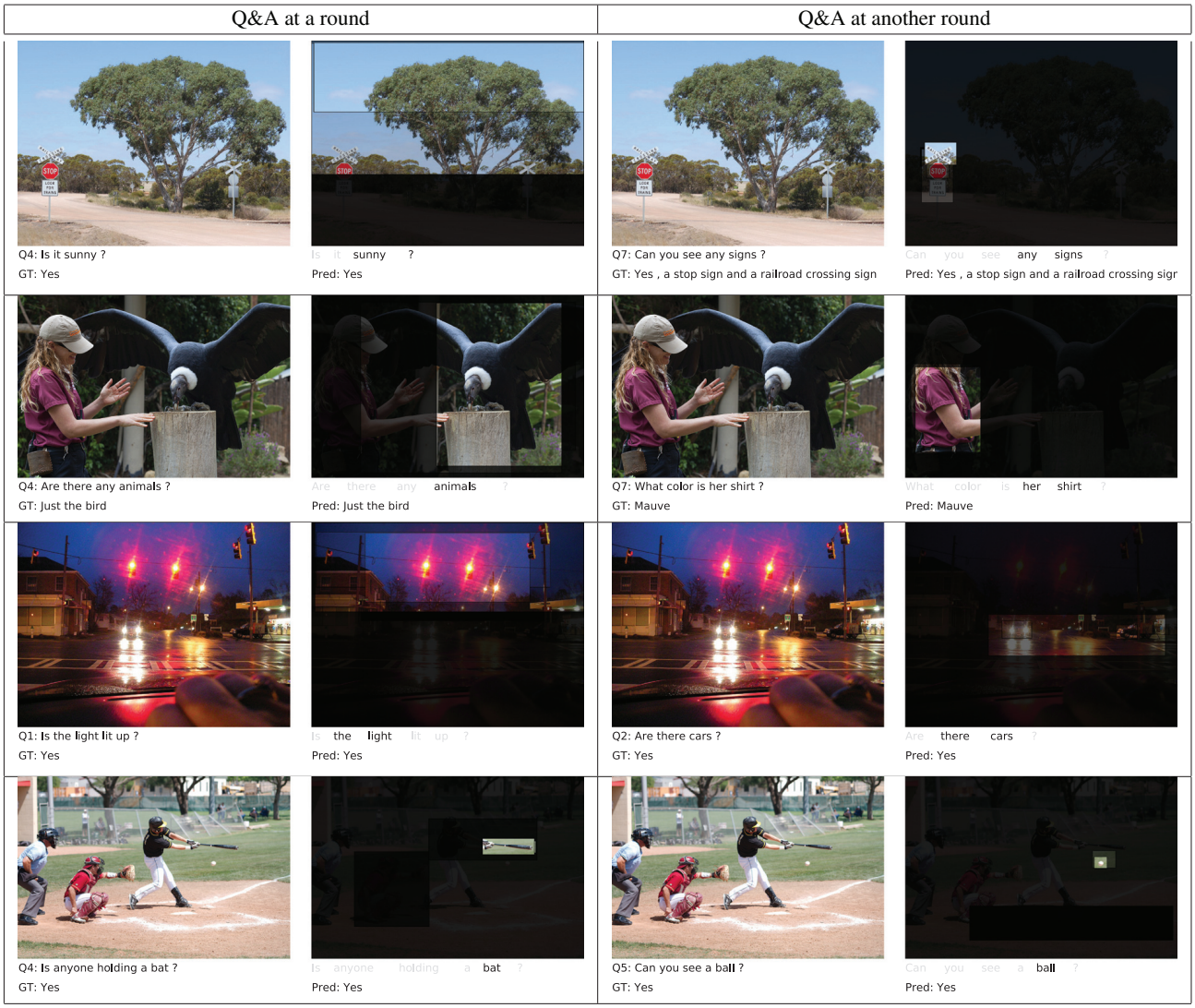

Fig. 3: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

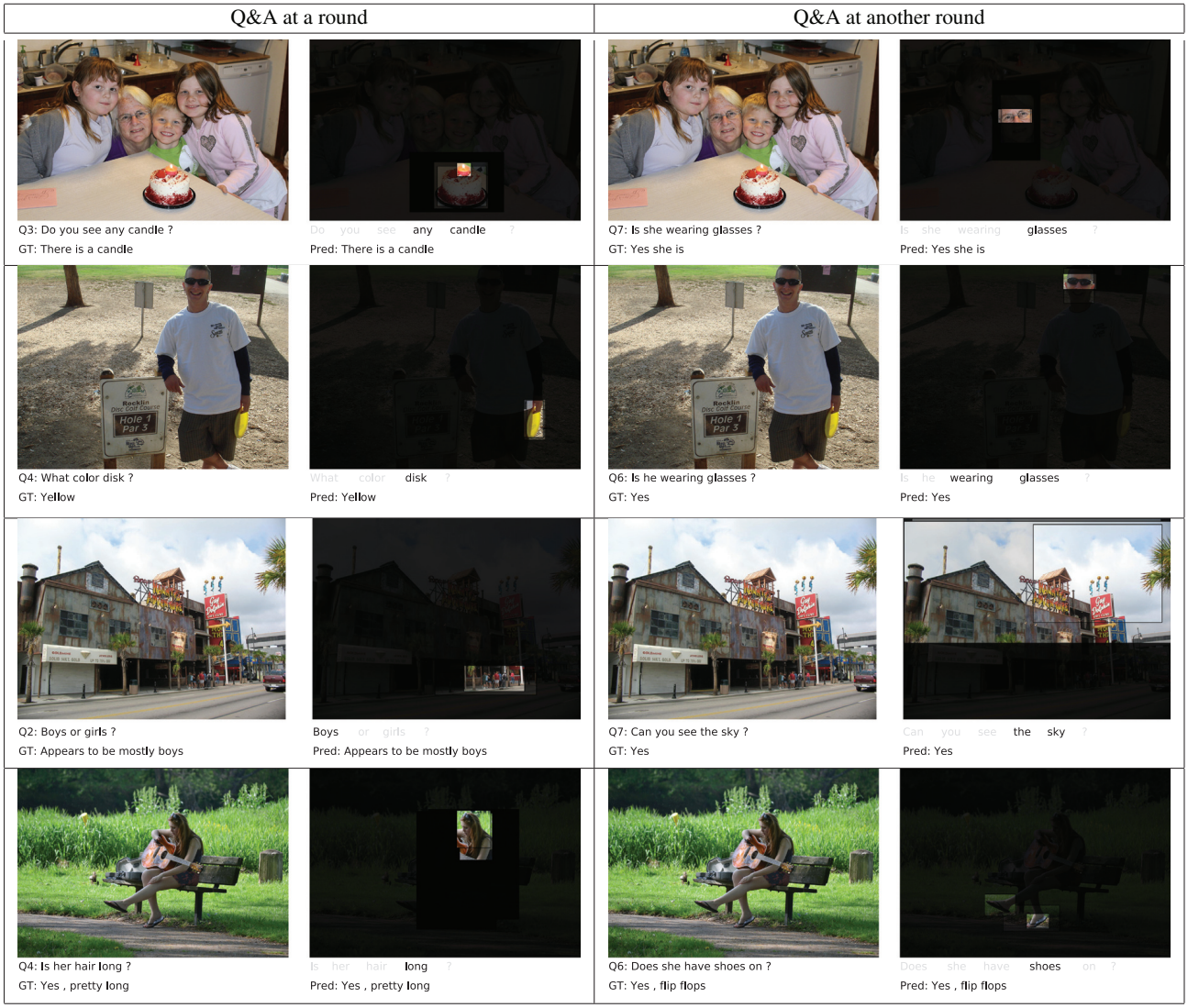

