COS529: Assignment 4 Report
Ayush Alag (aalag)

December 8, 2021

1 Background

Visual Question Answering (VQA) is the process of creating an intelligent model that can identify an answer given simply an image and a question [1]. Since 2015, when a large-scale dataset was created, VQA has been an ongoing challenge in computer vision [2]. The implications of VQA are immense, including disability assistance, intelligent robotics, and virtual chatbots. Thus, it is imperative that robust VQA algorithms are identified [1].

2 Approach

2.1 Data Processing

Dataset

Training and validation were performed on the VQA v2.0 dataset, specifically analyzing the Balanced Real Images subset that consists of 82,783 training and 40,504 validation images from COCO, as well as 443,757 training and 214,354 multiple-choice questions with corresponding answers [1] [3].

Question Modelling

Since there are virtually infinite amount of answers that a model could choose from, I sought to bound this problem by keeping only the questions that had the top 1000 answers during training time, and limiting the prediction space of the model to a 1000 x 1 vector. A vocabulary of the words in all of the questions was constructed, which corresponded to a vectorization mapping. To keep the model input to a standardized length, each question was capped at a length of 25 words, with shorter questions having 0 padding to fill the space. Thus, the question was inputted to the model as a 25 x 1 vector, where each element in the vector corresponded to a vocabularic mapping of the specific word at that position (*e.g.,* the sentence "the dog barked" could be represented as [3, 7, 986, 0, ... , 0]).

Image Modelling

I utilized the pre-trained image features from a ResNet deep convolutional neural network, where each input image corresponded to a 2048 x 1 feature vector. The input images were mapped to their corresponding questions by utilizing the 'image_id' tag that each question contained. Thus, the combined input to the model (described below) was a 25 x 1 vector for the question as well as a 2048 x 1 vector for the image.

2.2 Model Architecture

Overview

To tackle the VQA problem, I utilized a joint training approach, where the image encoding and question encoding were computed in parallel before they were concatenated and passed through a Multi-Layer Perceptron. A generalized visual of the joint training can be seen in the figure below:

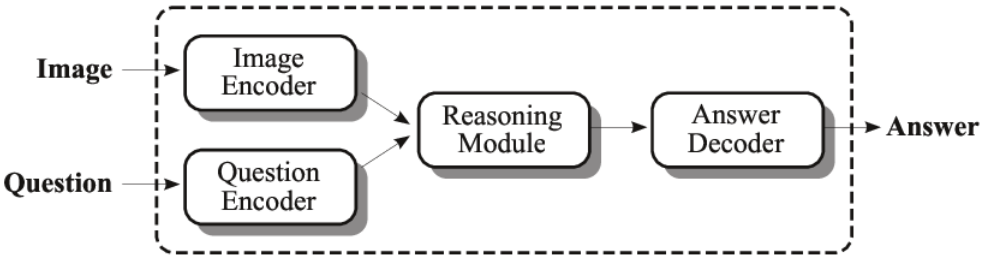

Figure 1: Joint Image-Question Training

Image Encoding

As mentioned earlier, the model receives as input a 2048 x 1 feature vector for each image that corresponds to the output of a pre-trained ResNet101 model [4]. From here, a fully-connected layer with the relu activation function yields a 512 x 1 vector.

Question Encoding

The model receives as input a 25 x 1 sequence vector, where each of the (up to) 25 elements corresponds to the index the word in that position resides in the overall vocabulary. Subsequently, an embedding layer maps each of the words to a 300-dimensional embedding. The embeddings used were pre-trained GLoVE embeddings, which were pre-trained on a different task by the Stanford NLP group in 2014 [5]. Next, the embedding matrix was fed forward through two layers of LSTMs, each with 512 hidden units. For the first hidden layer, a sequence (rather than a singular vector) was passed on to preserve the structure of the question; and between hidden layers, dropout with a probability of 0.5 was added to increase regularization and prevent overfitting.

Combining Encodings

There were several options to be explored when combining the two 512 x 1 encodings: element-wise addition, element-wise multiplication, outer product, and compact bilinear pooling. Emulating the given baseline, and seeking to start with a simple solution, I decided to use element-wise addition to start off with and, time permitting, explore the other options.

The 512 x 1 summed vector was passed through two 1000-node hidden layers, each with a relu activation. Again, dropout with a probability of 0.5 was used in between each layer to prevent overfitting. Finally, a 1000 x 1 vector was formed by passing the output of the last layer through a softmax activation, which yielded the results. Categorical cross-entropy was utilized for this multi-class classification task, where the ground-truth answers were in one-hot encoding form corresponding to the top 1000 answers identified earlier.

2.3 Training

The model was created using the Keras library in Python, a wrapper around Tensorflow. The model utilized vanilla stochastic gradient descent algorithm on the first try, and ultimately Adam was decided upon for the optimizer. The code was implemented in Google Colab Pro+ and trained

on a GPU instance for 15 epochs, which took 8 minutes. I created a data generator class which lazily loaded batches of the training data, so as to minimize the amount of RAM used.

3 Results

After performing data processing, training, and inference as shown above, the accuracy of the validation data was computed to be **47.11%**, which surpasses the baseline of 45%. Specifically, 88237 out of 187364 validation samples were predicted accurately.

3.1 Examples

Below are a few examples from the model's predictions on the validation set. A common theme seems to be that the model is able to discern the objects in the images well, but is unable to accurately understand the meaning of the questions, specifically which objects should be in focus and in what way they should be analyzed.

Figure 2: ID: 388531

Figure 3: ID: 69366

Figure 4: ID: 327567

Figure 5: ID: 237568

3.1.1 Correct Validation Answers

Image ID: Figure 2 **Question:** Where is he looking? **Answer:** Down
It is clear that the model was able to infer both the intent of the question (asking for a direction), as well as recognize the object's pose as to see the direction of his face. Thus, this is a fairly impressive result for the VQA model.

