Visual Question Answering using Convolutional Attention

Siddharth Kannan
siddhark

Vignesh Kannan
vkannan2

1 Introduction

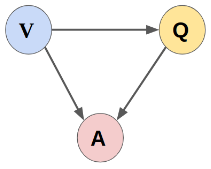
We have chosen the VQA (Visual Question Answering) Challenge (1) as the problem of interest for this project. The goal of the task is to develop a model to answer questions about a given image. The challenge focuses on one-word answers including simple *Yes/No* answers, numerical answers for counting related questions and other answers pertaining to color, size, location or some other characteristic of the image. The model receives an image and a question about the image as input and it must be able to select a single word from the set of all possible answers in the dataset. This essentially makes the task a classification problem. We will be using the VQA v2.0 dataset for the purpose of our project. There are a total of 82,783 training images, 40,504 validation images and 81,434 testing images. There are a total of 443,757 training questions, 214,354 validation questions, and 447,793 testing questions. Each question in the training and validation sets comes with 10 annotations (answers) based on responses from 10 people. This is needed to account for the subjective nature of certain questions. If the system's predicted answer gets a minimum number of required matches among the 10 annotations, then it is considered correct. Going with the metric used by the VQA challenge, we use the following custom accuracy metric in this project.

$$custom\_acc = \min\left(\frac{num\_matches}{3}, 1.0\right) \tag{1}$$

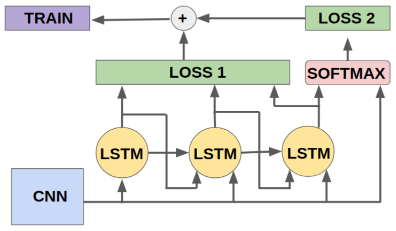
where $num\_matches$ is the number of matches of the predicted answer with the 10 ground-truth annotations. Deviating from our initial proposal, we chose v2.0 instead of v1.0 because of the following reasons. Firstly, though the number of images remain the same, for each image there are more number of associated questions in v2.0 than in v1.0 (nearly double). Secondly, v2.0 reduces language biases present in v1.0 and is more balanced in some sense. The problem with v1.0 is that the distribution of image, question, answer tuples allows models to "memorize" answers to questions without accounting for image information. But v2.0 ensures that for most of the questions, there are at least two similar images which result in different answers for the question thus mitigating this problem. Lastly, the testing server for v1.0 was not available since the challenge has moved on to v2.0.

2 Background

In our midway report we ran our experiments on the v1.0 dataset. We used a CNN+LSTM baseline model (1) which extracts features from the image and question separately and uses element-wise multiplication to merge the two sets of features to perform the answer classification (code borrowed from (2)). The results indicated that the model is able to answer Yes/No questions significantly better than number-based and other questions. We reasoned this out by noting that the chances that a model gets the answer right after narrowing down to the type of question in the case of a Yes/No question is 50% but for other types of questions the number of possible answers is more than 2. This baseline method lacked a coupling between image and question feature extraction in our opinion and to address this, we proposed the VQA dependency graph in Fig. 1a. In accordance with this, we proposed the architecture in Fig 1b wherein given the image, the model learns to generate a question and given the image and question, the model predicts an answer.



(a)



(b)

Figure 1: (a) Dependency graph between images, questions and answers; (b) Proposed model architecture for Visual Question Answering

3 Related Work

Visual question answering has become a hot topic of research due to the promising results produced by Convolutional Neural Networks on image recognition tasks and deep sequential models like LSTM for processing natural language. Many large datasets have been introduced to train these models, the most popular ones being VQA(1), COCO-QA(3) and DAQUAR(4). The basic method adopted by majority of the works is to use deep architectures to extract representations of images and questions and then combine them to classify the correct answer. Zhou et al. (5) perform a simple concatenation of image and text features and feed this to a softmax layer to make a prediction. Another simple technique proposed by (1) is to perform an element-wise multiplication of the image and text representation. Fukui et al. (6) argue that such methods are too simple to capture the complex interactions between features and propose a strategy called Multimodal Compact Bilinear Pooling (MCB) which computes an approximate outer product of the image and text features in a lower dimensional space. They were able to beat the state-of-the-art on VQA at the time.

