Visual Question Answering

Zi He
Department of Electrical and Computer Engineering
University of California, San Diego
z1he@eng.ucsd.edu

Daoyu Li
Department of Electrical and Computer Engineering
University of California, San Diego
d6li@eng.ucsd.edu

Renjie Zhu
Department of Electrical and Computer Engineering
University of California, San Diego
rezhu@eng.ucsd.edu

Houjian Yu
Department of Electrical and Computer Engineering
University of California, San Diego
h9yu@eng.ucsd.edu

Abstract

Combining computer vision and natural language processing (NLP), Visual Question Answering (VQA) has become one of the most focused machine learning research areas. With object detection in vision and semantic understanding in NLP being well studied, researchers have put more effort to scene understanding of images to achieve higher level artificial intelligence. In our report, we mainly tried to implement a LSTM based question representation combining with a CNN based image feature extractor to obtain the final answer. A transform learning method has been applied to our model, where we use Stanford NLP pretrained word vectors to form the dictionary and the VQA 2017 winner image feature extraction model. The accuracy of our prediction reached around 57% with the Multi-level Attention Network model.

The GitHub link: https://github.com/Nwoodle/Visual-Question-Answering

1 Introduction

A VQA system is designed to answer the natural language questions based on understanding of the image information. This process not only require multi-classes object detection in computer vision, semantic understanding in natural language processing as well as understanding in their mutual relationship[1]. The majority challenge for VQA is to fill the gap from language to image based on well performed CV and NLP algorithm as prerequisites. Not only the visual system has to be precise with high recognition accuracy, but it needs to provide the AI system with orders for specific object class. Moreover, with different features the question asks for, the system is supposed to generate corresponding feedback. For instance, when asking "how many people are there? ", our system will focus first on people recognition, then count number of the bounding box if applicable.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

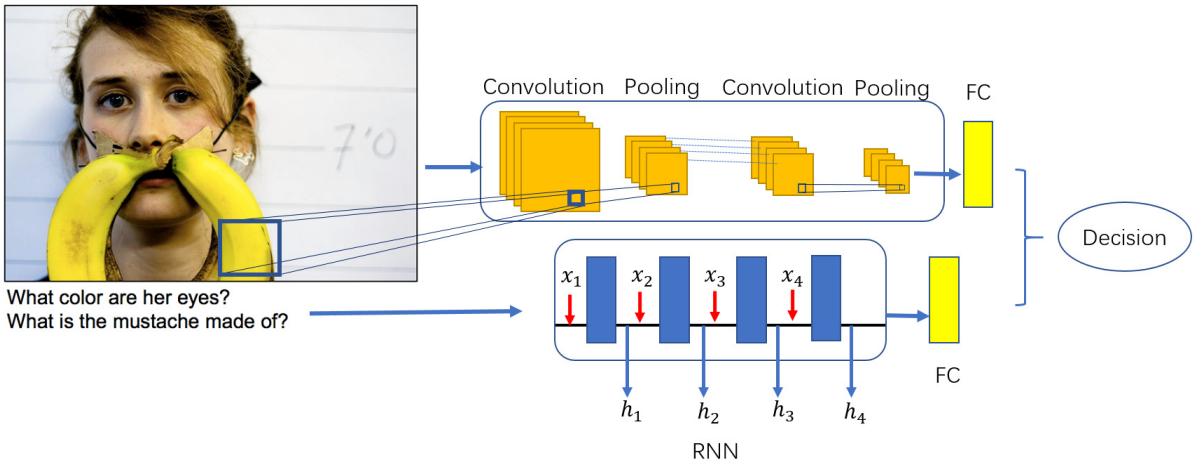

Figure 1: VQA overview. The image will be fed into a CNN for the feature extraction while the question splitting in single word in high dimensional vector form will be fed into RNN to record the status changes. Then according to the results combining in the fully connected layer, the final decision for the prediction of the answer will be provided.

