What is in that picture ? Visual Question Answering System

Juanita Ordonez
Department of Computer Science
Stanford University
ordonez2@stanford.edu

1 Introduction

Visual Question Answering is a complex task which aims at answering a question about an image. This task requires multimodal representation learning for both images and text. To solve this problem, an image and text representation is required and high level interactions between the two must be carefully encoded into the model in order to provide the correct answer. The answers can be a word, a phrase, a yes/no answer, selection from multiple choice or fill in the blank [6]. For this project, I decided to focus on building a model to answer open-ended questions. Answers to open-ended questions are specially challenging because it requires vast set of AI capabilities to answer. Some examples of the capabilities needed are object detection, activity recognition, knowledge base reasoning, and commonsense reasoning[2].

There are many potential applications for Visual Question Answering systems. The most immediate is as an aid to blind and visually impaired individuals, enabling them to get information about images both on the web and in the real world[9]. Other applications are enabling analysts to sift through large quantities of surveillance data and human computer interaction[6].

We introduce VQA system that inputs a natural language question and real world image pair. Using a Neural Network architecture, it outputs the answer to the input question. This work is organized as follows. First, related work is discussed in Section 2. Then, the dataset used described in Section 3 and the proposed approach and in Section 4. The experiments and results are described in Section 5. Finally, we conclude with next steps in Section 6.

2 Related Work

Previous work introduced baselines where image and/or question features were fed into either a logistic regression classifier or a multi-layer perceptron [8]. Another proposed baseline, named IBOWIMG, first extracts image features using a pre-trained convolutional neural network and uses bag of words for the textual question features. The image and textual features are concatenated and a softmax regression is performed across the answer classes [15]. The main limitation of these works is that representing the question as a bag of words doesn't take into consideration the ordering of the words, only their occurrence.

A currently used approach, uses pre-trained neural networks such as ResNet [7] and VGG [12], pre-trained on the ImageNet corpus, to extract the visual information of the input image. This image feature extraction method, has proven to achieve state of the art results in many systems, including VQA [2, 3, 15].

Other work uses recurrent neural network modes such as LSTM [14] and GRU [5] to encode the questions into a vector representation instead of summing over all word vectors. These methods have proven to out-perform the bag of word approaches [2, 3, 15].

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Figure 1: Examples from the VQA dataset. Each image contains three questions and each question has 10 candidates answers along with confidence rate.

Bilinear models are powerful approaches for the fusion problem in VQA because they encode full second-order interactions. They currently hold state-of-the-art performances. The main issue with these bilinear models is related to the number of parameters, which quickly becomes intractable with respect to each input and output dimension.[3]. Other work has been successful on representing full bilinear interactions while maintaining the size of the model tractable, where they achieve VQA score of 67.36 on the open-end task overall questions.

3 Dataset

We perform all of our experiments on the VQA dataset, which is one of the most widely used datasets in Visual Question and Answering research community [2]. This dataset has two sub-parts, one contains real-world images from MS-COCO and the other contains abstract clipart scenes. For this project, we will be using the real-world images and discarding the abstract clipart scenes.

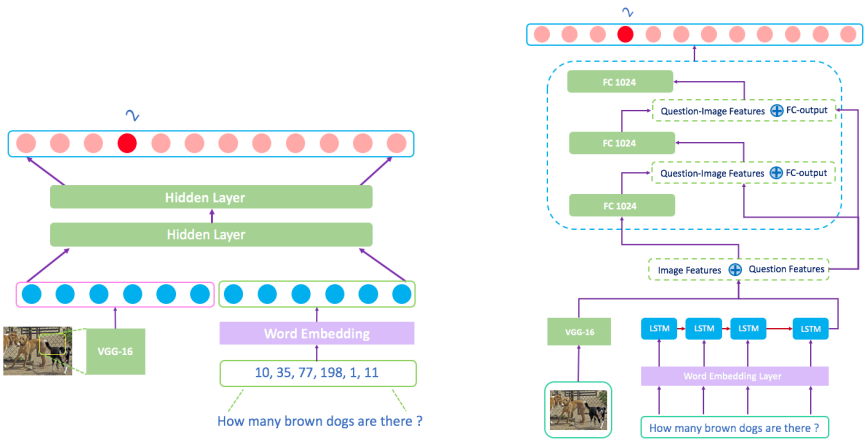
Besides images, the dataset also contains questions and answers generated via crowd-sourcing where each question has ten answer candidates. A typical answer is either a single word or a short phrase with around 40% of the answers being “yes” or “no”. The dataset contains different type of questions, including yes/no, counting and others. In general the VQA dataset contains 760,000 questions, 10 million answers, 123,287 training and validation images and 81,434 test images from Microsoft Common Objects in Context (MS COCO) [2]. For the purpose of this project, we will be using the training and validation splits because VQA doesn’t provide the answers for the test set questions due to the fact that they hold a yearly competition. To account for this we split validation into val-test and val-dev, 70% and 30% respectively.