Another direction of research experiments with various models for extracting image and text features. Image features are predominantly extracted using state-of-the-art CNNs like ResNet and VGGNet. For extracting a question representation, Zhou et al. (5) used bag-of-words features, whereas, (1) used the last time-step of an LSTM as the question representation. Kafle et al. (7) were the first to use skip-thought vectors for visual question answering since these representations are trained in such a way that they have contextual meaning embedded in them. Yet another successful question representation was proposed by Lu et al. (8) wherein they extract textual features at three levels; words, phrases and questions. They then combine the features at all three levels in a recursive manner to make the final answer prediction.

The most successful idea in the context of visual question answering, however, has been the use of attention mechanisms. The argument in favor of using attention models is that there is often few parts of the image and/or few parts of the question which are the most important for answering a question. Yang et al. (9) propose a stacked attention model which they reason out by claiming that the stacking of attention modules enables the model to progressively narrow down to the most important segment of the image. Xu et al. (10) propose an attention model to learn the correlation between image patches and words in the question using a Spatial Memory Network which stores the activations of different regions of the image and takes cues from the question to select which regions of the image to use. In a similar spirit, the model described in (8) leverages the question representation to generate visual attention maps and conversely, the visual feature maps to produce attention values for the question. Using this mechanism they call "Co-attention", they were able to beat the state-of-the-art on VQA dataset by 2%.

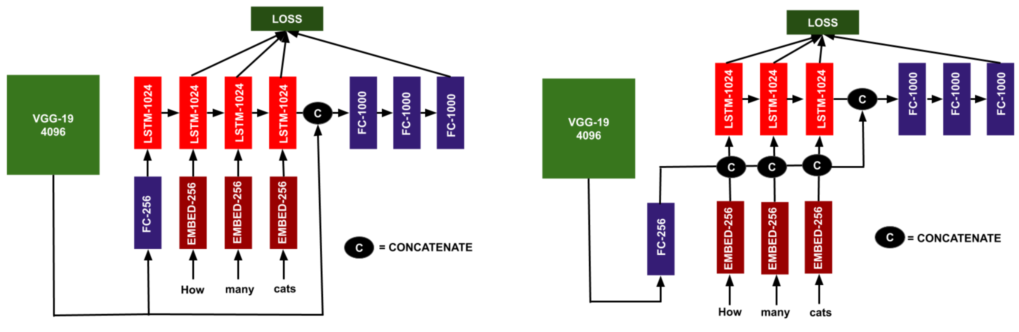
4 Methods

In keeping with the *VQA-dependency-graph* in Fig. 1a, we implemented and experimented 5 different network architectures in this project. Each one progressively builds on the previous one by incorporating potentially better ways of realizing the dependency graph. The 5 architectures are described in detail below.

4.1 Show and Tell

Going in line with our initial proposal, we start by extending the Show and Tell image captioning network proposed by (11). This architecture comprises of a CNN and an LSTM connected sequentially. For each (image, question, answer) tuple, the image is passed as input to the CNN and the output from its last fully-connected layer is retrieved as the embedding for the image. The embedding is fed into the LSTM as the first timestep input. This captures the idea that the distribution of questions depends on the image on which they are based. The image embedding influences the question generation through the hidden state of the LSTM after the first timestep. In subsequent steps, the question is fed through an embedding layer into the LSTM one word at a time. The hidden state produced at each timestep is fed to a fully-connected layer (shared across time) for classification into one of the words in the vocabulary of the dataset. The final hidden state of the LSTM is then concatenated with the image embedding and fed into fully-connected layers for classification into one of the answer categories. The last step essentially allows the architecture to learn a distribution over the answer categories given the question and the corresponding image.