One of the reasons VQA has been widely studied is for multiple beneficial applications in real life. VQA can be applied to aid visually-impaired users and blind users. With visual understanding and reasoning into use, humans and machine can interact for information extraction even without seeing the scenario in one's flesh. For instance, blind users will be able to understand their surroundings by simply asking questions to the AI system[8] ("Is it raining outside?"). VQA can also apply to Human Computer Interaction (HCI) in intelligent system or robotics remote control. For example, users order the robotic arm to fetch a laptop among multiple objects on the table. With such substantial potential, solving the VQA problem would be of high beneficial.

Aiming to solve VQA, multiple solutions has been proposed[5, 6, 7]. One of the general approaches for VQA is to use an attention based model[5], which basically adjusts the weights for the representation of the forefront based on the CNN training ignoring the background and only focus on the salient objects mentioned by the input question. [6] proposed a LSTM based question representation, CNN based image feature representation approach to even solve the linguistic VQA in both English and Chinese. Similarly, researchers in [7] designed a jointly trained end-to-end LSTM and CNN architecture for the answer prediction.

Image captioning has similarity to VQA, which is able to use the semantic segmentation in computer vision then feed the feature representations to decoder like Long Short-Term Memory (LSTM) or Recurrent Neural Network (RNN) for captioning[2]. On the other hand, VQA has different architecture from image captioning by using image channel and question channel model simultaneously to get the output[3]. Comparing with image captioning, VQA is indeed more challenging in finding the reasoning rather than simply output predefined critical words. For instance, in Fig. 1[4], in order to have the reasoning of the question, we first need to locate the face related area of the woman, which is a object detection problem. Then, our AI system have to understand the word "mustache" and directs itself to the mustache feature in our classifier. However, the "mustache" in our image is actually made of "banana" rather than real mustache, which makes our reasoning process even more complicated because banana has its own feature and cannot relate to mustache directly.

In our project, we implemented two different methods based on [3,12] and try to make improvement on them. In Section 2, we compare the VQA related work and computer vision based task in language meaning representation e.g. image captioning. Section 3 provides a LSTM question embedding and VGGNet[3] image embedding. Moreover, Section 4 introduced the bottom-up and top-down attention network consisting of semantic attention network and visual attention network[12]. Section 5 introduces the experiment setting and results. Conclusion is provided in section 6.

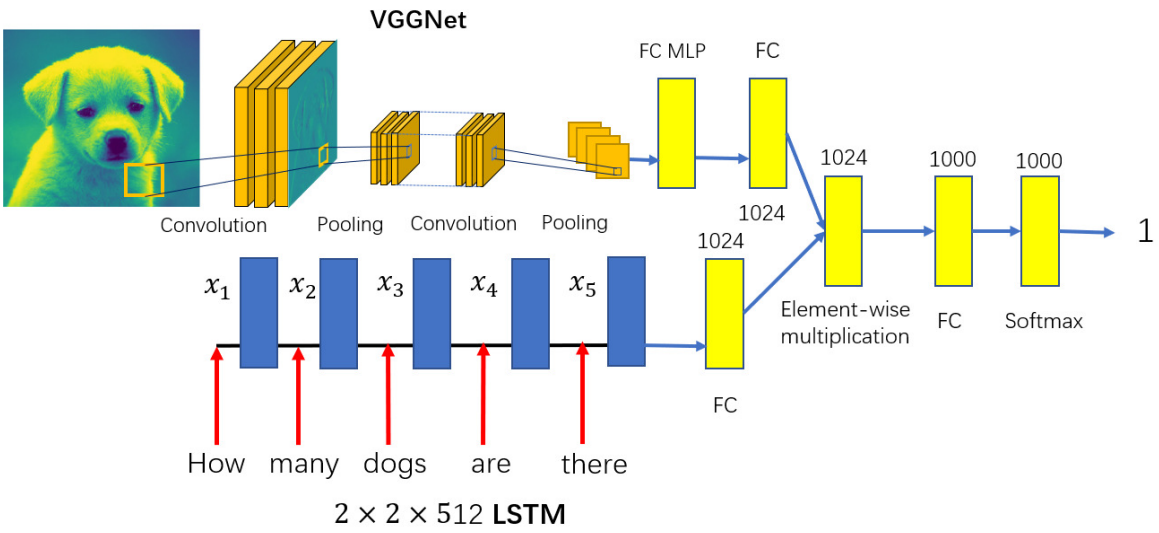

Figure 2: For encoding the question and the image, a two layer LSTM and a VGGNet are used respectively. After forwarding to the fully connect layer, the semantic and image features are fused by using the element-wise multiplication for 1024 dimension. And eventually the softmax function will provide answers with different probability. We will choose the top one prediction.

