Stacked Attention for Visual Question Answering

Bingbin Liu
Department of Computer Science
Stanford University
bingbin@stanford.edu

Weini Yu
Department of Computer Science
Stanford University
weiniyu@stanford.edu

Abstract

Our project explores Visual Question Answering on multiple-choice questions given an image. Based on recent works using different mechanisms, we first build a Bag-of-Words model with MLP [1, 2] using GloVe word embeddings [3] and ResNet image features [4], which outperforms the original LSTM with attention model [5]. We further extend it by replacing our language model with LSTM and add stacked spatial attention layers following [6] to capture the interaction between the words and image regions. We investigate different aspects of the VQA task on the Visual7W dataset [5] by experimenting with many different settings and obtain interesting results. Finally, we present analysis on which options contribute to better results.

1 Introduction

By combining visual and language understanding, two of the most important input modalities in artificial intelligence, **visual question answering (VQA)** has sparked wide interests in research community since the term was officially coined [7]. Under the VQA setting, a model should be able to answer a natural language query about an image. Different from object recognition, the free-form query requires natural language understanding beyond classification with discrete labels. Different from text-based question answering, VQA further requires the model to find semantic links between the textual description and the image, thus complicating the learning task.

In this project, we are interested in learning what is important to a VQA model. Besides trying to train a good performance model on the dataset. We are interested in understanding where the model gets information for predicting the answer, and how visual and textual component correspond to each other. With this goal, we start off with a simple bag-of-words model, and incrementally adding complexities to the model including LSTM and stacked spatial attention, based on inspirations from related works. The paper is organized as follows: section 2 provides a brief overview of recent development in VQA; section 3 will explain our approaches, while section 4 provides implementation details and discussion about results.

2 Related Work

2.1 VQA

Since its introduction in [7], VQA has attracted wide interests in joint understanding of vision and natural language, and various datasets have emerged to support related research [7, 5, 8]. VQA requires understanding of both visual and language input. For the visual component, it is common practice to use a pretrained CNN [9, 10, 4] as feature extractor of the input image [5, 1, 2]. For the natural language query, several works [5, 11] used LSTM [12], which is widely considered as the default model for handling sequence input such as languages [13]. However, [1] presents a very interesting finding that a simple bag-of-words (BOW) model may suffice for some particular types of tasks and is even able to outperform some of the complex LSTM frameworks [11, 7]. We hence chose to start with this simple implementation to set a performance baseline.

2.2 Language Understanding

Understanding the language query is one of the two pillars of VQA, towards which various approaches have been explored. In addition to LSTM [5, 11] and BOW [1, 2], [14, 15] use 1D CNN to extract information from query based on fixed context windows, which provides superior performance under certain settings.

2.3 Attention

Using language input to spatially attend the image has been proved to be effective for VQA tasks [5, 11, 16]. Moreover, multi-step or hierarchical approaches to aggregating spatial attentions have been adopted by works such as [17, 6, 15, 11]. [17] uses a multi-hop attention scheme; [15] follows the word-phrase-sentence hierarchy to obtain attention at different granularity separately; [6] recursively refines the attended representation by stacking attention layers. [11] takes one step further to explore more flexible network configuration by adjusting network layout based on a language parser. In this project, we follow the approach in [6] for its visual interprebility and relatively shorter training time.

3 Approach

3.1 Bag-of-Words

We implemented a baseline model following [1, 2], where the textual input and the image are each represented by a vector. The image is encoded using the layer 4 of ResNet101 [4]. The question is concatenated with each of the four candidate answers to get four set of language inputs; for each set, words in the question and the answer are first embedded using GloVe [3], then aggregated as a fixed-size textual vector representation using averaging bag-of-words (BOW) model. Then, the textual vector is concatenated with the visual encoding, and a three-layer multi-layer perceptron (MLP) will take in the concatenation and produce a score representing the correspondence between the textual vector and the visual encoding. The candidate corresponding with the highest score is output as the prediction.