Fig. 4: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

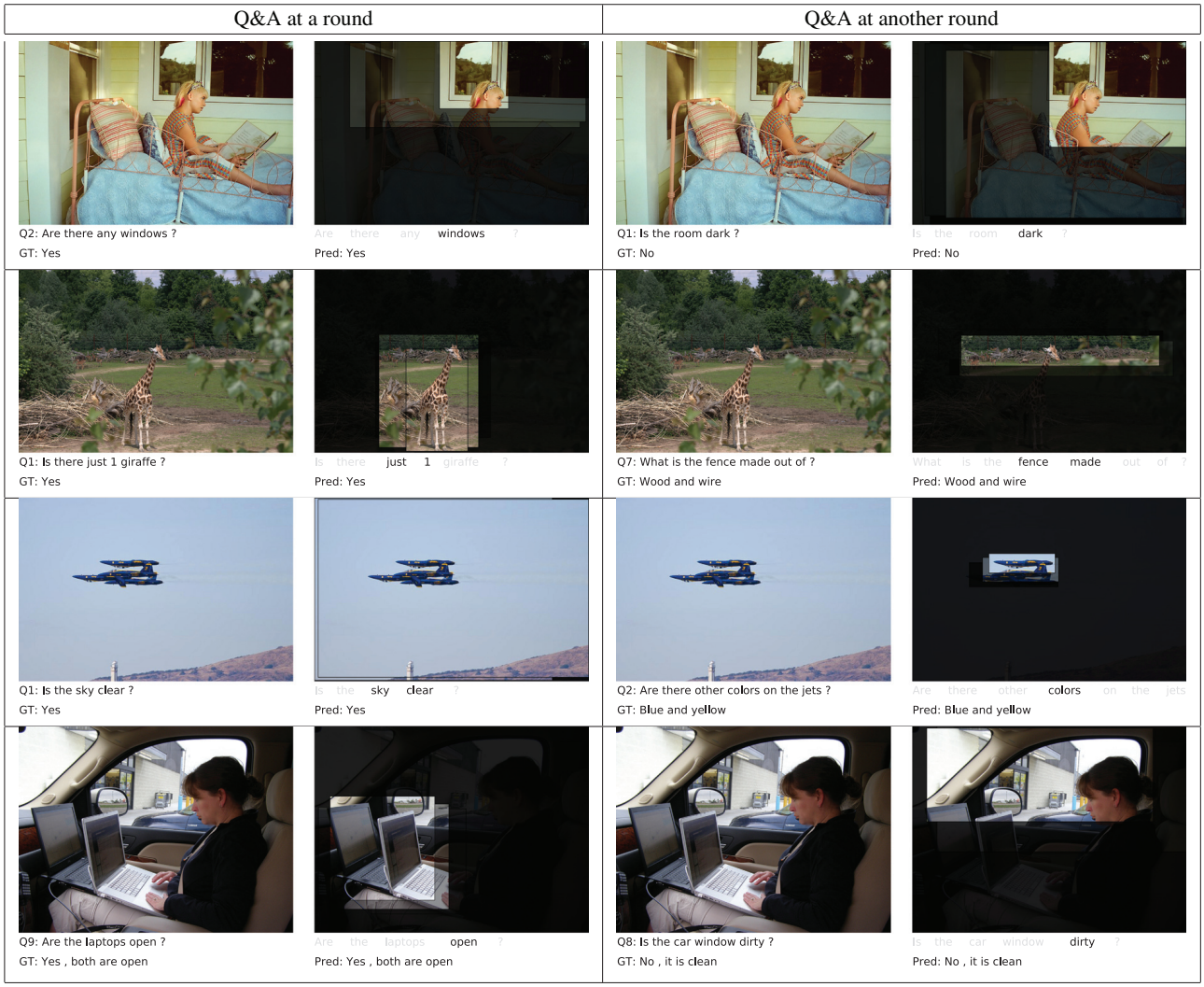