2 Related Work

Previous work used deep neural network architecture has solved multiple problems in computer vision and NLP. For NLP, the Recurrent Neural Network which can connect the previous knowledge of the sentence is quite useful for sentence reasoning and captioning[14, 15]. Machine translation develops faster with LSTM, an variant of RNN, widely used[16, 21]. In computer vision, VGGNet[17], GoogLeNet[18], ResNet[19], DenseNet[20] have achieved incredible lower error rate in image classification with less training parameters and time consumption. By using their pretrained model, it greatly reduces our training time consumption and computation complexity. We also changed the training set from over 64GB to 24GB in training our fully connect layer, which enables us to train our model on private computer with basic version of GPU.

Despite the wide deployment of CNN-RNN architecture in VQA[3,5], [3] can only achieved a high performance with the dependence on question type and cannot provide a general solution. Similarly, [5] used a relatively small size dataset with 16 types of colors, 894 object classes, which cannot provide solutions to vocabularies that are not predetermined. Moreover, fully RNN network for natural language processing has shortcomings. For instance, when processing sentences like "The man wearing red T-shirt holding ice-cream is my physics class professor.", the main goal is to introduce that "the man is my professor". However, RNN will process every word in the sentence. Thus, the longer the sentence is, it inclines more to process redundant information and creates a gap between the main object and the descriptive meaning. Thus, we use LSTM here to control the weight of input vocabulary vectors which only proceeds the important information in the sentence to avoid the long term dependency problem[22].

3 Naive Two-Channel Visual and Question Model

In this section, we try to implement the LSTM based two-channel model from [3] with a relative smaller dataset compared with the model we discussed later. This model is more like a demo to test if the transfer learning in visual encoding and the question representation with GloVe can work well.

The reason why we call it a naive model is that: first, it uses the LSTM encoder for the question sequence as the general solution; second, it uses the normalized *VGGNet* to encode the whole image without filtering out the desired area. In such a case, the network will treat every image pixel

equivalently. Furthermore, given the classification top 1 accuracy about 70% of VGGNet[11], we cannot expect a really high performance of this VQA.

We follow the default set from [3], whose outputs are selected among 1000 most frequent answers. For the image channel, we adopt VGG16 and set the last hidden layer to be 4096 dimension for image embedding. In order to align the dimension for both image embedding and question embedding, a multi-layer perceptron (MLP) is used here to size the output dimension to be 1024. For question channel, we perform the two hidden layers LSTM model, where the LSTM concatenates the last hidden layer state and the output of the cell state to get a 2048 dimension representation as showed in Fig. 2. Then, the output is passed through a fully connected layer to reshape the dimension to be 1024. Moreover, after the element-wise multiplication, the embedded visual question representation will be passed to a new MLP as a classifier. The MLP has two hidden layers for 1000 hidden units in each layer. Eventually, after passing through the softmax function, we can get the top 1 class as the output for the visual question answer.

4 Bottom-up and Top-down Attention Networks

In this section, the VQA solution is provided combining a visual attention, a semantic attention network in the VQA 2017 winner's framework[12]. The difference between attention based network and the naive method are huge. In stead of directly element-wisely multiply the embedded question and visual representation, we narrow down the searching area not only for processing specific regional parts of image via faster R-CNN, but also get the corresponding classes from the question representation. It highly increases the answer accuracy and robustness.

The framework of this model is shown in Fig. 3. The bottom-up visual attention model of faster R-CNN for image is implemented here with over 24GB data from *Microsoft COCO* dataset with $82,783$ images[10]. The visual attention network decides regional features rejection probability, based on which an image mask containing only the question related pixels will be derived. Finally, the last network after the element-wise multiplication is quite similar to forwarding layer in Section 3, which can give the final answers.