3.2 LSTM

Starting from the above baseline, we gradually add complexity to the model to produce more sophisticated outputs. The first step is to replace the BOW model with a LSTM [12], which has demonstrated strong abilities in handling language inputs. The rest of the network remains the same as in the BOW, and the system architecture is shown in Figure 1.

3.3 LSTM with Spatial Attention

We further experimented a stacked attention scheme following [6], where the attention gets refined recursively in different passes of the attention layer. The visual representation of the image is now changed from a vector in $v_i \in \mathbb{R}^{2048}$ to a three-dimensional representation in $v_{spatial} \in \mathbb{R}^{2048\times7\times7}$, where 7×7 is the spatial resolution of the visual representation. We first flatten the three-dimensional representation to a matrix $v_I \in \mathbb{R}^{2048\times49}$ where 49 is the number of spatial locations (i.e. $49 = 7 \times 7$). The text input is then embedded to $\mathbb{R}^{300}$ using a LSTM as defined above, then an embedding layer will embed this 300-dimensional vector to a vector $v_L \in \mathbb{R}^{2048}$ in the common embedding space as the visual encoding as each spatial location. v_L is then used to attend v_I at each location, resulting in a spatial attention vector $p_I \in \mathbb{R}^{49}$. A single vector representation $\tilde{v_I}$ for the image is produced as the weighted sum of the spatially attended visual representation, and is added with the textual vector v_q as the input for the next attention layer.

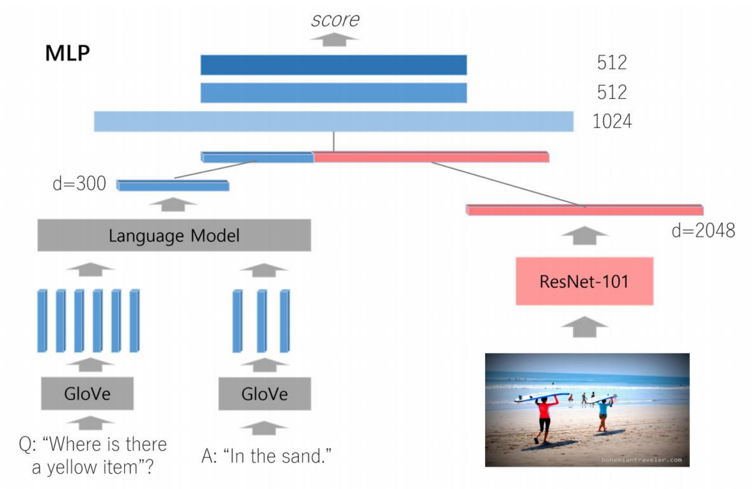

Figure 1: **Architecture that uses MLP to produce correspondence scores** from concatenated visual and textual encoding, where BOW or LSTM is used as language model.

Formally, at the k_{th} attention layer, the network computes:

$$\begin{aligned} h_A &= tanh(W_I^k v_I \bigoplus (W_L^k u_{prev} + b_L)) \\ p_I^k &= softmax(W_P^k h_A + b_P^k) \\ \tilde{v_I^k} &= \sum_i p_i^k v_i \\ u &= \tilde{v_I^k} + u_{prev} \end{aligned} \tag{1}$$

Where $W_I^k, W_L^k \in \mathbb{R}^{d_{hidden} \times 2048}$ map the visual and language encoding to a hidden embedding space of dimension d_{hidden} (a tunable hyperparameter), and $W_P^k \in \mathbb{R}^{1 \times d_{hidden}}$ maps each hidden vector to a score which is then turned into attention probability by softmax. For the first attention layer, u_{prev} is set as v_L the language encoding.

