VISUAL QUESTION ANSWERING

Akshay Sharma(14062), Bhawesh Kumar(14181), Mandeep Singh(14363), Mayank Garg (14375), Nikhil Baranwal (14425)

CS671A: Introduction to Natural Language Processing
Indian Institute of Technology Kanpur

***Abstract* - The focus of our project is Visual Question Answering (VQA). In particular, we focus on questions with open ended answers. We use a Multi-Layer Perceptron (MLP) model with softmax classifier on the dictionary words for generating answers. We try various word embedding techniques (BoW, GLoVe, word2vec) for representing text questions and pre-trained CNN (VGG16) for representing Images. We then implement the LSTM architecture for text questions with GLoVe embedding for words. We also implement a unique multi-model approach in which we distribute questions in 3 different classes depending on the type of answer expected. Different neural-architectures are used for different classes of questions. Finally, we briefly review the emerging technique of image-question co-attention for VQA. We conclude with a brief discussion of our results and future works. MS COCO dataset is used in all models.**

***Index Terms*— CNN, word2vec, GloVe, LSTM, MLP**

1. INTRODUCTION

Visual Question Answering deals with answering text based questions about a given image. The question can be open ended type, yes/no type, MCQ type or numerical answer type. The VQA task requires concepts and techniques from Computer Vision and Natural Language Processing. Image features are extracted using Computer Vision techniques, while text features from questions and answers are extracted using Natural Language Processing techniques. A VQA task is challenging for a computer because any algorithm hoping to perform such a task must meaningfully combine the text and image features which comes from very distinct feature spaces[1][6].

1.1 Motivation

Computers are now easily able to do tasks like object recognition, scene classification quite well. However, the ability of the computer to extract deeper semantic meaning from the questions remains lacking. VQA provides a real test for such understanding. Besides, VQA also has some important real world applications. Imagine a visually impaired person who could just click an image from his phone and he would be informed about his surroundings. VQA can also be used to perform Visual Turing Task..

1.2 Overview

In the project, we implement a trained model for VQA. We use MS COCO (2014) dataset for training and testing[2. The MS COCO dataset has 82,783 images and 248,349 questions for training and 40,504 images and 121,512 questions for testing.
In Section 2, we describe the basic pipeline of VQA. In section 3, we try a new approach involving classification of questions based on the type of answer expected. Section 4 describes the experimental setup of the methods mentioned for this project. In section 5, we discuss our results. In the final section, we briefly describe scopes of improvement in the VQA task and future works.

2. BASIC PIPELINE

The basic pipeline to solve this problem is to get features from images and questions and then combine them and use them to predict the answer using a classifier (MLP). The top most answers from training dataset is used as different classes in which our answer is predicted.[5][6]

2.1 Image features

Mainly the image features are extracted through Convolutional Neural Network (CNN). CNN is used to extract features from the images because it takes care of the structures and locality in the image. A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. Many pretrained models are available for CNN: ResNet, AlexNet, VGGNet, GoogleNet etc [7][8]

2.2 Text features

We have used four different word embeddings namely word2vec[11], GloVe[12], BoW and LSTM[9] (using

GloVe). In the first three embeddings, for every question we take the average of its word embeddings and use the resulting vector as the question vector. Although this approach seems reasonable, this leads to loss of information related to the relative position of the words of the sentence, i.e. a question with all constituent words jumbled up will produce the same question vector.
To address this issue LSTM based GloVe embeddings was used. As LSTM preserves the sequential information it gives a much better representation for the question vector.

2.2.1 LSTM

LSTM (Long Short Term Memory) units are used in RNN layers to model sequential information[9]. A basic LSTM unit works by using a feedback loop in which the output of one time step is used as an input for the next.

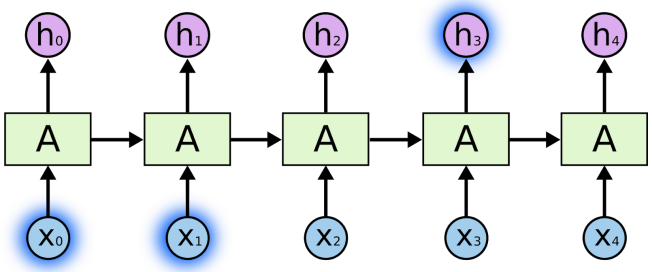


Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Apart from using the sequential information directly LSTM cells also have three types of gates namely (forget, input, output). Using the forget gate the network can learn whether to forget the past information or update it. This ensures that if needed the network can remember information in long sequences like the questions that are used in visual question answering. This system ensures that the information regarding the relative position of words in the question is taken into account when the final question vector is generated.