For the top-down LSTM model, We adapt the training set size to overcome the memory overflow problem. For simplicity, we directly use the Stanford Global Vectors for Word Representation (GloVe) pretrained word vectors to form our dictionary, which has 6B token, 400k vocabulary, about 822MB data[9]. We get our question representation v_q via the dictionary from GloVe and the proposed question passing through GRU.

4.1 Top-down Semantic Attention Network

Semantic attention network has question embedding and question words highlighting goals. In such a case, this network has to record the question state with the order of time t and combine the detected classes knowledge with the hyper parameter for question representation generated from GloVe.

Even after getting the answer to the object detection, the detected object won't be further proceeded for all. Because we only use regional pixels which are informative and related to the question. As a result, the embedded question representation together with the detected concept will help us determine the object mask from the semantic perspective. Moreover, the detect concept in single word will use the same dictionary as the question representation.

Our object detection concept is trained with COCO image captioning dataset[23]. p_c^{img} can be derived here by taking the activation function f_c for each image I in training set. It is the confidence of the objects names occurring in images.

$$p_c^{\text{img}} = f_c(I).k \tag{1}$$

From now on, we will use the semantic relevance to filter out the effective words. At beginning, we pass the question one-hot vector representation matrix $Q = [q_1, q_2, ..., q_T]$ for each word with the

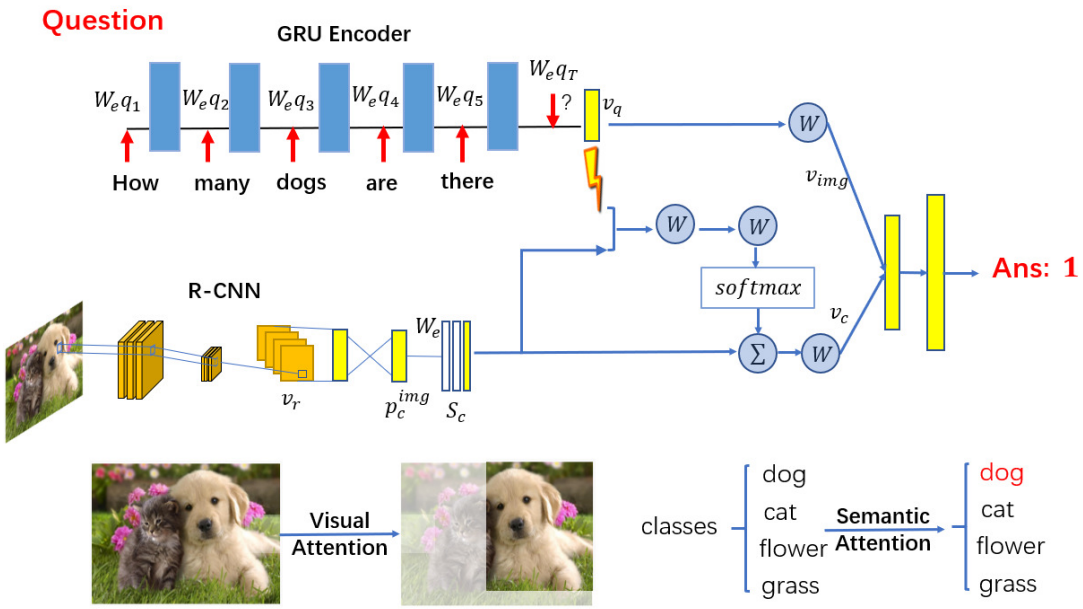

Figure 3: The framework of the attention network. The cat and dog image is from ECE 285 slide Chapter 5, pp 20.

embedding matrix W_e through GRU to get the question representation v_q from the last hidden layer of GRU. This process is described as:

$$x_t = W_e q_t, \tag{2}$$

$$h_t = \text{GRU}\left(x_t, h_{t-1}\right), \tag{3}$$

$$v_q = h_T, \tag{4}$$

For detected concepts representation, the same embedding matrix W_e is used here at the first layer for concept vector. Then at the second layer, a different embedding matrix W_c is used to project the concept vector to the same dimension to form the semantic representation s_c.