Fig. 5: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

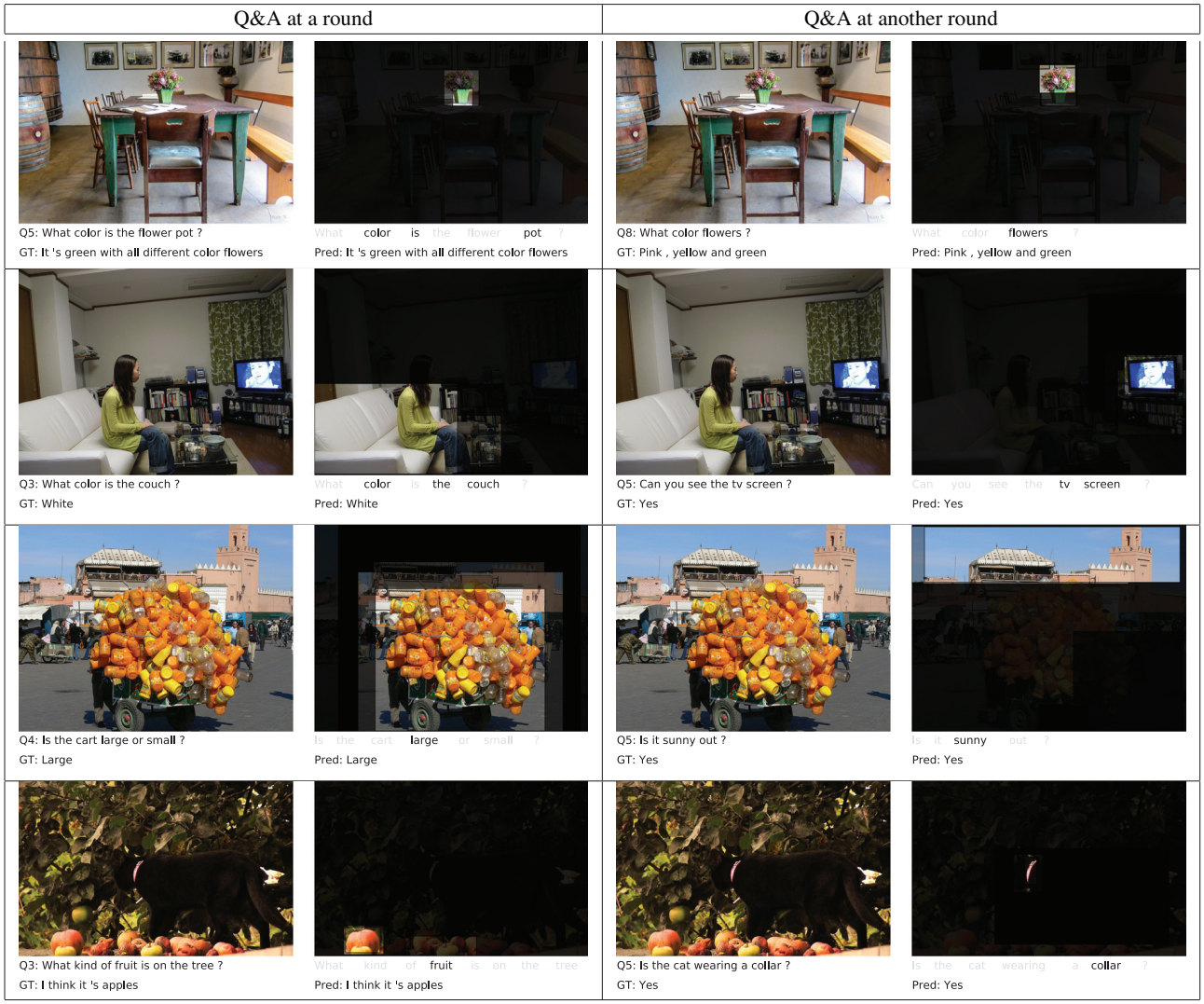

Fig. 6: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