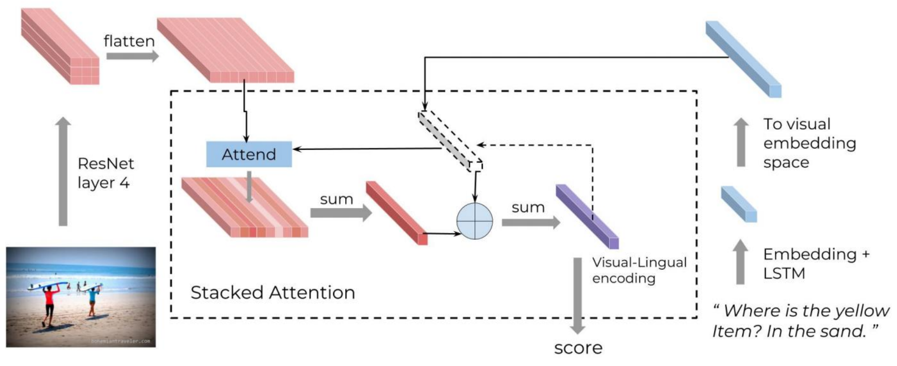

Figure 2: **Architecture for stacked image attention using language guidance**. The text encoding is used to generate an attention over spatial locations of the visual representation, where a visual-lingual encoding is produced, and can then used to calculate the next level of attention. The total number of levels is a hyperparameter we can set. The visual-lingual encoding of the last level will be used to produce the image-textual correspondence score by passing through a multi-layer perceptron, which is omitted in the figure.

4 Experiments

4.1 Dataset

We use Visual7W[5] Telling dataset, which contains 69817 questions for the training set, 28020 questions for validation set, and 42031 questions for test set. Each question is one of the 6 Ws (i.e. *what, where, when, who, why*, and *how*) and is paired with 4 candidate answers, from which the network learns to select the uniquely correct answer.

4.1.1 Preprocessing

For textual information, we use 300-dimensional GloVe[3] word vectors pretrained on 6 billion tokens from Wikipedia 2014 and Gigaword 5 to represent words in the questions and answers, where sentences are tokenized the same way the GloVe was trained. Unseen words are set to zero vectors.

To encode the image, we use ResNet-101[4] pretrained on ImageNet[18] as the feature extractor. For single-vector representation of images, such as the case for the baseline BOW model and the LSTM base model, we take the output of the average-pool layer, a 2048-dimensional vector. For spatial attention, we take the representation before average pooling to preserve spatial information, which is a $2048 \times 7 \times 7$ feature map for each image.

4.2 Evaluation Metrics

We compute the *top-1 accuracy* as the performance metric, which is defined as the percentage of correctly answered questions.

4.3 Results

4.3.1 Overall Results

To train a better performance model, we spend a fair amount of time on hyperparameter search regarding 1) single-directional or bi-directional LSTM, 2) number of layers of LSTM, 3) hidden dimension of LSTM, 4) learning rate, 5) batch size, 6) dropout rate, and 7) value for weight decay (i.e. L2 regularization). Figure 3 shows the hyperparameter search for LSTM hidden dimension, dropout rate and L2 regularization.

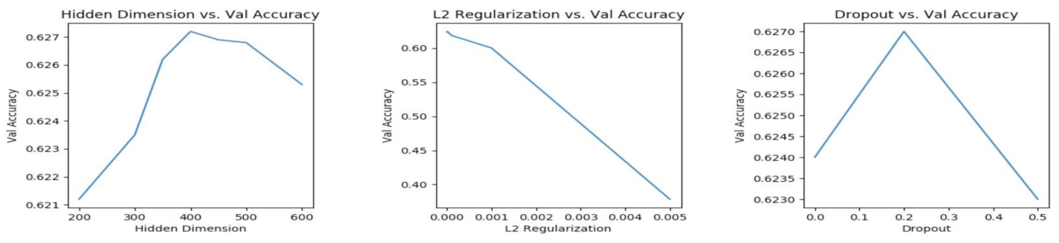

Figure 3: Hyperparameter values vs. validation accuracies

We find that the single-directional LSTM with 1 layer, 400 hidden dimension, a dropout rate of 0.2, weight decay equal to 10^{-5}, batch size 128, and a learning rate of 10^{-6} using an Adam optimizer [19] yields the best results. With the appropriate hyperparameters and feature selections, we report the test set accuracies of our 3 models in Table 1.