$$s_c = W_e\left(W_c c\right) \tag{5}$$

where c is the one hot vector for concept representation. Given the image concept representation and the GRU question representation, we compute:

$$a_{i,t} = \boldsymbol{w}_a^T \tanh\left(W_{va}\boldsymbol{s}_{c,i} + W_{ha}\boldsymbol{v}_q^i\right) \tag{6}$$

$$\boldsymbol{\alpha}_t = \text{softmax}\left(\boldsymbol{a}_t\right) \tag{7}$$

$$\hat{\boldsymbol{v}}_t = \sum_{i=1}^{K} \alpha_{i,t}\boldsymbol{v}_i \tag{8}$$

where ω_a^T,W_{va},W_{ha} are the learned weight parameter. $\alpha_{i,t}$ is the attention weight for each of image feature s_c during the time step t.

In such a case, we can highlight the detected classes related to the question in this semantic attention model.

4.2 Bottom-up Visual Attention Network

Semantic attention model helps us to understand the detected objects and the highlight words in the question, however, a visual attention network is still necessary for marking the target regions rather than the whole image. For instance, the background will be of smaller interest to detect and won't provide us with enough information. Furthermore, given that the images shot by a camera will lose the depth information from the original 3D world, the spacial information is blurred. Ideally, a 3D reconstruction method will give much more information towards the scenario understanding. But it would be time consuming and we normally just use one camera recording the scene at one specific moment. Thus, regaining the spacial information from 2D images can greatly alleviate the reasoning process.

For visual attention part, we first use the bidirectional GRU encoder from the last convolution layer of the feature extraction step. Thus, the regional visual representation is embedded into a left-to-right and top-to-bottom order. The reason we use bidirectional GRU for image information is to obtain the knowledge for image region surrounding information to preserve the spacial relationship between regions. Moreover, the image vector feature output from bidirectional GRU will have the same dimension as the question vector embedded in another GRU encoder.

After that, we follow [12] to measure the attention scores to determine the image mask. A multi-layer perceptron is deployed here to parallel the question representation and the region feature. Specifically, We compute the element-wise multiplication of the two vectors and feed the result into MLP to learn the attention score.

So far, we have decided the visual information for image region representation as well as the semantic information for highlight word in question representation. The last network will determine the output answer for the input question.

$$u = v_{\mathrm{q}'} \circ v_c \tag{9}$$

$$p_a = \text{softmax}(Wu + b) \tag{10}$$

To begin with, the question vector is added into both of the outputs, the semantic representation and the visual representation respectively. Then the element-wise multiplication is used to put the two attentions together. And finally, feed the multiplication into a softmax layer to determine the answer set. The answer with the highest possible probability will be the final answer.

5 Experiment Setting and Results

5.1 Attention network

The result from the attention network is on par with many of the established models even though due to limitations on computing power, we trained on half of the dataset. As seen in fig.4, the evaluation accuracy is able to reach around 57% percent (given that the original performance from [12] is 63.2%). From a more intuitive view, fig.5 shows some of the demos ran on the trained network.

5.2 Two-Channel Visual and Question Model

Considered as a separate and parallel route, we implemented this model from the ground up using PyTorch as described above in fig.2.

5.2.1 Visual Model

We used VGGNet16 to implement the transfer learning model for extracting features from the images. To use the pretrained parameters for this model, we only modified the last fully connected layer to output a 1024 dimension visual embedding.

