IET Research Journals

IET Journals
The Institution of Engineering and Technology

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Object Sequences: Encoding Categorical and Spatial Information for a Yes/No Visual Question Answering Task

Shivam Garg[1] *Rajeev Srivastava*[1]

[1] *Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi – 221005 (UP) India*
** E-mail: shivam.gargcd.cse14@itbhu.ac.in*

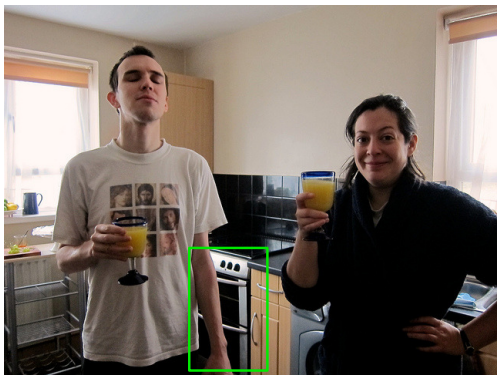
Abstract: The task of Visual Question Answering (VQA) has gained wide popularity in recent times. Effectively solving the VQA task requires the understanding of both the visual content in the image and the language information associated with the text based question. In this paper, we propose a novel method of encoding the visual information (categorical and spatial object information) about all the objects present in the image into a sequential format, which we call as an object sequence. These object sequences can then be suitably processed by a neural network. We experiment with multiple techniques for obtaining a joint embedding from the visual features (in form of object sequences) and language based features obtained from the question. We also provide a detailed analysis on the performance of a neural network architecture using object sequences, on the Oracle task of GuessWhat dataset (a *Yes*/*No* VQA task) and benchmark it against the baseline.

1 Introduction

Visual understanding of the world involves identification of different objects present in a view, their properties, relationship with other objects and many other associated details. With the recent advances in singular tasks related to visual understanding, like image categorization [1], object detection [2] and semantic segmentation [3], the research focus has started shifting to more open ended tasks that require a holistic understanding of the visual content offered by a scene. However, datasets involving just visual cues, i.e. annotated parts of images with words and labels from a fixed sized vocabulary, often fail to capture properties like the complex nature of the actions being performed in the image or the inherent visual ambiguity. On the other hand incorporating human descriptions for images into these datasets, such as [4], [5], provides very rich and open–ended details about images. Therefore, learning systems, when trained to understand and automatically generate these descriptions, acquire an in–depth visual understanding of the image. Such systems learn a joint representation of the visual and language based modalities in these multi–modal tasks.

One such multi–modal task is image captioning, in which the primary goal is to generate a small natural language description of a given image. Modern machine learning techniques, to solve this problem, involve optimization of some kind of a learning model (such as a neural network architecture consisting of RNNs and CNNs [6]) with respect to a quantitative loss function of the performance of this architecture. However, coming up with a quantitative measure for the goodness of the response generated by the model for such open–ended problems is a challenging task, because there exist innumerably many "correct" ways to describe a single image. Visual Question Answering (VQA) alleviates some of this problem by formulating this task in form of an open–ended dialogue with respect to the image.

In visual question answering, these dialogues are structured in form of questions and answers. Though, the questions are allowed to be unconstrained and open–ended, the answers are constrained to a small set of concepts like color, object names [7] or even a simple *Yes*/*No* [8]. This often reduces the problem of evaluating the performance of the system into a simple classification task, without diminishing the challenge of the problem. The questions can still refer to any visual concept in the image and there exist multiple challenges in solving a VQA task [9], such as concept ambiguity, ambiguity in deciding the questioner's frame of reference, incorporating common sense knowledge or even considering societal factors like obtaining a social consensus on the answers given by an artificial system. A successful VQA system would need to solve these sub–tasks in order to engage in multi–modal dialogue. This way, VQA provides a holistic approach to visual understanding.



Question	Answer
Is it a liquid?	No
Is it a person?	No
Is it in the background?	Yes
Is it on the left side?	No
Is it between the people?	Yes
Is it the oven?	Yes

Fig. 1: A typical example in the GuessWhat dataset [8]. The hidden object (an oven in this case) is highlighted by a green bounding box. Following the image is the dialogue between the Questioner and the Oracle.

In this paper we work with GuessWhat [8], a VQA dataset which is centered on a two–player game between a Questioner and an Oracle. The task of the Questioner is to find an unknown object in the given image by asking questions related to it. The Oracle in turn, has to answer these questions. Fig. 1 shows a sample GuessWhat game in progress. The hidden object, an oven in this case, is highlighted with a green bounding box. The Questioner asks polar questions

(questions that can be answered in a *Yes* or *No*) in order to find this unknown object using visual information associated with the scene. The Oracle has access to the knowledge about this hidden object and its job is to answer these questions in *Yes* or *No* mode. Answering such questions requires the Oracle to understand the various aspects about the image and to represent all this information in a format suitable for a classification model. We model this Oracle using a neural network architecture and our main contribution in this paper is the proposal of an effective way of structuring the inherently unstructured visual information obtained from the image into a succinct form, that can be fed into such architectures.

Contributions: We propose a simple and effective way of encoding the spatial and categorical information about all the objects present in an image into sequences. We call these sequences as Object Sequences, and these can be passed into a neural network for further processing. We give the motivation behind object sequences and the algorithm to extract them from images in Section 3.1. We experimentally compare different ways to combine the visual features (obtained from object sequences) with the language features (obtained from questions) into a joint embedding in Section 4.2. In Section 4.3 we provide a detailed experimental analysis of object sequences and benchmark their performance against the baseline for the Oracle task [8].

2 Related Works

The first attempt at the creation of a VQA dataset was DAQUAR [7]. It consists of 12K question–answer pairs where the answers are constrained to colors, counts and a set of objects, and the questions are open–ended. DAQUAR is built on top of the NYU-Depth V2 dataset [10] consisting of about 1400 RGB–D images. In [7], a baseline was setup for DAQUAR. The authors used a generative Bayesian model for answering the questions. They showed that the performance of their system was heavily dependent on its visual recognition capabilities.

In [11], the authors improved significantly upon their previous results on DAQUAR, by using a neural network model. They experimented with various approaches and found that a combination of LSTM [12] for handling language data and ResNet [1] for visual data gave the best results. They also created an augmentation of the original DAQUAR dataset, called as the DAQUAR–Consensus. DAQUAR–Consensus provides five different answers given by five different human annotators, for each question from the original dataset. This was an attempt to bring out the societal factors in VQA, and thus create systems that generate socially acceptable answers to questions. The differences in the human responses in DAQUAR–Consensus also illustrated the inherent ambiguity in the task of question–answering.

An extremely large VQA dataset [13] was later proposed, consisting of 0.25M images, 0.76M questions, and 10M answers, with multiple answers for each question. This dataset contains both real world scenes and abstract cartoon–like images that can help a system learn abstract visual concepts and also provide a controlled experimental setup. This dataset was later modified [14], by associating each question with a pair of complementary images, resulting in different answers. The VQA dataset [13] also hosted an associated challenge. The state of the art on this challenge was achieved by [15]. This model learned a joint RNN–CNN embedding of the input questions and image, with question guided attention modules to solve this task.