We use a pretrained VGG-19 network to extract a 4096 dimensional representation for the image embedding. This representation is further reduced to 256 dimensions for it to match with the input dimension of the LSTM. Likewise, the embedding layer generates a 256 dimensional representation for each word of the question to be fed into the LSTM. We use a two-layer LSTM each with 512 units. The LSTM is allowed to operate bidirectionally and the last hidden states arising out of each direction are concatenated with the 256 dimensional image embedding to be fed into the fully-connected layers. Our architecture uses 3 fully-connected layers each with 1000 units. The last fully-connected layer uses a softmax function to classify answers. Fig. 2a summarizes the model.



(a) Show and Tell (b) Time Distributed CNN

Figure 2

4.2 Time Distributed CNN

The main focus of this model is to make the image embedding directly accessible to each timestep of the LSTM. This might help improve the quality of question generation and in turn, the predicted answer because each timestep can learn a distribution of words which is not only conditioned on the previous hidden state, but also on the image embedding. Such an input to the LSTM helps ensure that each generated word of the question is relevant to the image on which the question is based. The previous architecture uses the image embedding only in the first timestep to initialize the hidden state for the subsequent unrolling of the LSTM. But a well-known weakness of deep sequential models like RNNs and LSTMs is their inability to learn very long-term dependencies. So with the Show and Tell model, it is possible that the influence of image information fades away as we reach later timesteps. Introducing the image embedding into the LSTM input at each timestep may help retain image information while generating each word of the question.

We implement this idea by concatenating the word embedding at each timestep of the question with the image embedding. Hence, the dimension of the input to the LSTM at each time step is 512 instead of 256 (256 from image embedding + 256 from word embedding). Also, since the image embedding is now available at each timestep, we discard the image embedding from the initialization of the first timestep of the LSTM. The remaining architecture is retained from the Show and Tell model. Fig. 2b summarizes this model.

4.3 Question Attention

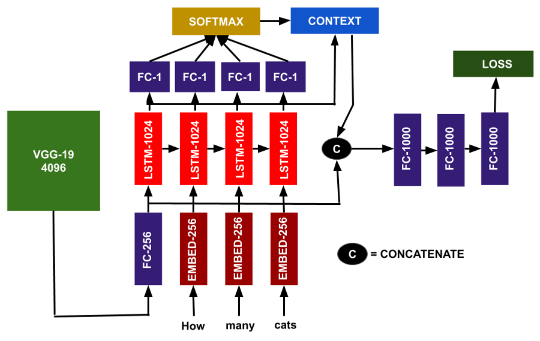
In the previous two models, we only pass the hidden state from the last time step of the LSTM to the fully-connected layers of the network. The last state is assumed to have aggregated information from all the timesteps of the question. However, if the aggregation does not happen in the desired way, the fully-connected layers responsible for answer prediction may not see the necessary information from previous timesteps. To solve this problem, we need to allow the fully-connected layers to access the hidden states from all the timesteps. A well-established technique in deep learning which helps us here is attention. Allowing the answer to attend to words of the question helps the network decide which part of the question to focus on while predicting the answer.

We implement attention on words of the question as follows. The hidden state at each timestep is passed through a fully-connected layer with 1 unit (shared across time). This gives us a scalar for the hidden state at each timestep which can be interpreted as a score. The softmax function is applied across time to all these scores to convert them into attention weights in the range [0,1]. We then compute the question context vector as the linear combination of the hidden states, each weighted by its corresponding weight. The new input to the fully-connected layers will now be the question context vector concatenated with the image embedding. The context vector helps capture the desirable combination of hidden states required for predicting the answer. The following equations describe this method.