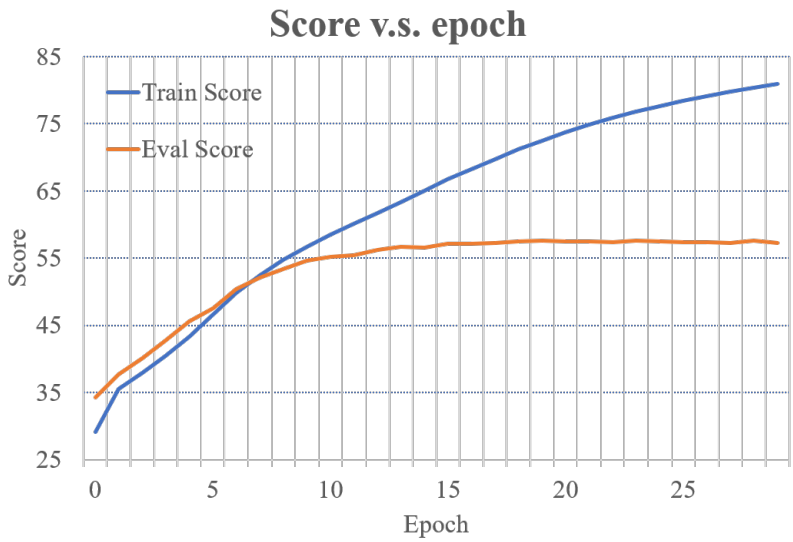

Figure 4: Score v.s. training epoch. The final accuracy equals to scores in percentage format.

5.2.2 Question Model

For the question model, we implemented the long short-term memory (LSTM) model for the question embedding.

The question is first turned to a list of words. Given that we have all the words in the training set of both questions and answers, we can assign them each a number. The input is now turned into a list of numbers from a list of words (strings where models cannot understand). Since we have more than 10,000 words in this collection, it is impossible to onehot the input. So we use an embedding layer to turn the numbers that represent a word into a vector of a certain dimension that represent the same word that can be accepted by LSTM. Alternatively, we also used GLoVE [9] as our word embedding and it will be discussed later.

As described in fig.2, the deep LSTM model input has a dimension of 512 and thus the output of the two layers can be concatenated into a 1024 question embedding. This embedding is then combined with the visual embedding described above for another layer of softmax for the result.

5.2.3 Combined Model

The visual embedding and the question embedding is combined by an elementwise multiplication and followed by a fully connected layer and a softmax layer. The softmax layer has a classification output of size 1000. This poses a potential problem which will be discussed later in the reflection section.

5.2.4 Data Preprocessing and Loading

To make data loading easier, we preprocessed the question data and answer data extracting only the following information from the original dataset: image ID, question ID, question, and the first confident answer. We choose to include only the first confident answer to reduce the data size and make it easier to calculate the loss.

The dataloader grabs a question by the unique question IDs and finds the image by the corresponding image ID. The question is send to the question model and the image is send to the visual model described above. The answer is used to calculate loss.