Image ID: Figure 3 **Question:** What is the man doing? **Answer:** Walking

Again, the model is accurately able to infer a complex task, such as the action of walking, from the image, as well as ascribe it to the individual question.

3.1.2 Incorrect Answers

Image ID: Figure 4 **Question:** What is to the right of the soup? **Model Answer:** Soup
Ground Truth: Chopsticks
As seen here, it is likely that the model is being overly biased by the wording of the question, which also contains an object in the input image. The combination of these two biases prevents the model from fully understanding the intent of the question, which asks the model to look to the right of a certain object.

Image ID: Figure 5 **Question:** How many photo's can you see? **Model Answer:** 2
Ground truth: 1
As seen here, it is likely that the error is induced by the difficulty of the question, as it asks a meta-fact about the image (or otherwise refers to some hard-to-identify object). However, the language model is actually doing quite well, as we deduce that the answer is numerical from the question alone.

4 Ablation

For the ablation experiments, I wanted to find out whether the model was indeed identifying images extremely well and failing to understand the question, or if, on the other hand, the model was using cues from the question to guess the answer without even seeing the image. The latter is a common case of questioning bias that would be insightful to uncover.

Thus, I detached one portion of the network from the other; specifically, I ran one experiment with solely the image encoding (**I**), and then ran another experiment with solely the question encoding (**Q**). I then compared the results of the three experiments (**I**, **Q**, and **I** + **Q**) in the table below:

Model	Accuracy
I	22.66 %
Q	37.87 %
I + Q	**47.11 %**

These results both affirmed and qualified my hypothesis. Firstly, I was shocked at how well the model performed with just the question, and no image supplied. It seems that the model was able to infer the type of answer ("yes/no", for instance) based on the phrasing of the question, and then create an educated guess based on potentially question priors or simply how frequent a potential answer was in the training data. The model performed much worse on the image-only embeddings; however, given that the model did not even know the question to answer for, an accuracy for 24.83% on just the image is fairly decent, and is likely the result of the ResNet101 model identifying objects very well and choosing the best response based on the object composition. Lastly, the fact that the image + question model outperformed both the question-only model and the image-only model shows that our classifier is learning novel insights that combine the two data streams, which is a positive sign.

5 Reflection

Initially, I sought to emulate the original VQA's paper exactly [1], which means taking the LSTM embedding by concatenating all of the memory and state cells from each of the hidden units, thereby yielding a 2048 x 1 vector that would then be fed into a fully connected layer of output size 1024 x 1.

This approach, while possibly better in the long run, led to an extremely slow training time where the loss seemed to plateau at certain points. Furthermore, the validation accuracy was much lower, at only around 33%. When I changed the LSTM embedding from the concatenation of the hidden states to the actual y-output of the LSTM, my training time was much faster and the accuracy was significantly higher. In the future, I would want to further understand the original VQA paper to understand how I may have misinterpreted their algorithm; or if my interpretation was correct, what steps I could take in order to have the model train as effectively as they did.

Additionally, I also found that using the 'relu' activation function over the outdated 'tanh' function led to much faster training convergence: I was able to achieve a loss of 20 epochs in only around 6-7. Finally, I experimented with multiple optimizers, and was initially using vanilla Stochastic Gradient Descent (SGD); however, I quickly found that the Adam optimizer was much more efficient.

6 Next Steps

6.1 Embedding Concatenation

As mentioned earlier, there are many key ways of combining the image and question embedding vectors. These include element-wise addition, element-wise multiplication, the outer product, and the compact bilinear layer. In this project, I utilized element-wise addition; however, there was no reason to perform this operation over element-wise multiplication. Thus, I tried supplanting the addition layer with the multiplication layer, which resulted in a marginal decrease in accuracy 46.93%. The multiplication layer also took longer to train than the addition layer. I suspect that this may be due to fewer operations when computing the gradient during backpropagation for the addition layer.

As future work, I would want to try implementing the compact bilinear layer, which would not be as expensive as an outer product but still capture more dependencies between the image features and question features at different indices. I hypothesize that the model is currently able to both understand the question well and interpret the image well; the biggest bottleneck is learning how the question affects the way to process the image. Although changing the combine layer from Addition to Multiplication only slightly affected accuracy, I estimate that with the compact bilinear layer, I could achieve an accuracy of around 50%. This 3% increase stems from the increased interdependencies between features of different indices in the compact bilinear layer.

6.2 LSTM Layers

Another factor I would like to try would be increasing the number of layers in the LSTM while simultaneously increasing dropout to prevent overfitting. My hypothesis is that currently, the classification model is unable to fully understand the nuances of each question, and that adding further layers to the LSTM would help improve the translation between the free-form text and the embedding. Simultaneously, I'd look to add a dropout to not only the fully-connected layer, but also between each of the layers of the LSTM.

I attempted such an experiment, changing the number of layers from 2 to 3 and adding a dropout with probability 0.5 between each layer. After training with the Adam optimizer for around 20 epochs, the accuracy actually decreased to 45.85%. The fact that the accuracy decreased after adding further layers may suggest that the question representation is sufficient for the task, and that a simpler representation may be more optimal. More broadly, it suggests that more focus be put into the embedding-concatenation layer (as mentioned above) or other ways to jointly interpret the logical flow of a question, rather than focusing on solely image embedding or question embedding.

References

[1] Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE international conference on computer vision. 2015

[2] Wu, Qi, et al. "Visual question answering: A survey of methods and datasets." Computer Vision and Image Understanding 163 (2017): 21-40.

[3] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

[4] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)

[5] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.