2.3 MLP

MLP (Multilayer Perceptron) is a type of vanilla feed forward neural network as they usually consist of only one hidden layer [4]. MLP consists of more than one perceptron and consists of input layer and output layer along with multiple hidden layers in between (as shown is the following image). Generally, MLP consists of a single hidden layer which is ideal for approximating any continuous function. MLP are extremely helpful in the classification problem

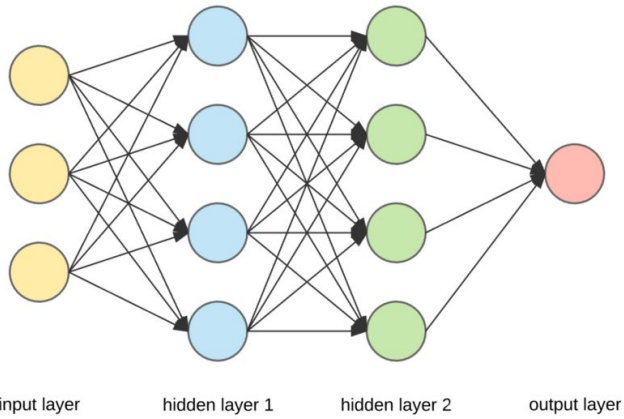


Image source: https://medium.com/@ksusorokina/image-classification-with-convo lutional-neural-networks-496815db12a8

The features extracted from 2.1 and 2.2 are concatenated and sent to MLP described in 2.3 to predict the correct answer.

2.5 IMAGE QUESTION CO-ATTENTION [3]

The idea is to emphasise more on certain important words in the question and get their relationships in the image so as to get the desired context in the question as well as in the image.
Two co-attention strategies have been proposed by *Lu et al.*
Parallel Co-attention (attends to image and question simultaneously)
Alternating Co-attention (sequentially alternates between generated image and question attention maps)
Following is the approach used which is based on alternating co-attention:

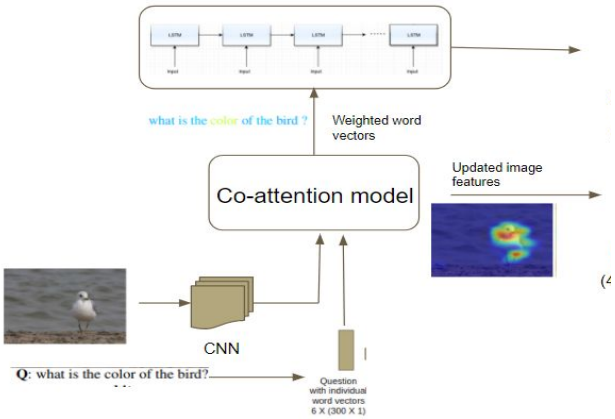
Given an image feature map V $\in R^{dxN}$, and the question representation Q $\in R^{dxT}$. Initially, the similarity (the affinity matrix C $\in R^{TxN}$) between each word vector, q_i in the sentence with the image feature map locations, v_j is calculated by using the following formula, as defined by *Lu et al.*

$C = tanh(Q^T W_b V)$,where $W_b \in R^{dxd}$ contains the weights.

After that, the image feature vectors and question vectors are attended by using an attention operation $\hat{x} = A(X, g)$ defined by *Lu et al.* along with the guidance vector, g. (Here weights used for question and image feature vectors are shared in different layers). First, the intermediate attended question feature $\hat{s}$ is generated by passing Q, question

feature with no attention guidance, i.e. g=**0** to the attention operator, A. Then, $\hat{s}$ is passed as an attention guidance vector along with the image feature map, V as input to generate the attended image feature, $\hat{v}$ which is only the weighted feature map without addition as mentioned in the paper. Finally, attended question feature vector, $\hat{q}$ without the addition of weighted word vectors is generated by passing the question feature, Q along with the guidance of attended image feature $\hat{v}$. This process can possibly be reiterated by first passing image feature map in the first step to get the intermediate attended image feature $\hat{t}$ and following the compliment steps as mentioned above.