This paper [15] also introduces the idea of bottom–up attention, which has been discussed in detail and also applied to image captioning in [16]. Conventional attention mechanisms, [17] (in case of image captioning), [18] (in case of machine translation), focus on part of features based on the partially generated output related to the final task. Bottom–up attention, instead, focuses on relevant image regions based on just the visual properties suited for the problem, oblivious to the context provided by the task. For example, [16] implements bottom–up attention by using a ResNet [1] based RCNN architecture to extract features corresponding to the objects in the image. They, then use the conventional (called as top–down) attention to weigh these bottom–up attention derived features and combine them to encode the image for final VQA task.

Machine learning models often resort to exploiting some kind of bias in the questions itself, rather than using and understanding visual cues when employed for VQA. This is a major hurdle in training them effectively, and affects their usability afterwards. To overcome this problem a different split of the original VQA dataset [13] was proposed in [19]. In this new split, there are fundamentally different types of questions in the training and test splits of the dataset, which discourages such sole dependence on superficial correlation between questions and their answers.

Another very large dataset that combines vision and language is the Visual Genome Dataset [20]. This dataset provides dense annotations on the images in form of detailed relationships between the various objects in the image.

The VQA dataset with which we work in this paper is GuessWhat [8]. For this dataset, the authors establish three different baselines for the three different tasks of modeling a Questioner, an Oracle and a Guesser associated with the dataset. The task of the Questioner is to ask relevant questions to discover the hidden object from the image, the task of the Oracle is to answer these questions and the task of the Guesser is to the guess the hidden object from the list of objects presented to it after the end of above dialogue between the Questioner and the Oracle. In [8], the joint embedding for the visual and language features is obtained by first evaluating the image and question features separately, and then concatenating them into a single vector. An alternate approach was presented in [21], where the language features are incorporated in the visual recognition pipeline (containing a ResNet [1]) using their newly proposed method of Conditional Batch Normalization, early in the process. They tested their method on the datasets [13] and [8], and it achieved significant improvements from the baseline results.

Such dialogue systems can also be modelled as goal based decision making tasks. In such a modelling scheme, the goal of the agent (Questioner, Oracle or the Guesser in GuessWhat) is to take actions in order to achieve its goal. This problem can be solved by the methods of Reinforcement Learning (RL). [22] discusses the creation of an environment, using the previously established baseline models, for applying deep RL to the GuessWhat dataset.

Multi–modal dialogue systems offer a much more closer view to what artificial general intelligence (AGI) would be like. Humans can handle a variety of modalities together, with each one strongly linked to another. AI systems need to learn this strong correspondence between the different modalities in order to be able to replicate human behaviour and cognitive actions.

3 Methodology

3.1 Object Sequences

In this section we present the main contribution of this paper: the method for encoding the categorical and spatial information about the various objects present in the image into object sequences.

A typical image in the GuessWhat dataset contains multiple objects, in addition to the one prime object that is not known to the questioner a priori. Ideally, to answer any questions regarding the image and this prime object, an Oracle model would require the information about all the other objects present in the image as well. One way to feed this information into a neural network is to use a big CNN to extract features for the whole image. This approach was tried in [8], using a pre–trained VGG Network [23], and didn't even perform to the level of relatively simpler approaches. We use object sequences to capture this information.

We define an Object Sequence as a sequence of objects, arranged in the order of the objects' spatial locations in the image. Fig. 2 displays the object sequences, along X and Y axes, for that image. The spatial locations of the objects in the image is calculated by the centers of their respective bounding boxes.

To find the Object Sequences for a given image, first the objects' bounding boxes and their categories need to be identified. This

process is known as object detection and can be carried out by efficient methods, such as the Region Convolutional Neural Network (RCNN) [2]. Once object detection is done, an object sequence is formed, one along each direction, from the object categories put according to the spatial ordering of the objects in that direction. The object categories are converted into word vectors using some pre–trained word embedding like GloVe [24]. Algorithm 1 presents these steps explicitly.

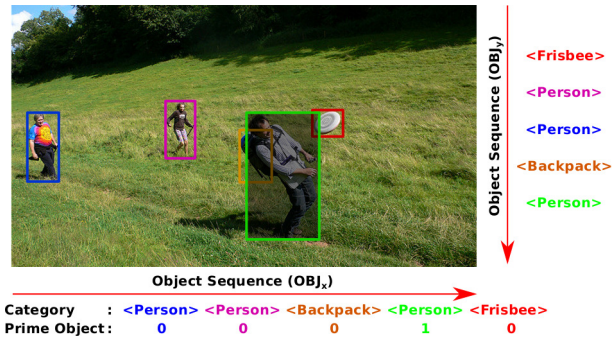


Fig. 2: Object sequences, along X and Y axes, for an image. The sequence is composed of the category of the objects (like `<Backpack>`) and whether the object is prime or not (in form of 0 and 1). Though the prime object information is shown only on X axis, it is incorporated in the sequences along both the axes.

Algorithm 1 Extracting Object Sequences from an image

1: Detect various objects in the image and obtain their categorical information along with their bounding boxes.
2: Sort the detected objects along X and Y axes separately, using the centers of their respective bounding boxes.
3: **for** each identified object **do**
4: Convert the object category into a word vector.
5: **if** Object is prime **then**
6: Append 1 to the categorical word vector.
7: **else**
8: Append 0 to the categorical word vector.
9: Form a separate sequence out of these categorical word vectors for each ordering identified in Step 2.

The motivation behind this kind of encoding is that many of the questions asked in GuessWhat dataset inquire about spatial information. Such as the questions: *"Is it in the background?"* or *"Is it between the people?"* from Fig. 1. Object sequences explicitly encode information, needed to answer such questions, in their structure. Further, if the learning model is powerful enough, it may even be able to map semantic concepts, like **riding** in the question *"Is the person riding the horse?"*, to the relative ordering of `<Person>` and `<Horse>` in the Object Sequence along Y axis. These object sequences are handled by Recurrent Neural Networks (RNNs) in our experiments as explained in the next section.

3.2 Extracting Features from Sequential Data

Once the object sequences are extracted from the image, they are converted into fixed sized feature vectors using an RNN. RNNs are suitable for handling sequential data. Fig. 3 shows the process of a single RNN unfolding over the whole sequence length. An RNN has a hidden state which keeps track of all the important and relevant information about the part of the sequence observed till that point in the RNN. This way the RNN is able to capture the most important and relevant information about the whole sequence in its hidden states.

In our experiments we use Long Short Term Memory (LSTM) [12], which is an extended type of RNN. An LSTM is able to capture long range dependencies amongst the elements present in a sequence.