Our LSTM base model has the best performance, which is 11.2% higher accuracy than the Visual7W baseline [5]. Despite its simplicity, our BOW model is able to outperform the baseline by 9.5%. Our LSTM with spatial attention is just a little lower than the LSTM base model but very close. Contrary to our expectation, stacked spatial attention does not give superior performance over LSTM, which we will analyze in section 4.3.6.

Table 1: Accuracy for different models across question types, using Question + Answers + Image and trained with BCE loss

Model	What	Where	When	Who	Why	How	Overall
Visual7W baseline [5]	0.515	0.570	0.750	0.595	0.555	0.498	0.556
BOW	**0.585**	0.694	0.797	0.673	0.566	0.511	0.609
LSTM	0.584	**0.719**	**0.803**	**0.684**	**0.593**	**0.526**	**0.618**
LSTM-Att	0.560	0.691	0.800	0.660	0.590	0.512	0.597

4.3.2 Image Feature as Input for LSTM

We also explore the option of using image feature vector as the first token in the input sequence of LSTM, along with other QA word tokens. This approach is proposed in Visual7W[5], but it does not improve the performance of our LSTM model - with an accuracy of 0.599, versus 0.618 without the image feature. We think the reason is since the image feature is already used, together with the last LSTM hidden state, as the inputs to the MLP, having an extra token representing the image at the beginning of the LSTM input sequence does not provide more useful information, and may make the model more prone to overfitting as witnessed by comparing the training and validation accuracy, which explains the reduced performance.

4.3.3 Ablation Study on Features

As an ablation study and to better understand the model, we are interested in learning how much information comes from the visual component. To verify this, we take away the image input, and let the model learn from only the textual input. We then further remove the question, to see how much prior information the candidate answers contain. The results for both BOW and LSTM models are reported in table 2.

Table 2: Accuracy with different features (BCE loss)

Model	Feature	What	Where	When	Who	Why	How	Overall
Visual7W baseline [5]	Q+A+I	0.515	0.570	0.750	0.595	0.555	0.498	0.556
	A	0.456	0.547	0.756	0.611	0.480	0.490	0.507
BOW	Q+A	0.528	0.582	0.776	0.646	0.548	**0.530**	0.562
	Q+A+I	**0.585**	**0.694**	**0.797**	**0.673**	**0.566**	0.511	**0.609**
	A	0.474	0.578	0.767	0.649	0.556	0.489	0.530
LSTM	Q+A	0.528	0.587	0.782	0.648	0.564	**0.531**	0.564
	Q+A+I	**0.584**	**0.719**	**0.803**	**0.684**	**0.593**	0.526	**0.618**

Interestingly, we find that both BOW and LSTM outperform the baseline model in [5] when using language features only, further explaining why adding image as the first input of the LSTM model does not help, which matches our observation of getting reduced performance when adding image to the input of LSTM.

Moreover, the question type ”how” seems to be an outlier for both models. Questions with type ”how” are predicted more accurately without the image information available. There are two observations that may explain this. One is questions asking about the quantity of objects in the image, such as ”how many” like the question on the left in Figure 4. Neither our base models nor the attention model is able to handle this type of counting problem decently. The other type of error is about more abstract concepts involving relatively complex reasoning, which may rely on information beyond visual clues in images, such as the example on the right in Figure 4.

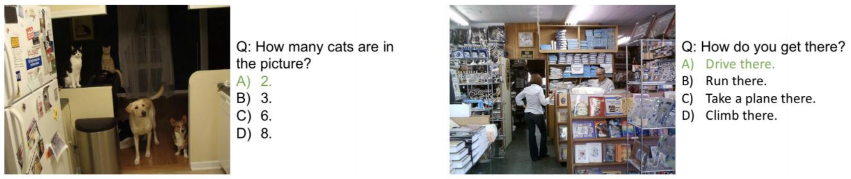

Figure 4: Examples of ”how” questions which the model may fail to answer: counting (*left*) and information outside image (*right*).