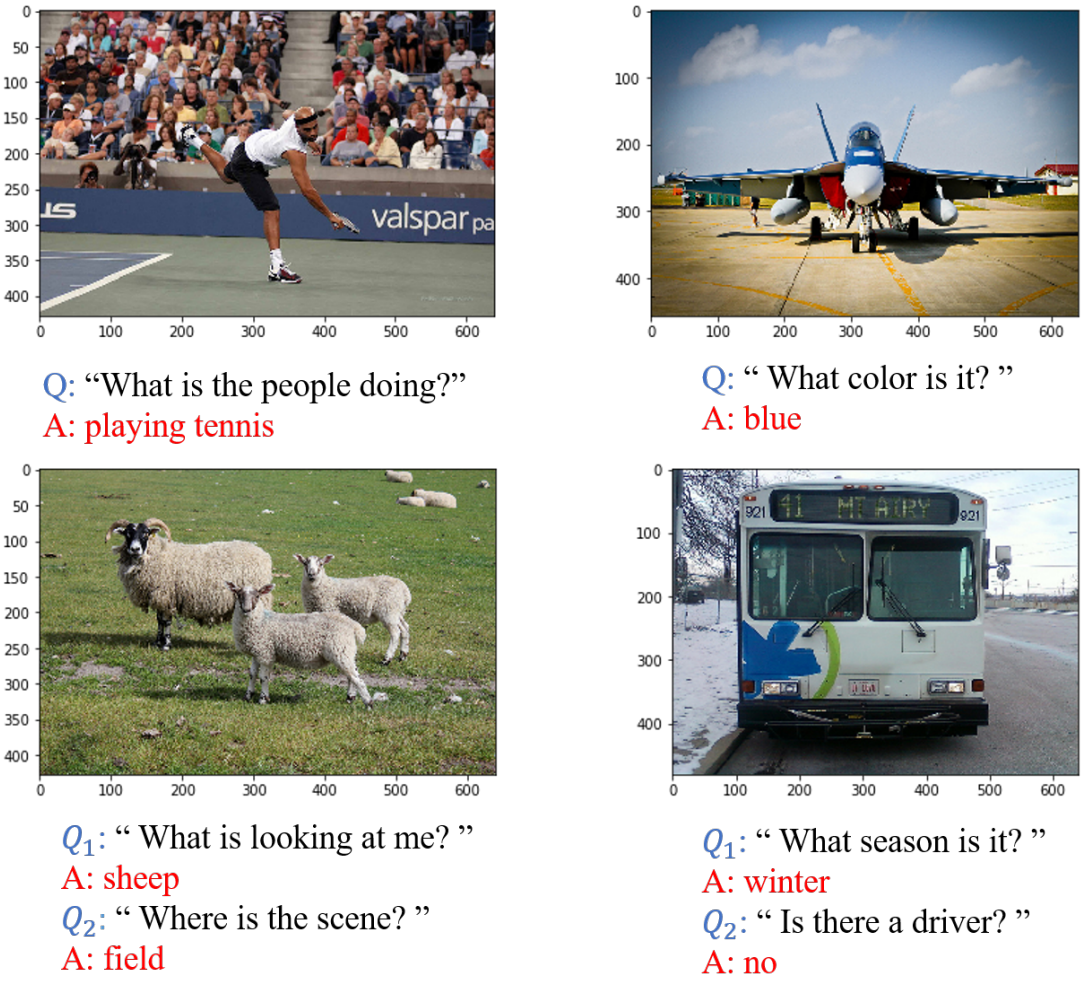

Figure 5: Results of VQA from attention based model

5.2.5 Hyperparameters

The choice decisions for the hyperparameters were really hard and we had to trail and error along the process. The first guess based on knowledge from the referenced literature was as follows: batch size 40; and learning rate: 5×10^{-5}.

To work around the limitations on the computing power, we used a random subset of the training set of size 4096 and our initial question embedding is of dimension 512. These two parameters are modified and tuned throughout the process.

5.2.6 Results

The result of the model with above hyperparameters can be seen as in fig.6 From the graph above, we can clearly see that there is an increasing trend for the training accuracy but the evaluation accuracy is dropping.

5.2.7 Reflection

We started with a 512 dimension question embedding for the question model with a batch size of 40. The model runs but one epoch takes around 1000 seconds (about 16.5 minutes). This means that it will take days for a whole training process (about 150 epochs) and hours for a few epochs for a trend

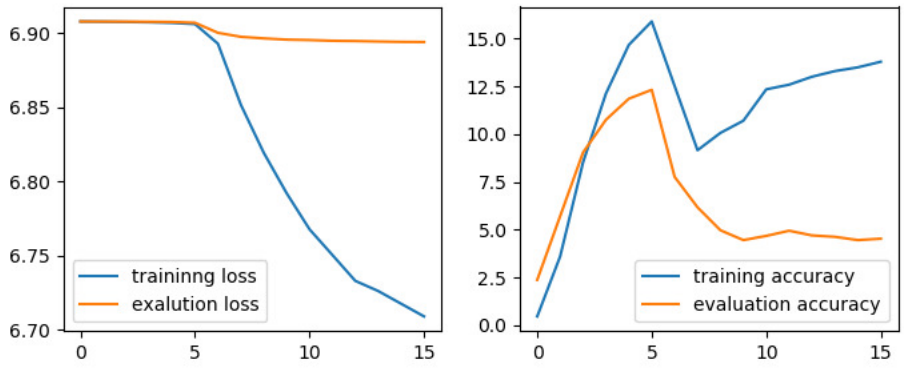

Figure 6: This is the result of the Two-Channel Visual and Question model with 40 batch size and 512 input dimension. The epoch running time is very long, so we weren't able to run for more epochs.

check. With some research we found that the run time for LSTM models scale with the dimension size. So to drop the training time, we need to drop the input dimension size.

After switching to a 100 dimension question embedding, the run time for one epoch drops to around 150 seconds. This is a great improvement on the speed that where we can inspect the training trend of using a plot.