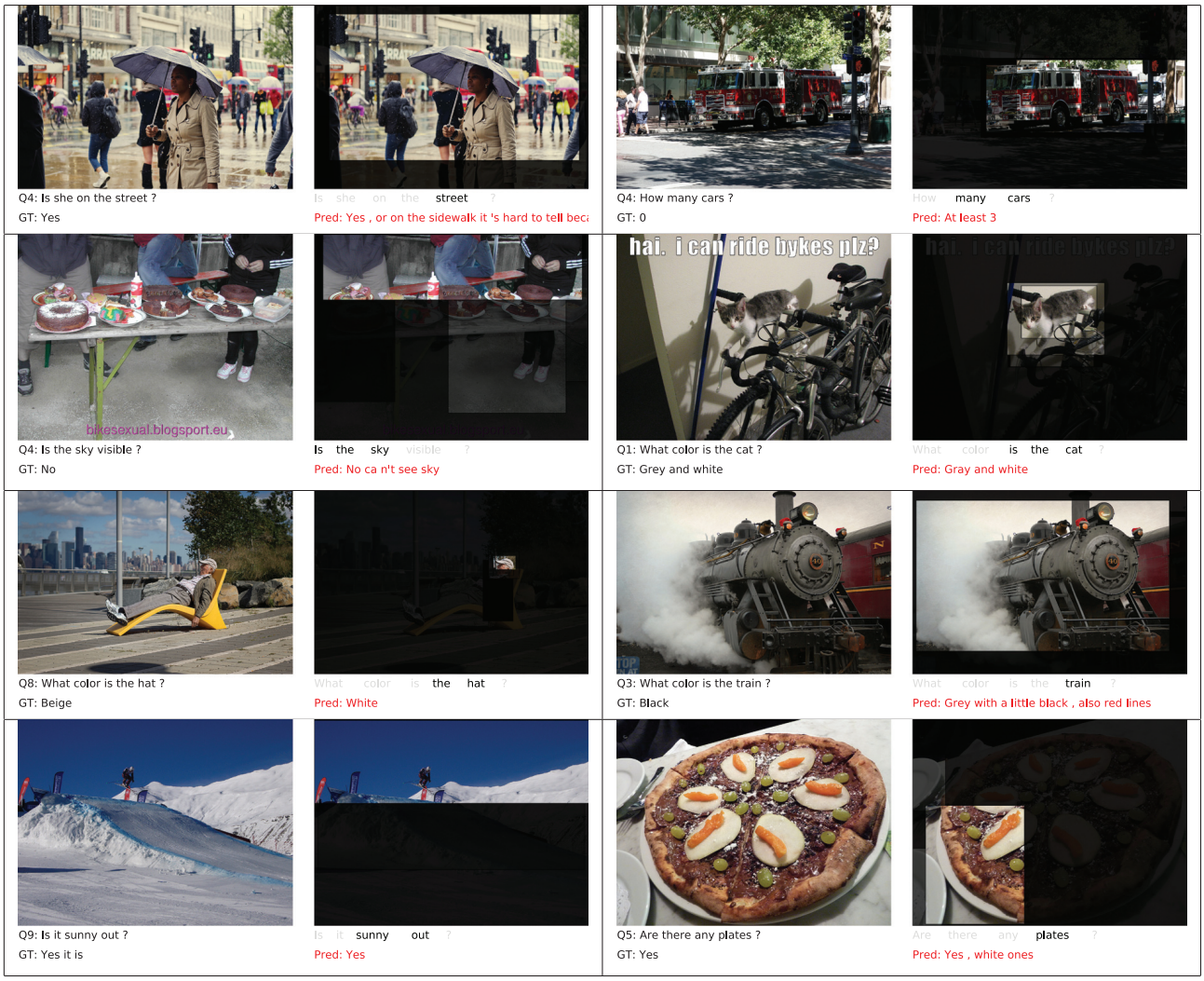

Fig. 7: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.