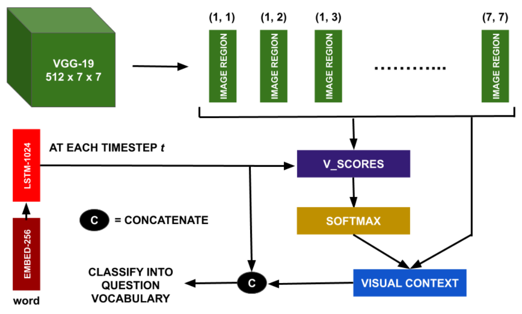
$$q\_score^{(t)} = FC(\mathbf{h}^{(t)}) \qquad q\_\alpha = \text{softmax}(q\_score) \qquad \mathbf{c}_{question} = \sum_{t=1}^{T} q\_\alpha^{(t)} \mathbf{h}^{(t)}$$

$$y_{answers} = FC([\mathbf{i}, \mathbf{c}_{question}])$$

where FC is a set of fully-connected layers, $\mathbf{h}^{(t)}$ is the hidden state of the LSTM at timestep t, q_α is the vector of attention weights, $\mathbf{c}_{question}$ is the question context and $\mathbf{i}$ is the image embedding. Fig. 3a summarizes this model.



(a) Question Attention



(b) Visual Attention

Figure 3

4.4 Visual Attention + Question Attention

The next model extends the attention technique to the input images. Visual attention is a popular technique used in applications which involve focusing on a specific region of the input image. VQA is one such application. A question about an image typically has a subject/subjects. Hence, it is natural to attend to different regions of the input image in search of this subject/subjects before predicting a word of the question.

Since visual attention requires attending to different spatial regions of the image, this network starts with the last pooling layer of VGG-19 which produces a $512 \times 7 \times 7$ dimensional feature map for the input image. This means that we have 7×7 image regions to attend to and each image region is represented by a 512 dimensional vector. At each timestep of the LSTM, a weighted dot-product of the hidden state with each image region (512 dimensional vector) is computed to obtain a score for each image region. These scores are then normalized to lie in the range [0,1] using a softmax function across the image regions. The normalized scores constitute the visual attention weights. A linear combination of the image regions weighted by their respective visual attention weights is computed to obtain the visual context vector. The visual context vector is then concatenated with the hidden state of the LSTM to be classified into one of the words of the question vocabulary. Fig. 3b summarizes this model. The following steps show the calculation at timestep t of the LSTM.

$$v\_score^{(t)}_{(r,s)} = \mathbf{i}^T_{(r,s)} A \mathbf{h}^{(t)}, \quad v\_\alpha^{(t)} = \text{softmax}(v\_score^{(t)}), \quad \mathbf{c}^{(t)}_{visual} = \sum_{m=1}^{7}\sum_{n=1}^{7} v\_\alpha^{(t)}_{m,n}\mathbf{i}_{m,n}$$
$$y^{(t)}_{words} = FC([\mathbf{h}^{(t)}, \mathbf{c}^{(t)}_{visual}])$$

where $\mathbf{i}_{(r,s)}$ is the 512 dimensional representation of the image region (r, s), $\mathbf{h}^{(t)}$ is the hidden state of the LSTM at timestep t, A is a trainable matrix, $v\_\alpha^{(t)}$ is the 7×7 attention weights at timestep t, and $\mathbf{c}^{(t)}_{visual}$ is the 512-dimensional visual context vector at timestep t.

Next, instead of just the hidden states, the concatenated embedding (visual context + hidden state) is passed as input to the question-attention module from the previous model. Hence, apart from attending to the words of a question , the network can also attend to the corresponding visual context of the word. The below steps explain this concretely.

$$vq\_score^{(t)} = FC([\mathbf{h}^{(t)}, \mathbf{c}^{(t)}_{visual}]) \qquad vq\_\alpha = \text{softmax}(vq\_score) \qquad \mathbf{c}_{vq} = \sum_{t=1}^{T} vq\_\alpha^{(t)}\mathbf{h}^{(t)}$$
$$y_{answers} = FC(\mathbf{c}_{vq})$$

4.5 Convolutional Attention

A common feature of all the previous models is that they use image features from a pretrained CNN model. Even the model with visual attention used features from the last pooling layer of VGG-19. Essentially, feature extraction (through convolutions) and visual attention were used as separate components of the model. A better approach would be to guide feature extraction using visual attention. In other words, instead of extracting features oblivious to the question, we can learn to extract those features which are relevant to answering the question. We propose to do this by interleaving convolutions with visual attention blocks.