4.3.4 Loss

The above experiments are performed using **Binary Cross Entropy** (BCE) Loss. We also experiment with training our BOW and LSTM models using **Ranking Loss**, with the intuition that Ranking Loss may focus more in capturing the relative relationships between answers, which correspond directly to the performance metric used in this task. However, the results show that the model actually performs better with BCE (see Table 3). This may be because the nature of how the question and answers are formed in this dataset – the correct answer does not often have a relative relationship with the incorrect ones. In most cases, the incorrect answers are utterly wrong and thus BCE loss would suit better.

Table 3: Accuracy with different loss (Q+A+I)

Model	Loss	What	Where	When	Who	Why	How	Overall
BOW	BCE	0.585	0.694	0.797	0.673	0.566	0.511	0.609
	Rank	0.515						
LSTM	BCE	**0.584**	**0.719**	**0.803**	**0.684**	**0.593**	0.526	**0.618**
	Rank	0.553	0.666	0.769	0.658	0.576	**0.535**	0.591

4.3.5 Fine-tune Word Embeddings

Fine-tuning word embeddings is a usual practice in lots of NLP tasks. We want to see if it helps improve our model accuracy, so we train our LSTM model with fine-tuning word embeddings on the pretrained GloVe word vectors and without. However, we observed a lower test accuracy of 0.5541 for fine-tuning, compared to 0.618 without tuning.

As seen in Figure 5, the six question words which are highlighted in circles have been moved drastically. Before fine tuning, the question words are close, which makes intuitive sense since they are all question words. After fine-tuning, the six questions words are scattered, which indicates the model is learning to distinguish the question types and possibly gaining information just from the question types. Specifically, for *where* and *who* questions, one can often get a good answer by paying attention to specific objects or regions of the image. For *when* and *why* questions, one may have to look for context in the entire image. *What* and *how* questions are even more involved that require deeper semantic understanding and complex reasoning.

On the other hand, this may also suggest that the word vectors may be moved around too much, which is not desirable considering how small the training set is: there are only around 600k words in the training set, whereas the GloVe vectors were originally trained on over 6 billion tokens. We posit the fact that our training set is not large enough to provide stable update to the word embeddings may be the reason for the lower accuracy.

4.3.6 LSTM with Spatial Attention

In figure 6, we visualize the attention vectors from different layer of the stacked spatial attention network.

As can be seen in the heatmaps, incorrect answer candidates often produce heatmaps that do not make intuitive sense, while heatmaps from correct answers are much more interpretable (e.g. the

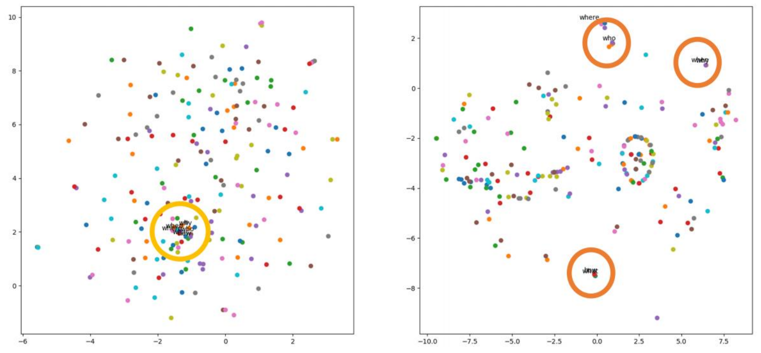
Figure 5: Word vectors for the six question words and 200 randomly sampled words, without (*left*) and with (*right*) fine-tuning. On the figure without fine tuning (*left*), the six question words are closely clustered together, whereas after fine-tuning (*right*) on text inputs in the training set, the question words are separated into 3 clusters, with *where* and *who* close to each other at the top, *when* and *why* on the right, and *what* and *how* at the bottom.