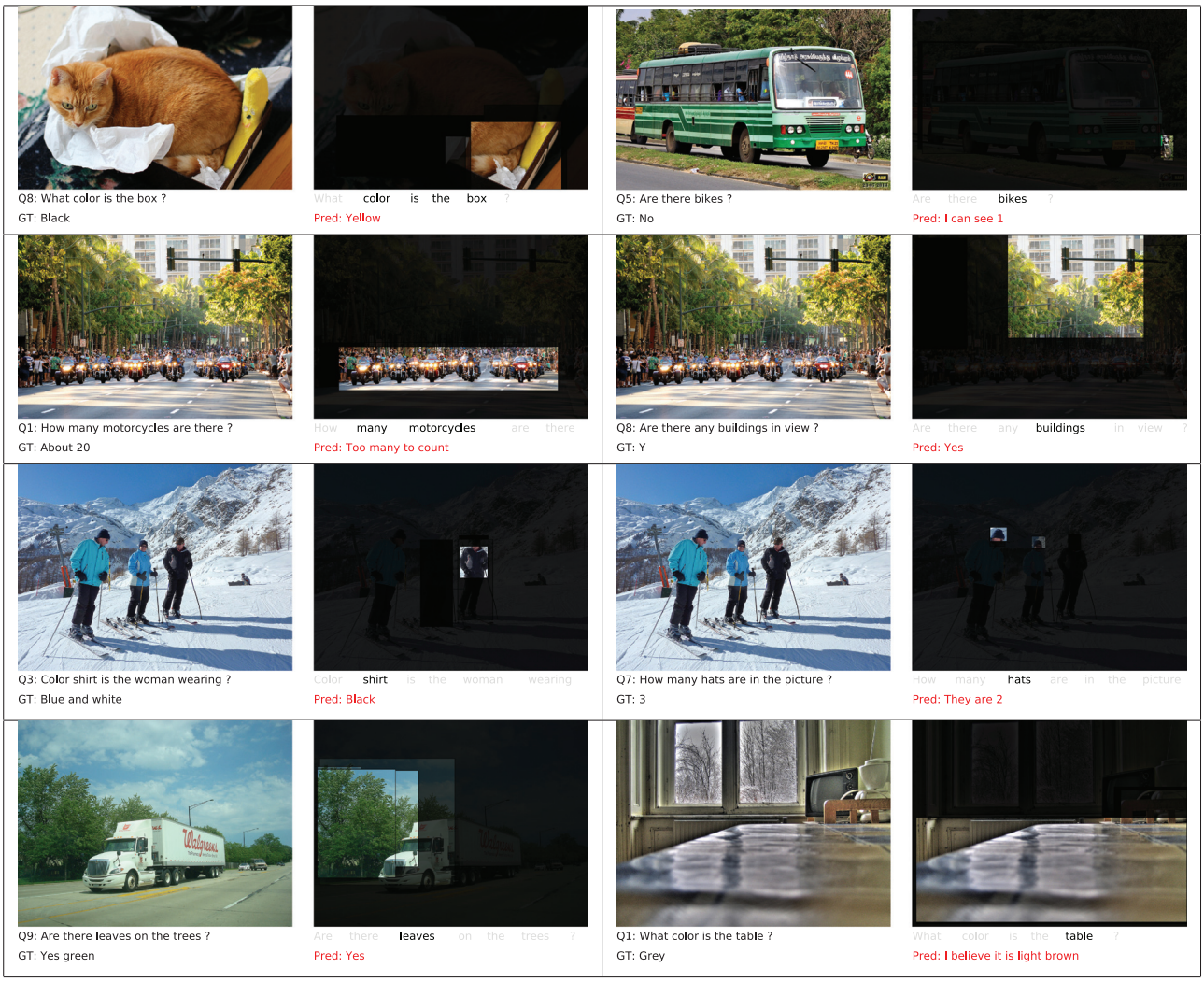

Fig. 8: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.