We also use Bidirectional LSTM (BiLSTM), a kind of Bidirectional RNN [25]. A BiLSTM consists of two separate LSTMs which process the input sequence in opposite directions. Consider the input sequence $a_1, a_2, \ldots, a_n$. First LSTM would process it in the forward direction from 1 to n and generate a sequence of output state vectors $o^{(1)}_{1\to1}, o^{(1)}_{1\to2}, \ldots, o^{(1)}_{1\to n}$. Similarly, the second LSTM would process the same input in the backward direction from n to 1, generating output state vectors $o^{(2)}_{1\leftarrow n}, o^{(2)}_{2\leftarrow n}, \ldots, o^{(2)}_{n\leftarrow n}$. Here, $o^{(1)}_{1\to k}$ represents the output corresponding to the first LSTM's k^{th} cell. The notation $1 \to k$ in $o^{(1)}_{1\to k}$ signifies that it captures information for the subsequence $a_1, a_2, \ldots, a_k$. Similarly, $o^{(2)}_{k\leftarrow n}$ represents the output corresponding to the second LSTM's k^{th} cell and $k \leftarrow n$ signifies that it captures information about the subsequence $a_n, a_{n-1}, \ldots, a_k$. The final output of the BiLSTM is then formed by the concatenation of these two output sequences: $(o^{(1)}_{1\to1} \bowtie o^{(2)}_{1\leftarrow n}), (o^{(1)}_{1\to2} \bowtie o^{(2)}_{2\leftarrow n}), \ldots, (o^{(1)}_{1\to n} \bowtie o^{(2)}_{n\leftarrow n})$, where $\bowtie$ is the concatenation operator (refer Section 3.3).

All the models presented in this work, except those marked by **[Prime–embed biLSTM]** (refer Section 4.3.1), use an individual LSTM to encode sequences (both object sequences and questions) and pick the last output state vector of the LSTM as its final output (as shown in Fig. 3), since the last output state vector of the LSTM encodes all the information in the sequence uptill the end.

Models marked by **[Prime–embed biLSTM]**, use a BiLSTM to encode Object Sequences into feature vectors. In these type of models, the object sequences don't contain the prime object information appended as a 0/1 bit to the categorical word vector (as opposed to Algorithm 1). Instead, to encode the information about the prime object, we pick the output state vector from the BiLSTM, corresponding to the index of the prime object in the object sequence. So if p is the index of the prime object in the object sequence and o_p is the corresponding output state vector of the BiLSTM, o_p is given by $o^{(1)}_{1\to p} \bowtie o^{(2)}_{p\leftarrow n}$. Now, $o^{(1)}_{1\to p}$ encodes information about sequence elements from index 1 upto p and $o^{(2)}_{p\leftarrow n}$ encodes sequence information from index n to p. This implies that the whole sequence information is captured by the single feature vector o_p. In addition to this, o_p also contains the position, in form of index p, of the prime object.

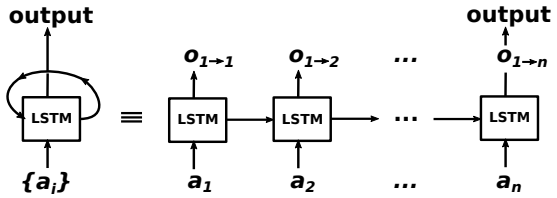


Fig. 3: Extraction of features from sequential data using an LSTM. The unfolding of LSTM on the sequence $\{a_i\}$ is shown. The sideways arrows connecting two consecutive cells of the LSTM illustrate the input of the information about the symbols observed in the past. The output of the LSTM in our experiments is taken as the output state vector of the LSTM cell at the last element in the sequence.

3.3 Models and Embedding Techniques

In this section we explain the different embedding techniques we have used to combine visual and language–based features. We also give the architecture of the neural networks used in our experiments.

The Oracle task of GuessWhat effectively reduces to a classification problem. So the crux of the problem is to calculate an effective

multi–modal embedding which contains both the visual information (from the image) and the language–based information (from the question). Put another way, given separately the visual and textual features, we need to find a way to combine them into a single feature vector. We consider five different approaches for calculating this embedding.

In the first three approaches, the hidden states of the LSTMs, used for calculating features from sequential data, are initialized with zero vectors. We display these approaches with the help of the following equations:

$$\text{Concatenation} = \boldsymbol{Obj_x} \bowtie \boldsymbol{Obj_y} \bowtie \boldsymbol{Q} \quad (1)$$

$$\text{Sum} = \boldsymbol{W_1}^\top(\boldsymbol{Obj_x} \bowtie \boldsymbol{Obj_y}) + \boldsymbol{Q} \quad (2)$$

$$\text{Dot} = \boldsymbol{W_1}^\top(\boldsymbol{Obj_x} \bowtie \boldsymbol{Obj_y}) \odot \boldsymbol{Q} \quad (3)$$

where $\boldsymbol{Obj_x}$ and $\boldsymbol{Obj_y}$ are the LSTM's output for object sequence along X and Y axes respectively, and $\boldsymbol{Q}$ is the LSTM's output for the question. $\boldsymbol{W_1}$ is a weight matrix which is used to make the dimensions of vector $(\boldsymbol{Obj_x} \bowtie \boldsymbol{Obj_y})$ equal to that of $\boldsymbol{Q}$. $(\boldsymbol{A}_{m\times 1} \bowtie \boldsymbol{B}_{n\times 1})$ represents the concatenation of vector $\boldsymbol{A}$ with vector $\boldsymbol{B}$ to create a vector of dimensions $(m+n)\times 1$. The symbol $+$ represents elementwise addition of two vectors and $\odot$ represents elementwise multiplication of two vectors.

Eq. (1) refers to the **Concatenation Model**, also shown in Fig. 4. The other two models are the **Sum Model**, given by Eq. (2), and the **Dot Model**, given by Eq. (3). In all three models, the extraction of the features is done similarly. First, various visual features of the image like object category, spatial location of the prime object and the Object sequences of the image, along with textual features from the Question are obtained. Then the features are combined to form a joint visual–language embedding vector. The visual and language features in both the Dot and Sum model are merged together via elementwise operations as mention in Eq. (2) and Eq. (3) respectively, whereas they are concatenated together in the Concatenation Model. Once this joint embedding vector is obtained, it is passed through a fully connected neural network (labelled by MLP in Fig. 4) and then a Softmax layer to obtain the final output of *Yes* or *No*.

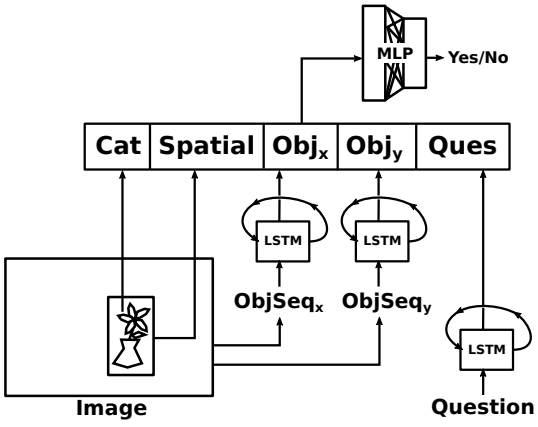


Fig. 4: The Concatenation Architecture for the Oracle model. In this model, the visual feature vector (from the image) is concatenated with the language feature vector (from the question) to form a single large embedding vector. The embedding vector is then passed through a fully connected neural network (labelled by MLP) to give a binary output of *Yes* or *No*. (The hidden states of all the LSTMs are initialized with zeros.) This model is inspired from the embedding used in [8].