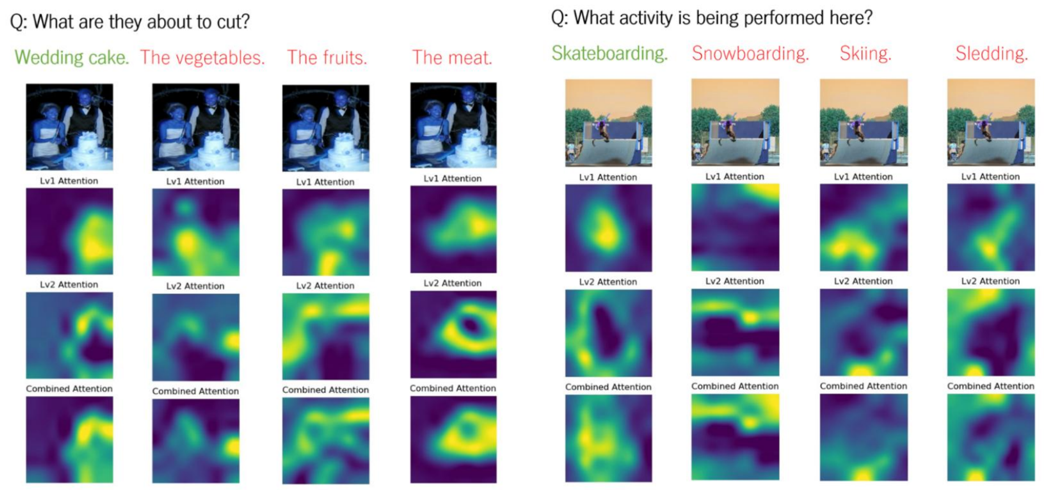

Figure 6: Spatial attention represented as heatmap. The first row are the original images, the second row are heatmaps for attention from the first attention layer, the third row are heatmaps from the second attention layer, and the last row are heatmaps for the combined attention (i.e. sum of first and second layer).

focus on the cake in the image on the left in figure 6, and the focus on the person in the image on the right.) There are also mainly two interesting patterns between heatmaps from different layers. On the one hand, the second layer sometime helps group scattered attentions in the first layer, similar to smoothing. On the other hand, there are examples where the first attention layer only focuses on part of the area of interest, and the second layer shows complementary activation, resulting in a decent combined attention.

However, the overall performance of the stacked spatial attention model is not as good as the LSTM base model. To understand the characteristics of the model, we compare examples where exactly one model gets the correct results. Out of the 38703 examples, LSTM base model outperforms the attention model on 3958 of them, while attention model outperforms LSTM base model on 3150 of them. Since LSTM base model does not offer easy features for visualization, we select some examples where the attention model outperforms the LSTM base model to understand its strength. Based on the examples in figure 7, it seems that the attention model is able to better pick up nuances than the LSTM base model. We do not find obvious error type from the 3958 examples; together with the training and testing performance, we hypothesize that the inferior performance may be the results of lack of model tuning due to limited time, since spatial visual features are around 50 times larger than single vector representation and thus takes much longer to train. Therefore, we expect the stacked attention model to work better with better hyperparameter search, which we leave as future work.