More precisely, instead of staring from the $512 \times 7 \times 7$ output of the last pooling layer of VGG-19, we start from the output of the penultimate pooling layer of VGG-19 which has a dimension of $512 \times 14 \times 14$. In doing so, we retain the 4 pretrained convolutional layers between the last 2 pooling layers of VGG-19. We then interleave these 4 convolutional layers with 4 *attention layers* (explained later). These attention layers are meant to guide subsequent convolutions (and hence feature extraction). We allow the weights of these alternating convolutional layers and attention layers to be trainable so that the model can jointly learn to extract features and attend to image regions. The output of the alternating sequence of convolutional and attention layers is a $512 \times 14 \times 14$ dimensional feature map. The goal is to finally reduce the spatial dimension from 14×14 to 1×1 which gives us our new visual context vector. We achieve this reduction by passing the $512 \times 14 \times 14$ dimensional feature map through 2 more modules each with a sequence of convolutional, attention and maxpool layers. Thus, we have obtained a new visual context vector only using convolutions guided by attention. At this point we can contrast this with the original visual attention model which simply computes a linear combination of image regions to obtain the visual context vector. We believe that our new approach is more well-suited to implement visual attention since it involves using convolutions to compute visual context and convolutions are known to work well with images.

The attention layer encapsulates almost all the operations involved in computing visual context. The operation which differs is the last operation. Instead of computing a weighted linear combination of image regions, the attention layer stops at performing a regionwise scaling using the computed attention weights. The convolution step of the next layer takes over from here. The following steps summarize the computation at a single attention layer for timestep t.

$$v\_score^{(t)}_{(r,s)} = \mathbf{i}^{(t)^T}_{in\,(r,s)} A\mathbf{h}^{(t)}, \quad v\_\alpha^{(t)} = \text{softmax}(v\_score^{(t)}), \quad \mathbf{i}^{(t)}_{out\,(r,s)} = v\_\alpha^{(t)}_{(r,s)} \circ \mathbf{i}^{(t)}_{in\,(r,s)}$$

where $\mathbf{i}^{(t)}_{in\,(r,s)}$ is the input representation of the image region (r, s) relevant to timestep t, $\mathbf{h}^{(t)}$ is the hidden state of the LSTM at timestep t, A is a trainable matrix of the given attention layer, $v\_\alpha^{(t)}$ is the 2D visual attention weights at timestep t for the given attention layer, and $\mathbf{i}^{(t)}_{out\,(r,s)}$ is the output representation of the image region (r, s) relevant to timestep t. Fig. 4 summarizes this model.