We call the other two models, for obtaining the embedding vectors, as **Sequential Models**. In these models, the hidden state of the LSTM for calculating the visual (or textual) features is initialized with the output of the LSTM applied on textual (or visual) features. Following equations represent these models:

$$\text{Sequential}_{Obj_{x,y}\hookrightarrow Q} = \boldsymbol{W_2}^\top(\boldsymbol{Obj_x} \bowtie \boldsymbol{Obj_y}) \hookrightarrow \boldsymbol{Q} \quad (4)$$

$$\text{Sequential}_{Q\hookrightarrow Obj_{x,y}} = (\boldsymbol{W_3}^\top \boldsymbol{Q} \hookrightarrow \boldsymbol{Obj_x}) \bowtie (\boldsymbol{W_3}^\top \boldsymbol{Q} \hookrightarrow \boldsymbol{Obj_y}) \quad (5)$$

where, $(\boldsymbol{A} \hookrightarrow \boldsymbol{B})$ represents the LSTM output of $\boldsymbol{B}$, when the vector $\boldsymbol{A}$ is used to initialize the hidden state of the LSTM associated with $\boldsymbol{B}$.

Eq. (4) corresponds to the first sequential model, also shown in Fig. 5. In this model, the hidden state of the Question LSTM is initialized with the concatenated outputs obtained from the two Object Sequence LSTMs. The output of the Question LSTM is then taken as the joint embedding vector. This embedding vector is passed through an MLP and then a Softmax layer to give the final binary output of *Yes* or *No*. Eq. (5) corresponds to the second sequential model in which the output from the Question LSTM is used to initialized the hidden states of both the Object Sequence LSTMs.

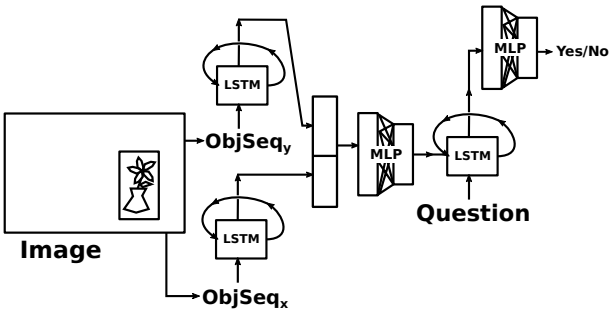


Fig. 5: The Sequential Architecture for the Oracle model. In this model, the LSTM extracted features from Object sequences of the image are used to initialize the hidden state of the LSTM for Questions. The features extracted by this LSTM from the Question then represent the joint embedding vector. The embedding vector is passed through a fully connected neural network (labelled by MLP) to give a binary output of *Yes* or *No*. This model is inspired from the approach taken in [6] for image captioning.

4 Results and Discussion

In this section we study the performance of Object Sequences and different embedding techniques on the task of Oracle in the GuessWhat [8] dataset. The aim for all the models is to give an answer of *Yes*/*No* (classify in binary classes *Yes* and *No*) given a question and the related image (as shown in Fig. 1). We report the accuracy of this binary classification problem in our results.

4.1 Experimental Details

We evaluate our approach on the task of Oracle in the GuessWhat dataset, which is split into separate training (580K question–answer pairs), validation and test splits (each consisting of 120K question–answer pairs). GuessWhat provides these question–answer pairs on top of 66K images from the MS COCO [5] dataset. Each image consists of 2 to 20 objects with a mean of 8 objects per image. The average length of a question is 5 words and about 96.6% questions have a word length of less than or equal to 10 words. Further, the answer for each question is one of *Yes* (46%), *No* (52%) or *N/A* (2%) in the dataset. So to simplify the experimentation, we discard question–answer pairs with questions whose length is longer than 10 words or whose answer is *N/A*. This doesn't alter the dataset in any significant manner.

We extract upto three different types of object sequences. Two are along the X and Y axes of the image. The third one, which considers the order of the size of the bounding boxes, can be interpreted as being the object sequence along the Z axis, i.e. the depth of the image. For extracting the object sequences from the images, we need the information about object categories and their bounding boxes. In this paper we acquire this information from the GuessWhat annotations themselves. This way, the experiments focus purely on the method of object sequences, avoiding any noise coming from possibly erroneous visual recognition. An object detection network (like RCNN [2]) could have been used instead with minimal changes in the procedure of forming the object sequences.

All the words (part of the question or object categories) are converted into word vectors of size 50, using GloVe's Wikipedia+Gigaword 6B dataset [24]. For handling unknown words, we create a special word vector `<UNK>` which is equal to the average of all the 50 sized word vectors present in the GloVe dataset.

We experimented with batch sizes of {64, **256**, 512, 1024}, and found 256 to be the best in terms of computational speed and final result accuracy. So, in all the experiments that we present here, we have taken the batch size as 256. To batchify the sequences (both object sequences and questions) before passing them into an LSTM, we pad the smaller length sequences (with zero vectors at the end) to make their length 10 (for questions) or 20 (for object sequences). We pick the output of the LSTM at the end of the padded sequence as the LSTM's final output (refer Fig. 3). Though, we found in experiments, done on smaller models, that picking the output state of the LSTM at the end of the actual unpadded sequence as its final output, as opposed to the end of the zero padded sequence, didn't give much improvements in the accuracy (on the downside, it lead to an increase in training time). This illustrates that LSTMs can effectively learn to ignore the zero vectors used to pad different length sequences.

Table 1 Accuracy results (in %) on varying the LSTM's hidden layer size applied to the Oracle task of GuessWhat. All the models were trained using ADAM, with an $\texttt{lr} = 1 \times 10^{-3}$ and $\texttt{weight decay} = 1 \times 10^{-4}$, for 40 epochs. Training and Validation refer to the accuracy scores obtained on the respective splits of the GuessWhat dataset. (The symbols in Architecture column are explained in Section 4.3.1.)

Architecture **(Concatenation)**	(Layer Size)	Training	Validation
$Q + Spat + Cat$	1	76.1	76.1
$Q + Spat + Cat$	2	76.0	76.5
$Q + Obj_{x,y}$	1	74.7	74.5
$Q + Obj_{x,y}$	2	74.2	73.9

We experimented with LSTM's hidden layers size of {**1**, 2}. Both the models gave comparable results as shown in Table 1. We choose the single layered LSTM for our final experiments. We use the optimizer ADAM [26] and train all the networks for 40 epochs. We train our models on the full training split, use validation split for hyperparameter tuning and test our final models on the test split of the dataset. The weight decay (weighing parameter for L_2 regularization) is set equal to 1×10^{-5} as this gave the best results; for comparison see the models (hidden layer size $= 1$) trained with weight decay $= 1 \times 10^{-4}$ in Table 1 and compare them with the equivalent models trained with weight decay $= 1 \times 10^{-5}$ in Table 3. Learning rate was chosen from $\{1 \times 10^{-3}, 5 \times 10^{-4}\}$ depending on whichever performed better, separately for each model. All the networks are implemented in the deep learning framework PyTorch [27].