Then, after the generation of attended word vectors and image feature maps are generated, the weighted word vectors are passed to the LSTM to get the overall question vectors. The whole pipeline is depicted in the following diagram:



3. OUR APPROACH

We also use a multi-model approach in which we distribute questions in 3 different classes depending on the type of answers we expect. The intuition behind this is, when we are asked a question, we somehow know what kind of answer is expected to be given. For example : if someone asks us questions like “Is there a dog in this picture?” we somehow know that the answer is going to be either yes or no. Also, for questions like “How many people are there in this image?”, it is expected that the answer is some integer value.

One simple approach to classify these questions on the basis of answer type is to look at the question word itself i.e question words like what, where, when etc should each be given a separate category. But this trivial method is not expected to achieve satisfactory results. The issue lies in the fact that some question words (such as what, which) can lead to a wide variety of answers, for example, questions like “what is the man doing in this picture?” expects the answer to be an action but if someone asks “what is the time in this clock?” the answer should be a value. This type of ambiguity in answer type based on question word theoretically limits the model. However, sometimes the type of questions that can be asked are limited by the motivation behind asking the question, for example, questions asked in Visual Question Answering are way different from the open ended questions like “what is the capital of India?”. Here, the questions that are generally floated are of the form “Is there a cow in this picture?” which gives either a “yes” or “no” as an answer. That means even if we just make two classes of questions one with “yes”/”no” types and the other with remaining answer types, we expect the overall accuracy to increase as the bias created in single model approach due to a lot of training data giving “yes” or “no” as an output is now removed.

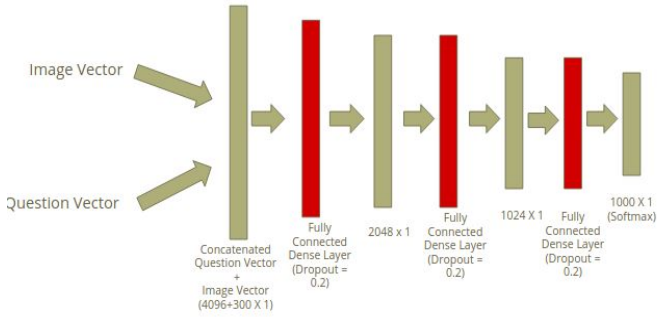
4. EXPERIMENT AND SETUP

We have experimented with both: basic pipeline and our approach in this project:

4.1 Basic Pipeline

First we have tried to implement a basic pipeline from scratch. For this we have used MSCOCO dataset.[2]
The MSCOCO dataset has 82783 images with three questions asked per image in the training set. It has 40504 images with three questions per image in the test dataset. There are 10 human annotated answers in the test dataset. Because of hardware limitations, we have taken 30000 data points for training and 15000 data points for testing.
We have borrowed VGG16 CNN features from stanford repository for image features [10]
We have used three different methods to extract features from questions: Google news dataset for W2V, Stanford dataset for GLoVe, and wrote code for BoW. We have also tried LSTM using GloVe features.
We have concatenated the image features and question features and passed it through MLP. We have used MLP with two hidden layers. Output of this MLP is 1000 x 1 vector corresponding to the top 1000 answers. Top 1000 answer from the training dataset was covering 87% of the

total dataset. So, choosing it is not a bad assumption. Our MLP looks like:



We have then compared each predicted answer to test answers and reported it correct if it matches with at least three out of ten of the human annotated answers.

4.2 Our Approach

On exploring our training dataset of 30000 observations we found that questions that had question words like 'Is'/'can'/'should' etc gave answers as either 'yes' or 'no' 11338 out of 12036 times. This implied 2 things - firstly, a lot of questions asked on images give only "yes" or "no" as an output (38% in this case). Secondly, in questions that begin with 'If'/can'/'should' etc 94% of the time we had to choose the right answer from only two possible outcomes. That means if we just make a separate category of these questions and train it separately, we expect the accuracy of output to be higher. Following this observation questions were classified in the following three categories -

1. Questions with question word 'Is'/'Are'/'Do'/'Does'/'Has'/'Have'/'Can'/'Should'/'Could'/'Will'/'Shall'/'Would'
2. Questions with question word 'How many'
3. Other questions

The questions were then trained in three different neural nets with similar architecture but different outputs. For the first neural net sigmoid was used an activation function in the last layer to give one output between 0 and 1. In the second neural net, softmax was used as the activation function with 100 integers as probable output and in the third neural net, top 1500 possible answers were chosen as probable outputs.