3.1 Dataset Analysis

The VQA dataset is specially challenging due to the fact that the output labels are not discrete values. The correct answer can also be fuzzy at times, for example, Figure 1 shows the image of a group of bananas and the question is “How many green bananas?". The candidate responses vary by a wide margin, there is not really one correct answer, instead there is an approximation of the correct answer. Another example on the same figure is the man doing a hand stand. If we look at the question: “Do you wear different colored socks sometimes" we clearly see that it is not about the image. These types of questions and answers make it very difficult for a model to learn all the nuances that are involved in picking the correct answer.

4 Approach

The following methods were implemented in python using the Tensorflow framework [1]. The input is an image and question features pair, and the output is the maximum softmax scores over top 1000 most frequent answers in the training set. These top answers cover about 80% of the answers in the entire VQA dataset [2]. Questions that did not have answers on the top 1000 were discarded.



(a) MLP: baseline (b) LSTM-RMLP

Figure 2: High level representation of the MLP and LSTM-RMLP architecture

4.1 Image Features Extraction

In order to extract features from the image the VGG-16 convolutional neural network architecture was used. The network takes as input a 224x224x3 dimensional image and outputs a 4096 dimensional vector (at the fc7 layer) representative of the visual features of the image. The VQA dataset contains images with a wide range of sizes, due to this the image was rescaled to match the VGG-16 input dimension requirements. The VGG-16 network was pre-trained on the ImageNet dataset prior to its use in this work.

The function of this network can be interpreted as summarizing the most salient features of each image prior to feeding it to the models. The intuition behind this feature extraction choice is that part of the task of a VQA system is to be able to recognize objects within an image, given that VGG-16 was trained on this same task they make for very good features.

4.2 Text Features Extraction

Prior to extracting the textual features, punctuation was removed and every word was converted to lowercase. The text was also tokenized using the Natural Language Toolkit (NLTK) [4] and the vocabulary was built only using the training set which left us with 13,758 words. We add to the vocabulary the <UNK> and <PAD> tokens where one the former represents uncommon words in our vocabulary and the latter is our padding token. Questions were limited to a total of 25 words, if the question was longer then we truncated, otherwise we simply add the <PAD> token.

Prior to inputting these words into our model we used an embedding layer to convert each of our words $(x_1, x_2, x_3, ..., x_n)$ to a vector representation $(w_1, w_2, w_3, ..., w_n)$. Our models learn these word embeddings during training, this allows them to learn the difference and relationship between words in our vocabulary.

4.3 Baseline: Multi-Layer Perceptron

The multi-layer perceptron contains a total of two hidden layers each followed by a tanh action function. As input, the network maps every word from the question to a 300-dimensional embedding vector. All of these vectors are then summed up to obtain a bag of words feature representation. We then concatenate the image features which amounts to $4096 + 300 = 4396$ total number of features. Finally, the logits are passed to a softmax layer over the top 1000 possible answers where cross entropy is performed.

4.4 Improvement: LSTM-RMLP

In order to handle variable-sized question inputs and not throw out word order information, we explore the LSTM recurrent neural network architecture. The entire input question sequence $(x_1, x_2, ...x_n)$ is first encoded, summarizing the question into one hidden state vector which is then concatenated with the features extracted from the image.

For some input x_t at timestep t, the LSTM computes a hidden state h_t, a memory cell c_t and the next output o_t. The equations describing the LSTM are:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

Where σ is the sigmoid non-linearity, $\odot$ is an element wise product and the weight matrices W and biases b are learnable parameters. During training the question encoder network takes as input word sequences $(x_1, x_2, ...x_n)$ and computes some sequence of hidden states $(h_1, h_2, ..., h_n)$ and cell states $(c_1, c_2, ..., c_n)$.

We take the last hidden state, and concatenate the image features, we call this the question-image vector, denoted as v_{qi}. Then we recursively encode the question and image features using multi-layer perceptron.

$$\begin{aligned} h^w &= tanh(W_w(v_{qi}^w))) \\ h^p &= tanh(W_p[(v_{qi}^p), h^w]) \\ h^s &= tanh(W_s[(v_{qi}^s), h^p]) \\ p &= softmax(W_h h^s) \end{aligned} \tag{2}$$

where W_w, W_p, W_s and W_h are the weight parameters.[.] is the concatenation operation on two vectors , p is the probability of the final answer.

5 Experiments and Results

The model was trained using the Adam optimizer [10] with a learning rate of $1e-3$. We tested the models in different sizes of batches, but in our experiments we found that the batch size of 512 worked the best. The model LSTM-RMLP is a relatively more complex structure compare to our baseline but we found that it tends to overfit. To avoid this we introduced a dropout [13] layer in between the MLP layers, we found that a dropout rate we found that .5 worked the best.

5.1 Evaluation Metric

To evaluate the generated answers from both models, we used the VQA metric scores against all candidate answers.