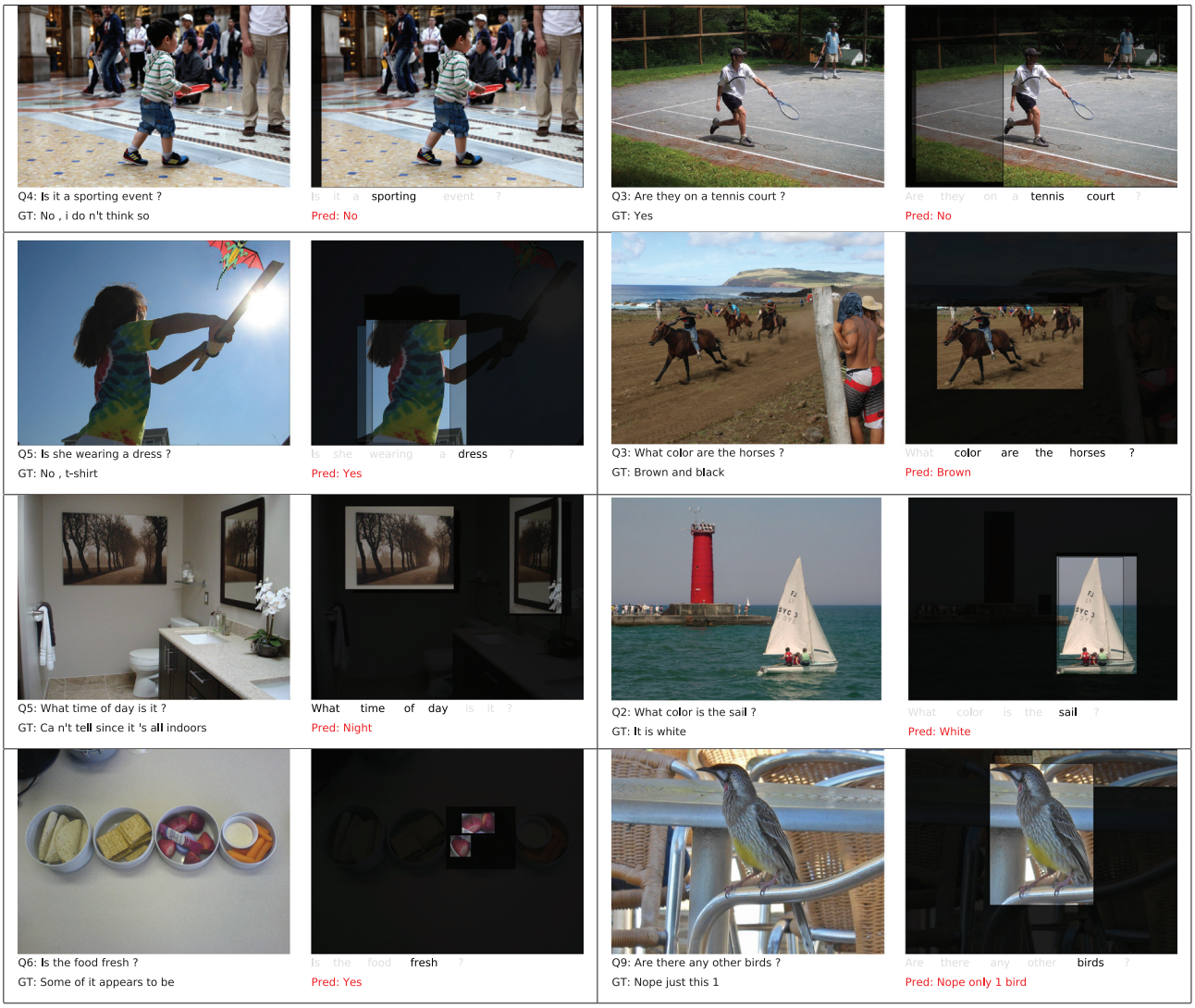

Fig. 9: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.