This decision, however, poses a potential problem, which is information loss. With a 100 dimension input, the question embedding is 200. If we combine it with our visual embedding of size 1024, we lose a lot of information from the visual embedding. This is definitely hurting the performance.

While the model was running fine, we noticed significant overfitting problem. And several actions were taken to mitigate it. Dropout at the final fully connected layer was not working for the reason described above, information loss from the visual embedding. Switching to GLoVe question embedding improved the validation error in the first few epochs but it didn't keep the trend. The third action was to allow additional answers for the validation set. As described, we used the first confident answer as our training answer, but we worried that the output answer can be any of the answers in the validation set. So a modification for the validation set dataloader was introduced to allow multiple answer comparisons. There is a slight improvement, but the overfitting is not fully resolved.

Some further problems which may cause the failing validation problem were proposed but unable to solve. The most important one is the smaller dataset. We are using a random 4096 sized (question) subset from a training set of size of more than 443,000 questions and more than 82,000 questions. It is very likely that we are only training over a not very inclusive set and causing the failing validation problem. A second problem is about how we treat our answers (output of our model). The idea for answers was to use onehot and here are limited slots for the output size being 1000. Many answers can fail during training because they would never be treated as a correct answer if its onehot value is larger than 1000.

We propose our solution to the above problem as follows. The structure of the model can stay the same but allow for the 512 dimension for the input question embedding, the full 443,000 training set and the full answer spectrum. Doing so would surely require additional computation power and time but is certain doable.

References

[1]Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel and Yoshua Bengio: "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", 2015;https://arxiv.org/pdf/1502.03044.pdf

[2]Oriol Vinyals, Alexander Toshev, Samy Bengio: "Show and Tell: A Neural Image Caption Generator", 2014; https://arxiv.org/abs/1411.4555v2

[3]Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra: "VQA: Visual Question Answering", 2015; https://arxiv.org/pdf/1505.00468.pdf

[4]VQA; https://visualqa.org/challenge.html

[5]Dongfei Yu, Jianlong Fu, Tao Mei, Yong Rui : "Multi-level Attention Networks for Visual Question Answering"; https://www.microsoft.com/en-us/research/wp-content/uploads/2017/06/Multi-level-Attention-Networks-for-Visual-Question-Answering.pdf

[6]H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu. "Are you talking to a machine? dataset and methods for multilingual image question answering.";

[7]Mateusz Malinowski, Marcus Rohrbach, Mario Fritz, "Ask Your Neurons: A Neural-based Approach to Answering Questions about Images";

[8]Kushal Kafle: "Visual Question Answering: Datasets, Algorithms, and Future Challenges", 2016; https://arxiv.org/pdf/1610.01465.pdf

[9]Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. "GloVe: Global Vectors for Word Representation.";https://nlp.stanford.edu/projects/glove/

[10]VQA download page;https://visualqa.org/download.html

[11]K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition. CoRR", abs/1409.1556, 2014. 8, 9, 10

[12]Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould: "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering", 2017;https://arxiv.org/pdf/1707.07998.pdf

[13]M. Malinowski and M. Fritz. "A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input.";

[14]J. L. Elman. "Finding structure in time." Cognitive science, 14(2):179–211, 1990.

[15]Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu: "Learning Longer Memory in Recurrent Neural Networks", 2014; http://arxiv.org/pdf/1412.7753.pdf

[16]S. Hochreiter and J. Schmidhuber. "Long short-term memory." Neural computation, 9(8):1735–1780, 1997

[17]Karen Simonyan: "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014; https://arxiv.org/pdf/1409.1556.pdf

[18]Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed,Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. "Going Deeper with Convolutions." https://www.cs.unc.edu/ wliu/papers/GoogLeNet.pdf

[19]Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition"

[20]Gao Huang, Zhuang Liu, Laurens van der Maaten: "Densely Connected Convolutional Networks", 2016; https://arxiv.org/pdf/1608.06993.pdf

[21]I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In NIPS, pages 3104–3112, 2014.

[22]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[23]Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick: "Microsoft COCO: Common Objects in Context", 2014; https://arxiv.org/pdf/1405.0312.pdf