4.2 Comparison of Different Embedding Techniques

In this section we compare the different techniques for obtaining a joint embedding of visual and textual features as mentioned in Section 3.3. Table 2 shows the accuracy of using different embedding techniques on the Oracle task of GuessWhat [8].

Table 2 Accuracy results (in %) for different embedding techniques on the Oracle task of GuessWhat. Training, Validation and Test refer to the accuracy scores obtained on the respective splits of the GuessWhat dataset. (The symbols in Architecture column are explained in Section 4.3.1.)

Architecture	Training	Validation	Test
$Q + Obj_{x,y}$ [Concatenation]	83.2	78.9	78.6
$Q + Obj_{x,y}$ [Sum]	81.2	78.0	78.1
$Q + Obj_{x,y}$ [Dot]	83.4	79.6	79.5
[Sequential$_{Obj_{x,y} \hookrightarrow Q}$]	82.4	79.1	79.2
[Sequential$_{Q \hookrightarrow Obj_{x,y}}$]	81.7	79.0	79.0

In Table 2, **[Concatenation]** refers to the model described in Eq. (1), **[Sum]** to Eq. (2), **[Dot]** to Eq. (3), **[Sequential$_{\boldsymbol{Obj_{x,y} \hookrightarrow Q}}$]** to Eq. (4) and **[Sequential$_{\boldsymbol{Q \hookrightarrow Obj_{x,y}}}$]** to Eq. (5).

From this table, we can see that all the models perform closely. Out of the first three, **[Dot]** performs the best, possibly due to the non–linear nature of its operation. **[Sum]** and **[Concatenation]** perform equally well. This can be attributed to the fact that both involve linear operations whose output is then passed into a fully connected neural network (which itself has a Linear layer at its beginning). Out of the Sequential models, first one performs better as compared to the second one.

4.3 Benchmarking of Object Sequences

In this section we benchmark the performance of object sequences with the Oracle baseline mentioned in [8] and run ablation experiments to understand the relative importance of each constituent feature in forming Object Sequences. We first explain Tables 3, 4 and 5 and then give a discussion on the results obtained.

4.3.1 Results: The Architecture column in Tables 3, 4 and 5, specifies which features are being used in the model. We explain the meaning of the individual symbols as follows:

$\boldsymbol{Q}$ refers to the output of an LSTM applied to the question.

$\boldsymbol{Spat}$ refers to the spatial features of the prime object. It is a vector of length 8, composed of different properties of the bounding box associated with the prime object. It is defined as follows:

$$[x_{min}, x_{max}, y_{min}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}].$$

$\boldsymbol{Cat}$ refers to the category of the prime object. It is calculated by evaluating the word embedding for the prime object's category using GloVe.

$\boldsymbol{Obj_{x,y}}$ refers to the concatenated outputs of the LSTMs applied to the object sequences along X and Y axes separately. Using our notation from Section 3.3, we can write $Obj_{x,y} = Obj_x \bowtie Obj_y$.

$\boldsymbol{Obj_{x,y,z}}$ refers to $Obj_x \bowtie Obj_y \bowtie Obj_z$. Here, Obj_z represents the LSTM features for the object sequence along depth of the image. Obj_z is calculated (approximately) by ordering the objects using the sizes (area) of their bounding boxes.

$\boldsymbol{Obj_{x,y} \hookrightarrow Q}$ and $\boldsymbol{Obj_{x,y,z} \hookrightarrow Q}$ refer to Sequential type of models as given in Eq. (4) and an extension to include object sequences along Z axis respectively.

$\boldsymbol{Q \hookrightarrow Obj_{x,y}}$ and $\boldsymbol{Q \hookrightarrow Obj_{x,y,z}}$ refer to Sequential type of models as given in Eq. (5) and an extension to include object sequences along Z axis respectively.

[Prime–embed matrix] marker incorporates the prime object information in the object sequences differently than by appending a 0/1 bit to the object category vectors. Appending just a single 0/1 bit to 50×1 dimensional categorical vector seems suboptimal. This marker provides the prime object information by using

two embedding matrices $M_{50\times50}$ and $W_{50\times50}$. Categorical vectors of non–prime objects are multiplied by embedding matrix M and those of prime objects by W.

For example, consider the object sequences of Fig. 2; using the notation (<category>, prime), these become:

```
#:     1        2        3        4        5
X: (<P>,0) (<P>,0) (<B>,0) (<P>,1) (<F>,0)
Y: (<F>,0) (<P>,0) (<P>,0) (<B>,0) (<P>,1)
```

Following the above described procedure, we would obtain:

X: (M<P>) (M<P>) (M<B>) (**W**<P>) (M<F>)
Y: (M<F>) (M<P>) (M<P>) (M<B>) (**W**<P>)

The resulting object sequences are processed as usual by an LSTM. The matrices M and W are trainable parameters of the model.

[Prime–embed biLSTM] marker incorporates prime information by using a BiLSTM to process the object sequences, as explained in the Section 3.2.

As an example, consider the first pair of object sequences given in the previous marker. The BiLSTM would process these sequences without the prime bit information and pick the outputs of the 4^{th} and the 5^{th} BiLSTM cells for the object sequences along X axis and Y axis respectively, as their encoded feature vectors.

[No prime] marker refers to the model in which the prime object information is not encoded in the object sequences (contrary to what is shown in Fig. 2). Thus, the object sequences reduce to a list of objects arranged by their spatial coordinates in the image.

[No order] marker refers to the model in which the objects are randomly arranged in the object sequence during each epoch. This way, the object sequences reduce to a list of all the objects in the image, along with the prime object, without any spatial information.

[Non–prime zeros] marker refers to the model in which the object sequences are formed by passing zero vectors (in place of the corresponding word vectors) for non–prime objects. This way, such a model lacks categorical information about all the non–prime objects. Instead, it contains information about the number of objects in the image, the category of the prime object and its relative position amongst all the other objects present in the image.

[Non–prime rand] is similar to the model with **[Non–prime zeros]** except that a random vector (with values in range $[-1, +1]$) is passed instead of a zero vector for all the non–prime objects.

[Order Index] marker refers to the model in which the individual elements of the object sequence also contain the index of each object. Note that this index is different from the index mentioned in the **[Prime–embed biLSTM]** marker. Here, the index serves as a unique identifier for each object, whereas in the **[Prime–embed biLSTM]**, it serves to identify the position of prime object in the sequence.