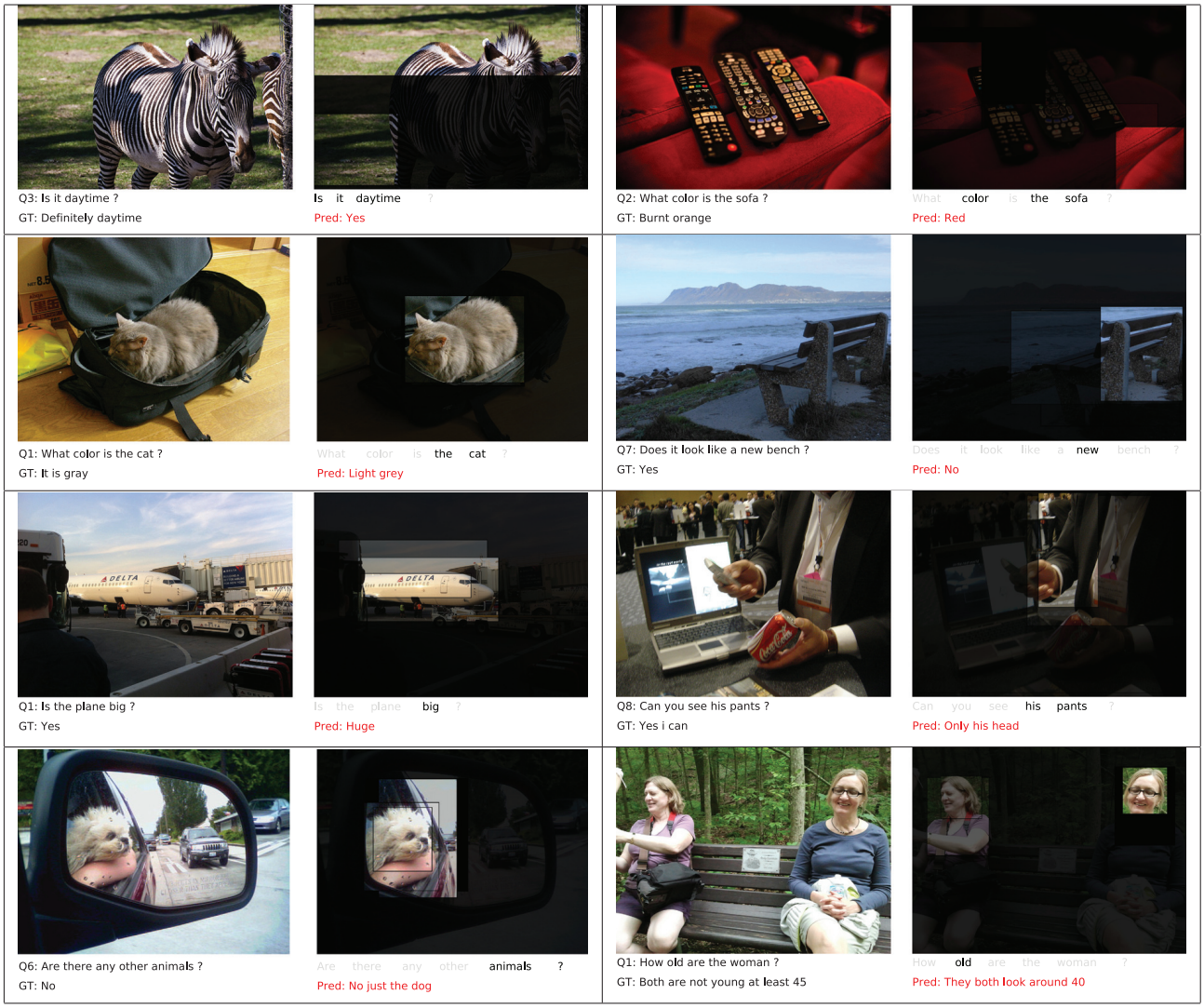

Fig. 10: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.