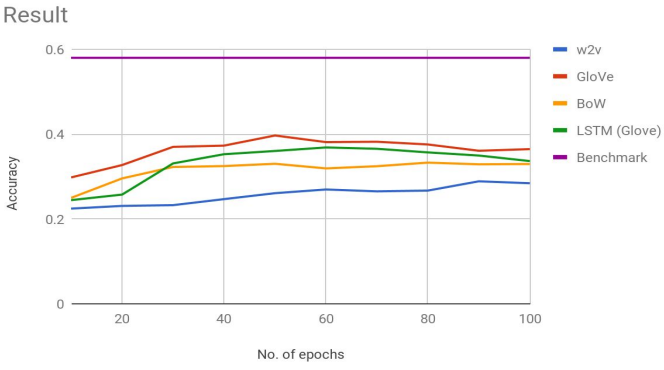
5. RESULT AND DISCUSSION

For the basic pipeline, we have tested each model for 100 epochs and got the following accuracy.

Q: What color is the clock?
A: Blue
Q: How many clocks?
A: 1
Q: What color is the hand on the clock?
A: Yellow

Q: What color is the balloon?
A: Blue
Q: How many elephants?
A: 2
Q: Are the people wearing hats?
A: Yes

Epoch/Model	w2v	GloVe	BoW	LSTM (Glove)
10	0.2244	0.2979	0.2497	0.2444
20	0.2308	0.3271	0.2956	0.2575
30	0.2326	0.3700	0.3226	0.3309
40	0.2468	0.3729	0.3247	0.3526
50	0.2608	0.3968	0.3301	0.3603
60	0.2695	0.3812	0.3192	0.3686
70	0.2651	0.3820	0.3244	0.3654
80	0.2669	0.3756	0.3328	0.3569
90	0.2888	0.3608	0.3288	0.3495
100	0.2842	0.3646	0.3296	0.3365



We were getting a good accuracy (around 40%) using a very less dataset and in the basic pipeline. We thought that the accuracy would improve further after introducing LSTM but it didn't happen as depicted in the graph. We were not able to debug this error.

In our approach where we are using multi model method, we have got an accuracy of 66.56% for Yes/No type of questions, 32.80% for numerical answer type questions and 21.5% for other type of questions, for 30 epochs using GLoVe as feature vector for questions. These accuracies are not upto the mark. We thought we would get upto 80% accuracy for Yes/No type but it didn't happen unfortunately. So, we didn't try other variants in this approach.

We also wrote the code for co-attention but unfortunately finally we weren't able to implement it due to some unresolved bug and time constraint as the system changed during the training time of the whole LSTM+Co_attention pipeline.

6. CONCLUSION

As our modest results show, the work on VQA is still in an early stage. However, Deep Learning approaches have shown promise on this task and they are expected to remain the weapon of our choice. In the last two years, there has been some work on including question and image Co-attention in the VQA pipeline and it has led to modest improvement in the State of the Art results[1][3]. However, the role of question and image Co-attention needs to be explored further for VQA tasks as it can even further improve the results on VQA tasks.

REFERENCES

[1] Liang-Wei Chen, Shuai Tang (2017). "Visual Question Answering." http://slazebni.cs.illinois.edu/spring17/lec23_vqa.pdf

[2] http://www.visualqa.org/vqa_v1_download.html

[3] Hierarchical Question-Image Co-Attention for Visual Question Answering, Jiasen Lu, Jianwei Yang, Dhruv Batra,, Devi Parikh, NIPS 2016

[4] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2, 359–366.

[5]VQA: Visual Question Answering, Antol et. al, ICCV 2015

[6] Akshay Kumar Gupta. "Survey of Visual Question Answering: Datasets and Techniques." 2017

[7] Razavian, Ali Sharif, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition." In Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 512-519. IEEE, 2014.

[8] He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034)

[9] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[10] https://cs.stanford.edu/people/karpathy/deepimagesent/

[11] Goldberg, Yoav, and Omer Levy. "word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method." arXiv preprint arXiv:1402.3722 (2014).

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