Again, consider the object sequences of Fig. 2, using the notation (<category, index>, prime):

```
X: (<P,1>,0) (<P,2>,0) (<B,3>,0) (<P,4>,1) (<F,5>,0)
Y: (<F,5>,0) (<P,2>,0) (<P,1>,0) (<B,3>,0) (<P,4>,1)
```

We represent the index, in this marker, as a one–hot vector to make it nominal (containing no inherent order). So, index 3 above would actually be represented by $[0, 0, 1, 0, 0]$. Incorporating this information makes it unambiguous, that which object is in what order along the different axes, even in the case where multiple objects belong to the same category (as is the case of <Person> in the above example).

4.3.2 Discussion: Table 3 displays the performance of Concatenation models on the Oracle task. The first block in this table considers models without object sequences. $\boldsymbol{Q + Spat + Cat}$ is the baseline for the Oracle model [8]. From the accuracy of $\boldsymbol{Q}$, we can see that the database has a certain bias, due to which this model is able to achieve an accuracy of greater than 50% solely from the question. Further, we find that the $\boldsymbol{Cat}$ is a more important feature than $\boldsymbol{Spat}$.

Table 3 Accuracy results (in %) for different Concatenation models (refer Section 3.3) on the Oracle task of GuessWhat. Training, Validation and Test refer to the accuracy scores obtained on the respective splits of the GuessWhat dataset.

Architecture **(Concatenation)**	Training	Validation	Test
Q	60.7	59.6	59.9
$Q + Spat$	67.5	66.8	67.1
$Q + Cat$	77.3	75.0	75.1
$Q + Spat + Cat$ [Baseline]	79.8	78.1	**78.0**
$Q + Obj_{x,y}$	83.2	78.9	78.6
$Q + Obj_{x,y}$ [Prime–embed matrix]	83.0	79.1	**78.9**
$Q + Obj_{x,y}$ [Prime–embed biLSTM]	82.2	79.0	78.8
$Q + Obj_{x,y}$ [No prime]	65.9	63.9	64.2
$Q + Obj_{x,y}$ [No order]	77.8	75.5	75.6
$Q + Obj_{x,y}$ [Non–prime zeros]	80.4	77.2	77.2
$Q + Obj_{x,y}$ [Non–prime rand]	80.6	78.4	78.2
$Q + Obj_{x,y}$ [Order index]	80.2	76.4	76.1
$Q + Obj_{x,y,z}$	82.5	79.2	**79.0**
$Q + Spat + Cat + Obj_{x,y,z}$	77.3	76.4	76.2

Table 4 Accuracy results (in %) for different Dot models (refer Section 3.3) on the Oracle task of GuessWhat. Training, Validation and Test refer to the accuracy scores obtained on the respective splits of the GuessWhat dataset.

Architecture **(Dot)**	Training	Validation	Test
$Q + Obj_{x,y}$	83.4	79.6	79.5
$Q + Obj_{x,y}$ [Prime–embed matrix]	83.3	79.9	79.6
$Q + Obj_{x,y}$ [Prime–embed biLSTM]	83.9	79.9	**79.6**
$Q + Obj_{x,y}$ [No prime]	67.9	64.2	64.2
$Q + Obj_{x,y}$ [No order]	78.1	75.9	76.1
$Q + Obj_{x,y}$ [Non–prime rand]	82.4	79.2	79.1
$Q + Obj_{x,y}$ [Order index]	82.5	79.5	79.5
$Q + Obj_{x,y,z}$	83.8	80.3	80.2
$Q + Obj_{x,y,z}$ [Prime–embed matrix]	83.7	80.6	80.4
$Q + Obj_{x,y,z}$ [Prime–embed biLSTM]	83.3	80.6	**80.5** (Observed) (Best)

Table 5 Accuracy results (in %) for different Sequential models (refer Section 3.3) on the Oracle task of GuessWhat. Training, Validation and Test refer to the accuracy scores obtained on the respective splits of the GuessWhat dataset.

Architecture **(Sequential)**	Training	Validation	Test
$Obj_{x,y} \hookrightarrow Q$	82.4	79.1	**79.2**
$Q \hookrightarrow Obj_{x,y}$	81.7	79.0	79.0
$Obj_{x,y,z} \hookrightarrow Q$ [Prime–embed biLSTM]	82.7	80.1	79.7
$Q \hookrightarrow Obj_{x,y,z}$ [Prime–embed biLSTM]	83.6	80.3	**80.2**

The second block in Table 3 displays results for the ablation experiments on Object Sequences for concatenation models. We discuss the other models with respect to the base model of this block: $\boldsymbol{Q + Obj_{x,y}}$. This model incorporates the information about the prime object using a 0/1 bit appended to object's categorical word vectors in the object sequences. On the other hand, **[Prime–embed]** markers incorporate this information using other methods (using two different embedding matrices for prime and non–prime object, and the method of using a BiLSTM). This marker moves the prime object information from feature level to the architecture level in the models. Models using these methods perform slightly better than the $\boldsymbol{Q + Obj_{x,y}}$ model. Both **[Prime–embed matrix]** and **[Prime–embed biLSTM]** achieve an almost similar accuracy. This shows that both these methods, of incorporating the prime object information, are equivalently good and better than appending a single bit. Removing the information about the prime object from Object Sequences results in a drastic drop of 14.4% in the test set accuracy

as shown in **[No prime]**. This confirms that the knowledge of prime object is very important. Next we see that randomly shuffling the objects in the Object Sequences, results in a minor drop of 3.0% in accuracy as shown in **[No order]**. This result is comparable to the drop in accuracy from $\boldsymbol{Q + Spat + Cat}$ to $\boldsymbol{Q + Cat}$ in the first block. We again infer from this, that the spatial information about the objects is secondary to the performance on the Oracle task.

[Non–prime] markers show a negligible drop in accuracy (1.4% and 0.4% in case of **zeros** and **rand** respectively). Models with **[Non–prime]** markers retain both the spatial and the categorical information about the prime object. They just lose the categorical information about the non–prime objects. This implies that the models (embedding methods and the neural networks), we used, are unable to learn to make an effective use of the extra information about the category of these non–prime objects. Neither are the models powerful enough to make use of the extra information provided by **[Order index]** marker.

$\boldsymbol{Q + Obj_{x,y,z}}$ model in the third block of Table 3 achieves the best performance in this table, achieving an improvement of 0.4% in accuracy over the base model of second block that lacks Obj_z. This shows that depth information (or equivalently information about object sizes) is useful in answering questions about them.

Table 4 shows the results for similar ablation experiments on Object Sequences for dot models. All the markers behave similarly in this table as they did for concatenation models. Thus, this table re–confirms our earlier findings about the importance of different parts of the Object Sequences. Though, in this table we see that **[Order index]** performs comparably to the base model $\boldsymbol{Q + Obj_{x,y,z}}$ **[Dot]**. We also observe that the Dot models outperform their corresponding Concatenation models. This leads us to conclude that Dot embedding is definitely better than Concatenation embedding, possibly due to its non–linear nature.