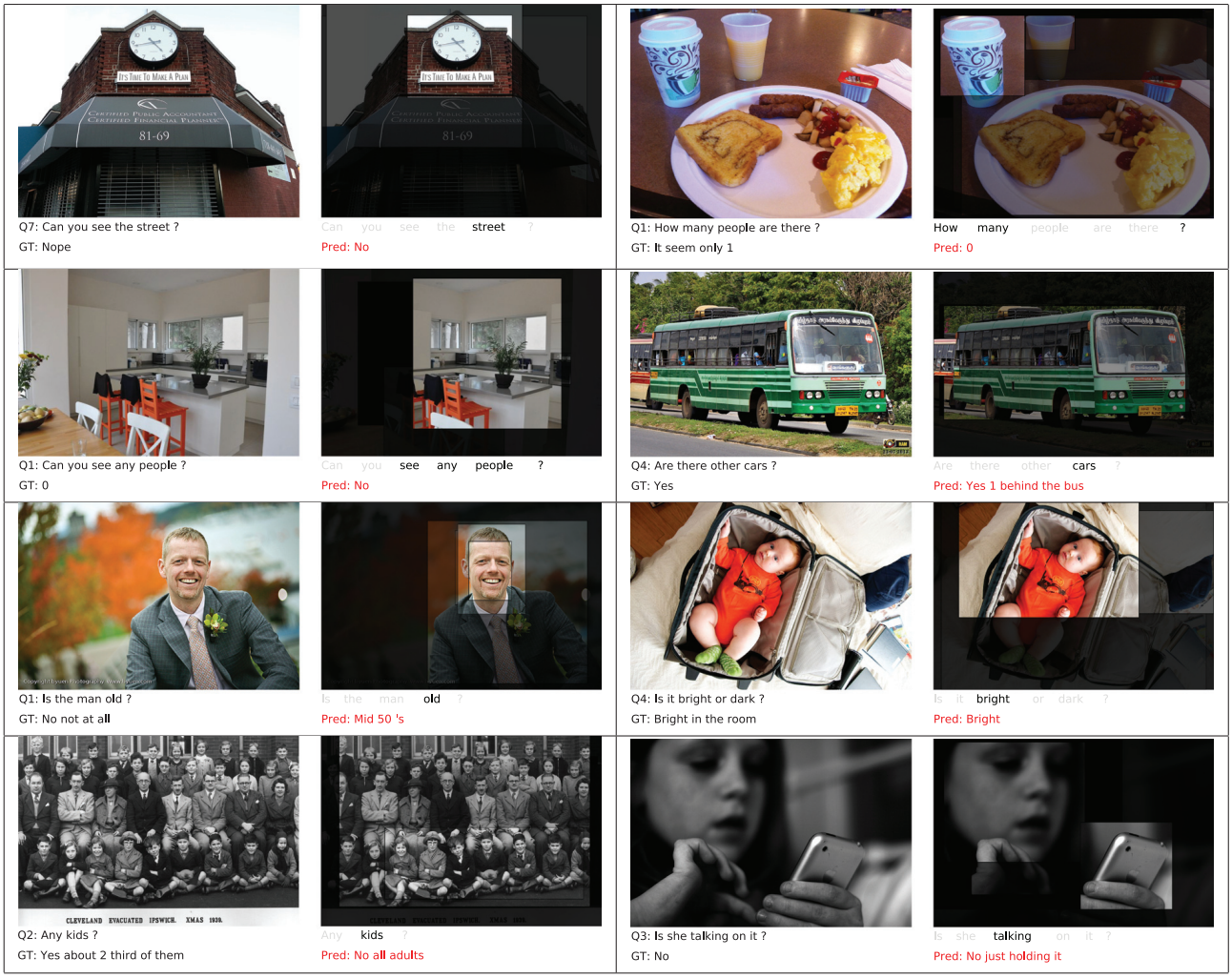

Fig. 11: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.