This score is computed based on how many times the answers appears on the candidate list. If it appears 3 or more times, then it is given a score of 1, otherwise, it is given less score. For this reason, even though the model treats everything like classification, in reality the task does not have discrete outputs. The following is the equation describing the VQA metric:

$$VQA_{score}(a) = \min(\frac{a}{3}, 1) \tag{3}$$

Where a represents the number of matches to the candidate answers. [2]

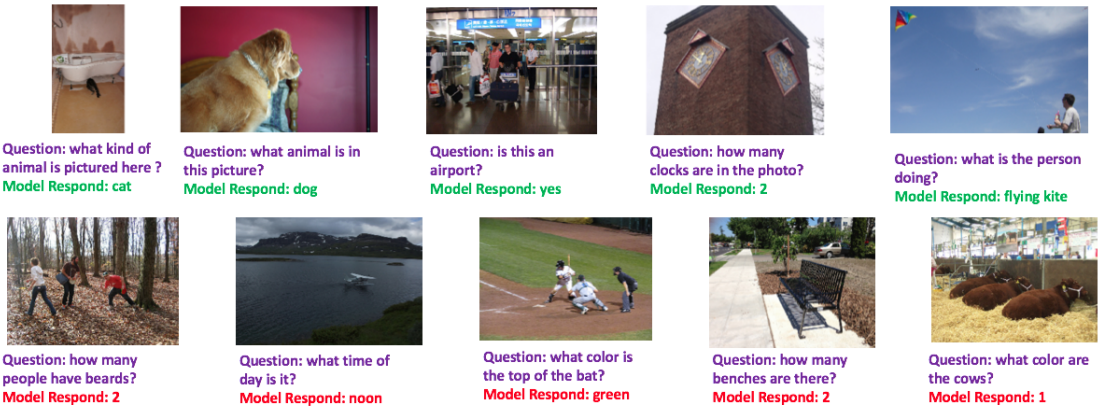


Figure 3: LSTM-RMLP image and question inputs and model's ouput, on the val-test dataset. Green indicates that the answer generated is correct and red vice versa

5.2 Experiment Results

The following table shows a break down of our results evaluated on the val-test dataset. Each model was trained for a total of 50 epochs with the same hyper-parameters. We show evaluations for the following models: MLP baseline, Recursive MLP with bag of words and LSTM-RMLP. We also show the results of a language-only LSTM-RMLP model wherein no image information is used.

Table 1: Results on my val-test dataset

	all	other	count	yes/no	train all	val-dev all
MLP	48.02	36.67	32.68	63.16	48.55	48.05
RMLP	49.32	36.67	**32.68**	63.14	71.64	49.09
LSTM-RMLP	**51.89**	**41.05**	32.52	**67.76**	78.68	51.8
Language Only	47.67	31.36	32.72	67.22	47.83	47.67

Our results show that encoding the question using an LSTM, as we do in the LSTM-RMLP module, our VQA scores went up by 3.87%. However, the RMLP model caused us to overfit after a certain point. The language-only model only did around 4% worse in comparison to the full information LSTM-RMLP. This result is extremely surprising as it means that the model does quite well in answering questions about an image without ever seeing it. The task of counting objects from an image is the most difficult accross all models, there are many factors in play, one is that the counting might not be exact as the people who built the dataset tend to give a ballpark estimate instead of an exact one.

We found the qualitative results to be quite interesting, even when the model gets a VQA score of zero we see that it is pretty close to its target. For example in figure 5.1 with the question "what color is the top of the bat?" it answered green but we could see that it got confused by the color of grass. The model failed to identify that the word “bat” corresponds to the “bat” within the image.

6 Conclusion

In this work, we started by building a simple baseline based on a two-layered MLP. We then extended this model to include an LSTM that encodes the questions and a recursive mlp (RMLP) module that processes the inputs. We found that concatenating the image and word features and keeping them accross the whole network was very beneficial to the performance of our network. The performance of the language-only model also indicates that we need better image features to really gain a performance boost. Current state of the art models extract image features at the last pooling layer and then apply attention over the feature map [11], we intend to extend our work in this fashion.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015. URL `http://arxiv.org/abs/1505.00468`.

[3] H. Ben-younes, R. Cadene, M. Cord, and N. Thome. Mutan: Multimodal tucker fusion for visual question answering. *arXiv preprint arXiv:1705.06676*, 2017.

[4] S. Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

[6] A. K. Gupta. Survey of visual question answering: Datasets and techniques. *CoRR*, abs/1705.03865, 2017. URL `http://arxiv.org/abs/1705.03865`.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] K. Kafle and C. Kanan. Answer-type prediction for visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] K. Kafle and C. Kanan. Visual question answering: Datasets, algorithms, and future challenges. *CoRR*, abs/1610.01465, 2016. URL `http://arxiv.org/abs/1610.01465`.

[10] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[14] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[15] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. Simple baseline for visual question answering. *CoRR*, abs/1512.02167, 2015. URL `http://arxiv.org/abs/1512.02167`.