Table 5 shows the results of applying **[Prime–embed biLSTM]** to both the sequential models. We find that using a BiLSTM to encode prime objects improves upon the accuracy of the respective base models, more so in $\boldsymbol{Q \hookrightarrow Obj_{x,y,z}}$ model.

Finally, we see that our best performing model $\boldsymbol{Q + Obj_{x,y,z}}$ **[Dot] [Prime–embed biLSTM]** obtained an accuracy of 80.5% on the test set, as compared to the baseline $\boldsymbol{Q + Spat + Cat}$ **[Concatenation]**, for the Oracle model in [8], which achieved 78.0% on the test set.

4.3.3 Error Analysis: In this section, we make a qualitative study of our best performing model ($\boldsymbol{Q + Obj_{x,y,z}}$ **[Dot] [Prime–embed biLSTM]**) and the baseline ($\boldsymbol{Q + Spat + Cat}$ **[Concatenation]**) on the Oracle task of GuessWhat using the validation split.

The validation split consists of 113,748 instances. We divide these instances into eight classes based on:

`(actual answer, model prediction, baseline prediction).`

For example the `(N, Y, N)` class would consist of questions for which the correct answer is `No`, our model predicts `Yes` and the baseline model predicts `No`. We then sample hundred examples each from these eight classes at random and manually categorize each of these examples into one of three types:

Object: Questions that should be easily answered using just the prime object's category.

Property: Questions that would require image properties and would be difficult for our model to answer; like "Is it facing left?".

Location: Questions that require spatial reasoning and its association with actions between different objects. We expect our model to perform well on such questions, given the structure of object sequences.

Table 6 Explanation: Left portion of the table shows the count of the eight classes. The columns 'Answer', 'Model' and 'Baseline' represent the actual answers and model predictions respectively (with N–No and Y–Yes). This portion also shows the division of hundred samples of each class into *Object*, *Property* and *Location*. The right portion of the table gives some question instances for each of the eight classes. 'Object Sequences' column gives just the relevant object sequences for that problem (although the model uses all three object sequences along X, Y and Z axes).

As shown in the left portion of Table 6, the category of questions being solved correctly by both the baseline and our model predominantly consist of *Object* type questions, whereas questions that have been incorrectly answered by both the models consist a major portion of *Property* type questions. This shows that the object's visual properties constitute important features that can be included in object sequences to improve their performance. Questions, that our proposed model answers correctly and the baseline answers incorrectly, have a higher number of *Location* type questions. Thus, object sequences are able to utilize the spatial information from the images. Cases, where the "baseline is able to answer *Location* and *Property* type questions" or "our model is able to answer *Property* type questions", indicate dataset biases. In such questions, the models exploit these biases instead of using visual information from the image.

Right portion of Table 6 and Figure 6 show illustrative examples from each of the eight classes described above. From these examples, we again see that our model can benefit by having access to object's visual properties and the background of the image. Other deficiencies of the model represent deeper problems: like acquiring common world knowledge, linking position in object sequences to actions between objects or reducing dataset bias (i.e. using pure visual information to answer questions). One way to tackle this is to use larger datasets, clever splitting of the dataset [19] and employ models that transfer knowledge from multiple domains into this task (similar to using GloVe vectors [24] or ImageNet [28] trained ResNet features [1]). Other ways can be to incorporate attention mechanisms or use dynamic memory networks [29] that allow models to revisit their states and thereby reason better.

4.3.4 Further Remarks: Object Sequences, in a manner, search over the list of objects for the most relevant information using an LSTM, either with the question information (as in Sequential$_{Q \hookrightarrow Obj_{x,y}}$ model) or without it. This search over objects can be made more efficient and task focused by using attention mechanisms.

There are multiple ways to use attention mechanisms with object sequences. One way is to use attention while extracting object features, such as bottom–up attention driven feature extractors [16]. Another way to use attention is just before applying the LSTM on object sequences: such as to replace the attention weighted sum over object features in VQA model [16] with the object sequence LSTM operating on attention weighted object features. Since our approach allows an image to be represented as a sequence, attention mechanism can also be applied just after the LSTM has processed object sequences. An example of this approach is using content–based neural attention [18] on the object sequence, after it has been processed by a BiLSTM, during the decoding stage of an image captioning problem or while modeling the Questioner of GuessWhat.

The current implementation of Object Sequences focuses primarily on the spatial correlation between the different objects, and ignores the visual properties of the individual objects themselves. This problem may be alleviated by incorporating the FC features extracted by the RCNN for individual objects' bounding boxes. Also, object sequences are inherently limited by the effectiveness of the object detector used to create them. Overall, we find that Object Sequences work well in obtaining a representation of the visual (categorical and spatial) information about different objects present in the image, as shown in Section 4.3.3.

5 Conclusions

Visual Question Answering is a holistic task that combines two different modalities. For a system to be able to give competitive performance on this task, it needs to effectively encode and utilize both the visual and language based information. In this paper we proposed Object Sequences, a simple and effective approach for encoding the visual (categorical and spatial) information about all the objects present in an image. This encoding, in form of a sequence, can be readily fed into a neural network architecture. We then considered

Table 6 Examples of performance of the baseline model and our best model on Oracle task of GuessWhat. The 'Type' column refers to categorization of questions into *Object*, *Property* or *Location*. In the 'Object Sequences' column, bold object represents the prime object of that sequence. (refer **Table Explanation** 4.3.3.)

Answer Model Baseline	Count % (113748)	Object Property Location	EXAMPLES S. #	Question	Object Sequences	Figure	Type	Remarks
N Y Y	4.8	19 47 34	1.	Is it car?	X: Truck **Truck** Truck	–	O	Truck and Car are semantically close.
			2.	Is the horse eating?	X: **Horse** Horse Horse	6a	P	Needs more visual information to answer this.
			3.	Person using it?	X: **Kite** Person Snowboard Y: **Kite** Person Snowboard	6b	L	Person and Kite are close in ObjSeq.
Y N N	6.8	12 53 35	4.	Is it attached to wall?	X: **Toilet** Cup Sink Bed	6c	L	Model lacks background information.
			5.	Is it food?	X: **Hot Dog** Hot Dog Hot Dog	–	O	Lacking common sense knowledge.
N Y N	4.2	8 60 32	6.	Snowboard?	X: **Kite** Person Snowboard	6b	O	Clear misclassification
			7.	Is it facing left?	X: C C C C C C **C**	–	P	'C' is for cow. Baseline is coincidentally correct.
			8.	Is it on right side of picture?	X: Chair Clock Sink P **Scissors** P P	6d	L	'P' is for person. Scissors lie on right in ObjSeq.
Y N Y	3.6	15 45 40	9.	Is he near to the car?	X: P P P P P Car P Remote Car **P**	–	L	'P' is for person. Clear misclassification.
			10.	Is object a computer monitor?	X: TV TV Laptop K M Cup K M **TV**	–	O	'K' is for keyboard and 'M' is for mouse.
N N Y	4.6	17 34 49	11.	Is the person on skateboard?	X: P S P **P** S S P S P P P Y: P P **P** P P P P S S S S	6e	L	'P' is for person and 'S' for skateboard.
			12.	Is it the rightmost animal?	X: Cow **Cow** Cow	6f	L	–
Y Y N	5.6	24 32 44	13.	On the surfboard?	X: Surfboard **Person** Person Y: Person **Person** Surfboard	–	L	–
			14.	Is it a short piece?	Z: **C** C C C C C C Fork Bowl	6g	L	Prime object comes first in $ObjSeq_z$.
			15.	Is it black?	X: Person **Chair** Chair	6h	P	Model is coincidentally correct.
			16.	2nd from left?	X: E **E** E E E E E E E E E Car Truck	6i	L	'E' stands for elephant.
N N N	39.9	60 23 17	17.	Is it the whole bus?	X: **Car** Bus Person Person	–	O	–
			18.	Is it edible?	X: S **Dining Table** S Cup Cup	–	O	'S' stands for sandwich.
Y Y Y	30.5	66 16 18	19.	On the left?	X: **Car** Clock Car Car Car	–	L	–
			20.	Is it made of wood?	X: Banana **Dining Table** Banana	–	O	Correctly acquired common sense knowledge.