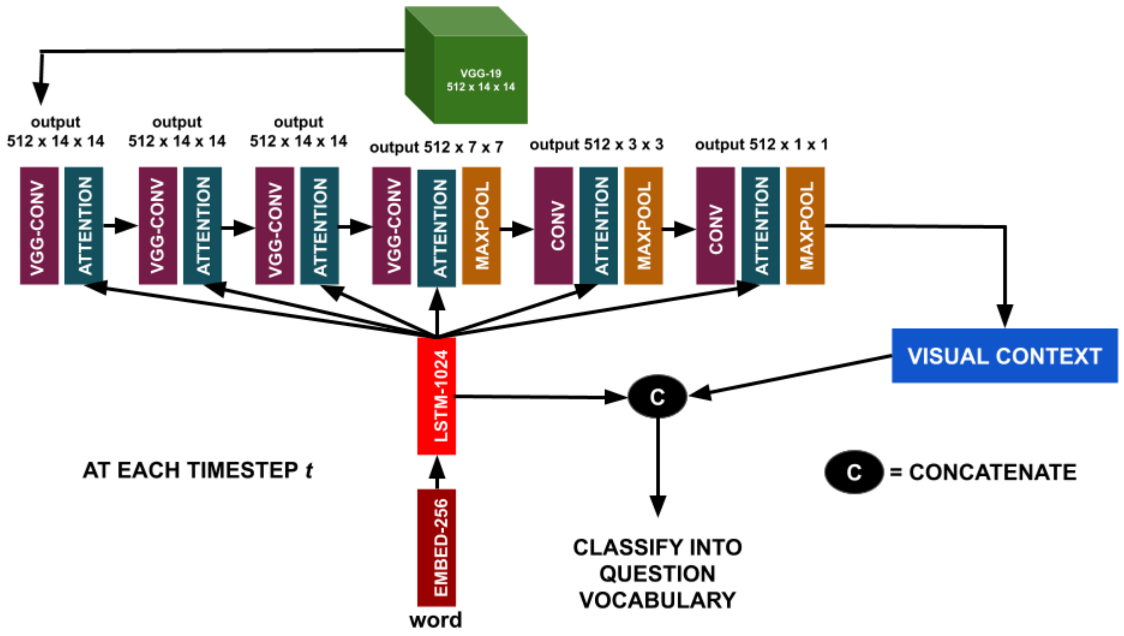


Figure 4: Convolutional Attention

5 Results

For our experiments, we first combined the training and validation sets provided in the VQA v2.0 dataset and used a 80-20 training-validation split ratio to train our models. This was done to have a higher ratio for training than the default (67.43%) provided in the VQA dataset since we train on questions as well. The results of our experiments are summarized in Table 1. We tested all our models on the Test-Dev Phase of the VQA challenge. The metric used to compare our methods is given by Eq. 1.

Model	**Accuracy** (%) (see Eq. 1)							
	With question training				**Without question training**			
	Yes/No	**Number**	**Other**	**Overall**	**Yes/No**	**Number**	**Other**	**Overall**
deeper LSTM Q + norm I (1)	-	-	-	-	73.46	35.18	41.83	54.22
Show and Tell	67.87	32.18	32.00	46.80	68.41	31.50	32.64	47.24
Time Dist. CNN	68.43	32.57	32.50	47.31	69.11	33.83	33.66	48.29
Question Attention	70.09	33.07	33.59	48.57	68.89	33.44	36.22	49.36
Visual Att. + Ques. Att.	67.35	30.83	30.47	45.70	62.92	28.43	20.60	38.92
Conv. Attention	-	-	-	-	66.13	31.08	25.37	42.81

Table 1: Comparative performance of proposed VQA models and benchmark from (1)