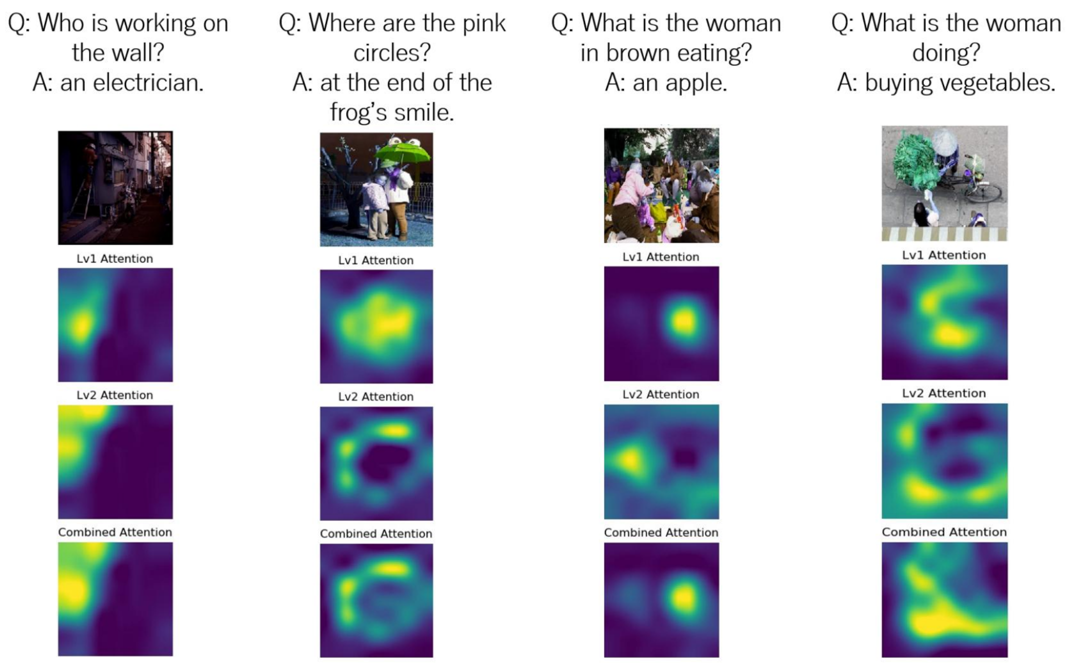
Q: Who is working on the wall?
A: an electrician.
Q: Where are the pink circles?
A: at the end of the frog's smile.
Q: What is the woman in brown eating?
A: an apple.
Q: What is the woman doing?
A: buying vegetables.
Lv1 Attention
Lv2 Attention
Combined Attention

Figure 7: Examples where the stacked attention model outperforms the LSTM base model.

5 Conclusion

In this project, we explored three models, BOW, LSTM and LSTM with spatial attention, by incrementally adding complexities, and are able to achieve significant performance gain compared to the baseline in [5]. We used ablation study and visualization tools to understand the contribution of features and model designs, and present interesting findings during our experiments.

As for future work, we would like to add text attention to our LSTM model, which is proven to be effective in [15]. We would like also like to perform a more thorough hyperparameter search for the stacked attention model, including the number of attention layers. Using finer image feature maps may also help improve spatial attention, for example, use $14 \times 14 \times 2048$ instead of $7 \times 7 \times 2048$.

References

[1] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus, "Simple baseline for visual question answering," *CoRR*, vol. abs/1512.02167, 2015.

[2] A. Jabri, A. Joulin, and L. van der Maaten, "Revisiting visual question answering baselines," *CoRR*, vol. abs/1606.08390, 2016.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[5] Y. Zhu, O. Groth, M. S. Bernstein, and L. Fei-Fei, "Visual7w: Grounded question answering in images," *CoRR*, vol. abs/1511.03416, 2015.

[6] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola, "Stacked attention networks for image question answering," *CoRR*, vol. abs/1511.02274, 2015.

[7] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: visual question answering," *CoRR*, vol. abs/1505.00468, 2015.

[8] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[11] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Deep compositional question answering with neural module networks," *CoRR*, vol. abs/1511.02799, 2015.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[14] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "ABCNN: attention-based convolutional neural network for modeling sentence pairs," *CoRR*, vol. abs/1512.05193, 2015.

[15] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," *CoRR*, vol. abs/1606.00061, 2016.

[16] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," *CoRR*, vol. abs/1603.01417, 2016.

[17] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," *CoRR*, vol. abs/1511.05234, 2015.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[19] J. L. B. Diederik P. Kingma, "Adam: a method for stochastic optimization," 2015. ICLR15.