various techniques for obtaining a joint visual–language embedding, out of which we found that the embedding formed by taking the element–wise product of visual and textual features gave the best results. We also conducted detailed experiments to understand the relative importance of the constituent features used to form the object sequences, on the Oracle task of the GuessWhat dataset. For solving the Oracle task, we observed that the prime object information was of highest importance, followed by spatial location of the prime object. The visual information about the other objects was of least importance. Finally, we benchmarked the performance of the Object Sequence based neural network architecture against the baseline for the Oracle model. Our best model outperformed the baseline.

6 Acknowledgment

We would like to thank the anonymous reviewers and the Editor for their insightful comments.

7 References

1 He, K., Zhang, X., Ren, S., Sun, J. 'Deep residual learning for image recognition.' In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016 (2016) pp. 770–778

2 Ren, S., He, K., Girshick, R.B., Sun, J. 'Faster R-CNN: towards real-time object detection with region proposal networks.' IEEE Trans. Pattern Anal. Mach. Intell., 2017. 39(6), pp. 1137–1149

3 Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Rodríguez, J.G. 'A review on deep learning techniques applied to semantic segmentation.' CoRR, 2017. abs/1704.06857

4 Plummer, B.A., Wang, L., Cervantes, C.M., Caicedo, J.C., Hockenmaier, J., Lazebnik, S. 'Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models.' International Journal of Computer Vision, 2017. 123(1), pp. 74–93

5 Lin, T., Maire, M., Belongie, S.J., *et al.* 'Microsoft COCO: common objects in context.' In Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V (2014) pp. 740–755

6 Karpathy, A., Fei-Fei, L. 'Deep visual-semantic alignments for generating image descriptions.' IEEE Trans. Pattern Anal. Mach. Intell., 2017. 39(4), pp. 664–676

7 Malinowski, M., Fritz, M. 'A multi-world approach to question answering about real-world scenes based on uncertain input.' In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada (2014) pp. 1682–1690

8 de Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., Courville, A.C. 'Guesswhat?! visual object discovery through multi-modal dialogue.' In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (2017) pp. 4466–4475

9 Malinowski, M., Fritz, M. 'Towards a visual turing challenge.' CoRR, 2014. abs/1410.8027

10 Silberman, N., Hoiem, D., Kohli, P., Fergus, R. 'Indoor segmentation and support inference from RGBD images.' In Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V (2012) pp. 746–760

11 Malinowski, M., Rohrbach, M., Fritz, M. 'Ask your neurons: A deep learning approach to visual question answering.' International Journal of Computer Vision, 2017. 125(1-3), pp. 110–135

12 Hochreiter, S., Schmidhuber, J. 'LSTM can solve hard long time lag problems.' In Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996 (1996) pp. 473–479

13 Agrawal, A., Lu, J., Antol, S., *et al.* 'VQA: visual question answering - www.visualqa.org.' International Journal of Computer Vision, 2017. 123(1), pp. 4–31

14 Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., Parikh, D. 'Making the V in VQA matter: Elevating the role of image understanding in visual question answering.' In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (2017) pp. 6325–6334

15 Teney, D., Anderson, P., He, X., van den Hengel, A. 'Tips and tricks for visual question answering: Learnings from the 2017 challenge.' CoRR, 2017. abs/1708.02711

16 Anderson, P., He, X., Buehler, C., *et al.* 'Bottom-up and top-down attention for image captioning and VQA.' CoRR, 2017. abs/1707.07998

17 Xu, K., Ba, J., Kiros, R., *et al.* 'Show, attend and tell: Neural image caption generation with visual attention.' In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (2015) pp. 2048–2057

18 Bahdanau, D., Cho, K., Bengio, Y. 'Neural machine translation by jointly learning to align and translate.' CoRR, 2014. abs/1409.0473

19 Agrawal, A., Batra, D., Parikh, D., Kembhavi, A. 'Don't just assume; look and answer: Overcoming priors for visual question answering.' In SUNw: Scene Understanding Workshop, CVPR – 2017 (2017)

20 Krishna, R., Zhu, Y., Groth, O., *et al.* 'Visual genome: Connecting language and vision using crowdsourced dense image annotations.' International Journal of Computer Vision, 2017. 123(1), pp. 32–73

Fig. 6: Images corresponding to some of the examples given in Table 6. The green bounding boxes mark the prime objects.

21 de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., Courville, A.C. 'Modulating early visual processing by language.' In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA (2017) pp. 6597–6607

22 Strub, F., de Vries, H., Mary, J., Piot, B., Courville, A.C., Pietquin, O. 'End-to-end optimization of goal-driven and visually grounded dialogue systems.' In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017 (2017) pp. 2765–2771

23 Simonyan, K., Zisserman, A. 'Very deep convolutional networks for large-scale image recognition.' CoRR, 2014. abs/1409.1556

24 Pennington, J., Socher, R., Manning, C.D. 'Glove: Global vectors for word representation.' In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL (2014) pp. 1532–1543

25 Schuster, M., Paliwal, K.K. 'Bidirectional recurrent neural networks.' IEEE Trans. Signal Processing, 1997. 45(11), pp. 2673–2681

26 Kingma, D.P., Ba, J. 'Adam: A method for stochastic optimization.' CoRR, 2014. abs/1412.6980

27 Paszke, A., Gross, S., Chintala, S., *et al.* 'Automatic differentiation in pytorch.' In Autodiff Workshop, NIPS (2017)

28 Russakovsky, O., Deng, J., Su, H., *et al.* 'Imagenet large scale visual recognition challenge.' International Journal of Computer Vision, 2015. 115(3), pp. 211–252

29 Xiong, C., Merity, S., Socher, R. 'Dynamic memory networks for visual and textual question answering.' In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (2016) pp. 2397–2406