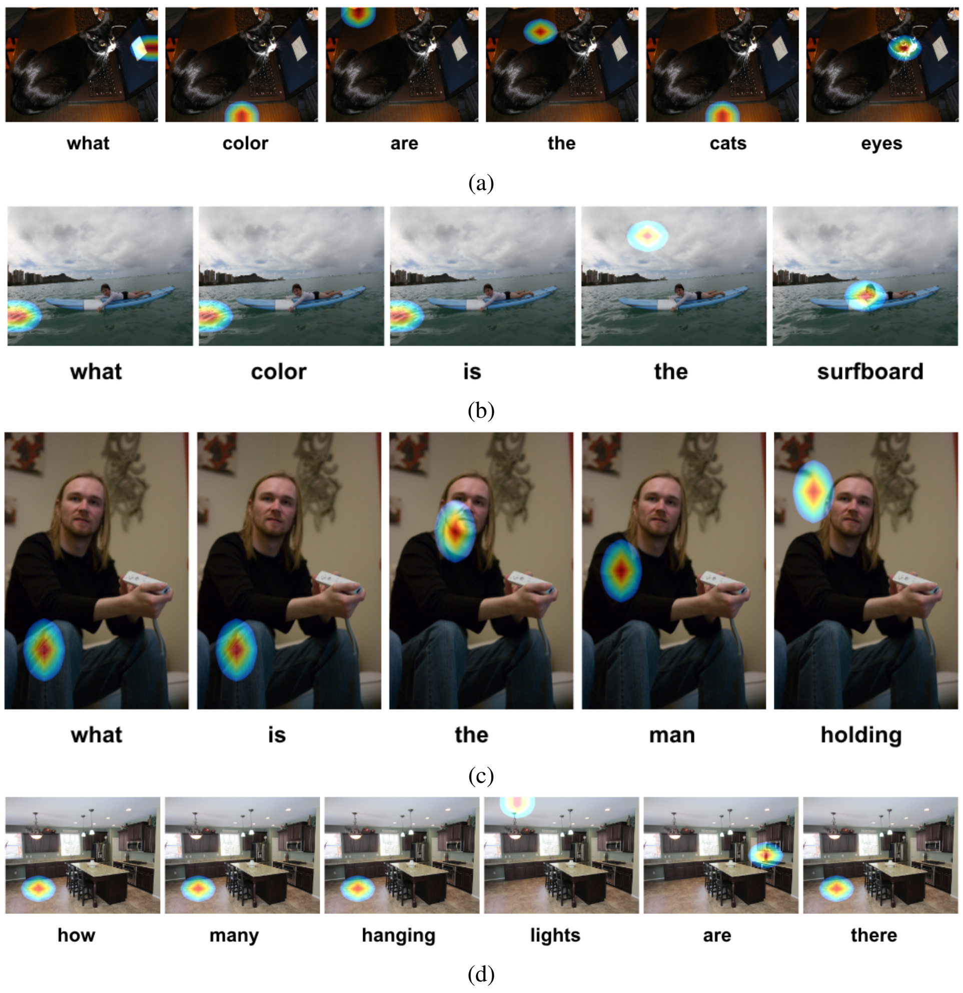


Figure 5: Attention maps generated by Visual Attention + Question Attention model at different timesteps of the question (code borrowed from (12))

Firstly, we observe in Table 1 that all the models are able to perform significantly better on Yes/No questions than other categories of questions. This is similar to the trend observed in our baseline experiments. Secondly, we see that training on questions has a negative impact on the performance of the model. Except for Visual Attention + Question Attention, all of our models gain around 1% without training on questions. Hence, we chose not to run the Convolutional Attention model with question training since this model takes significantly longer (3-4x) to train as compared to the other models. Thirdly, Question Attention performs the best among the models that we propose, with an accuracy of 49.36% without training on questions. In general, models which have access to image features at every timestep produce better results (with the exception of Convolutional Attention and Visual Attention + Question Attention, which modify the original image features at every timestep). An unexpected result for us was the poor performance of visual-attention-based models, especially the Convolutional Attention model. We expected that attending to the regions of the image would give the model better clues about where the answer can be found in the image, as has been shown in previous work outlined in section 3, but our experiments suggested otherwise. Finally, comparing our models to the benchmark in (1), our best model falls short by 5%. Moreover, our model mainly loses out on the ’Other’ category of questions. We believe that more elaborate hyper-parameter tuning could produce comparable or even better results, but our priority was to explore more models rather tweaking a single model due to constraints on compute resources.

6 Discussion and Analysis

6.1 Impact of Jointly Learning Question and Answer

It is evident from our results that jointly training over questions and answers is not conducive to our final task of classifying the answer. Interestingly, we found while training that the question prediction accuracy of all our models was consistently high (around 85%) but the answer classification accuracy always lagged it. This points to the relative ease with which the syntactic information in language modeling can be learned by the model in comparison to a complex task like comprehending language to answer a question about an image, which involves capturing semantic information. We think that the gradients arising out of the question prediction loss dominate the gradients from the answer prediction loss thereby, guiding the optimization towards a local minimum that has low question prediction loss. A weighted combination of these two apparently conflicting losses might be a simple approach to get around this but requires tuning of these weights.

6.2 Impact of Question Attention

Attending over different words of the question did improve our model's accuracy by 1%. This is because the subject in the question most relevant to the answer need not be at the end of the question. Attention enables the model to peek at all the words in the question and decide which ones to consider. However, since the questions are not very long (maximum length of 26 words) LSTMs can do reasonable well even without attention, thus giving only small improvements using attention.

6.3 Impact of Visual Attention

We were quite surprised by the poor performance of visual-attention-based models. The Convolutional attention model was expected to at least beat the Visual Attention + Question Attention model but this was not the case. We think that some more tuning of the dimensionality of the visual context features may have been required. We currently use 512 because this is the depth of the last VGG-19 layer we used, but perhaps a higher number like 4096 might have better represented the image features. In order to understand what visual attention was truly doing, we superimposed the attention maps generated by the Visual Attention + Question Attention model on the input image to see which parts of the image were being attended to for each word of the question. In Fig. 5a we see that the attention mechanism does well to narrow down to the subject of the question which is cat's eyes. But it is not clear what the first 5 timesteps are attending to. We notice a similar trend in Fig. 5b where the model attends to surfboard pretty well but the previous timesteps are not really obvious. Fig. 5d gives us insights about what our visual attention does in counting questions. It does well to track down one of the lights in the room when 'lights' is input, but it misses out on two other lights. Interestingly the answer output for this question was 'one' when the correct answer is 'three'. This means that our model does have some ability to correlate the visual attention with answer words, but it falters in the visual attention part and hence, may be getting the answer wrong too.

7 Conclusion and Future Work

We proposed 5 architectures based on our *VQA-dependency-graph* to tackle the problem of Visual Question Answering. Our novel contribution is the Convolutional Attention model which was intended to let the question guide the extraction of features from the image. We also proposed to jointly learn to generate questions and answer them so the model benefits from this multi-task training. Our findings suggest that this affects the answer prediction performance negatively and it is better to train the model on just the answers. We also saw that our visual attention model does learn to locate the subject of the question reasonably well but requires additional mechanisms to deal with challenging questions like number-based questions which requires the model to perform both question comprehension and object detection simultaneously. As our future work, we aim to focus on improving the answer prediction part of our model. We feel that treating each answer as a separate class is a flaw in the existing models for VQA and it might be better to learn embeddings for answers similar to the way word-embeddings are learned in language modeling. This might help capture semantic information between answers instead of treating them as disparate classes.

References

[1] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra, "Vqa: Visual question answering," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 4–31, 2017.

[2] A. Agrawal, J. Lu, and T. Khot, "Vqa." `https://github.com/GT-Vision-Lab/VQA`, 2017.

[3] M. Ren, R. Kiros, and R. Zemel, "Exploring models and data for image question answering," in *Advances in neural information processing systems*, pp. 2953–2961, 2015.

[4] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *Advances in neural information processing systems*, pp. 1682–1690, 2014.

[5] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus, "Simple baseline for visual question answering," *arXiv preprint arXiv:1512.02167*, 2015.

[6] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," *arXiv preprint arXiv:1606.01847*, 2016.

[7] K. Kafle and C. Kanan, "Answer-type prediction for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4976–4984, 2016.

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

[9] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 21–29, 2016.

[10] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *European Conference on Computer Vision*, pp. 451–466, Springer, 2016.

[11] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

[12] J. Gildenblat, "keras-cam." `https://github.com/jacobgil/keras-cam`, 2016.