SAMS_VQA

Arda Bati A53284500 **Marjan Emadi** A53250275 **So Sasaki** A53267875

Sina Shahsavari A53278804

Abstract

In this project we investigate neural network architectures and optimizers for the Bottom-Up and Top-Down Attention network for Visual Question Answering, which enables attention to be calculated at the level of objects and other salient image regions. As a result of the experiments on the VQA v2.0 dataset, our implementation achieved 63.61% validation accuracy, whereas the original implementation reached 63.15%. We pointed out that GRU unit in the question embedding outperforms LSTM by 0.36%. In addition, we quantitatively analyzed the impact of SGD and Adamax optimizers and their hyper-parameters. Our failure-example analysis suggests that the restricted vocabulary for answering is a limitation of these VQA methods. The source codes and additional information are on our Github page. [1]

1 Introduction

Visual Question Answering (VQA) Challenge [1] has attracted much interest since its inception in 2015. The system of the VQA challenge takes input images and free-form, open-ended questions that are of natural language format. The aim of the challenge is to produce accurate, open-ended answers to the questions. The challenge combines the disciplines of Natural Language Processing (NLP) and Computer Vision. Deep Neural Networks typically form the backbone of the system. The NLP role is usually filled by LSTMs or other RNN variants, and the Computer Vision role is filled by popular image classifiers/feature extractors such as VGGNet or ResNet.

The VQA Challenge has the advantage of being more complex than simpler tasks such as classification or image captioning. Image captioning and classification can be done by a relatively less understanding of the data. VQA Challenge, however, requires a more fundamental understanding of the underlying information. Even if the answer is just "yes" or "no", understanding and evaluating the questions in VQA is a quite complex task. Consider questions such as "Can you park here?" or "Is something under the sink broken?", which requires multiple levels of understanding from the machine. These types of questions not only require object recognition and detection, but also more abstract, human level reasoning. In fact, during the preparation of the dataset, human subjects were asked to prepare questions that a smart machine would have trouble answering. They were advised to avoid simple questions such as "What is the color of the car" or "Is there a dog" which could be solved by a simpler understanding. This is where the main complexity of VQA lies, and why it has garnered so much interest.

Another advantage of VQA is that it has a simple evaluation metric which can be automated. This contrasts with image captioning for which automatic captioning is difficult to achieve. Also, the answers to the questions are usually one to three words, usually yes / no answers. This way the complexity of the system is in the understanding of the questions and images, and combining this understanding, instead of the answers themselves. This also decreases the variability of the answers given by the human subjects, making the evaluation metric more robust.

[1] https://github.com/s0sasaki/ece285_vqa

For the purposes of our project, we elected to work on the Bottom-Up and Top-Down Attention model [2]. Our aim is to assess the performance of the model with different network characteristics. We analyze the effect on training and evaluation performance. We point out which architecture/parameters are the best for the given model and explain the possible reasons. Our focus is on final accuracy, complexity, time and memory requirements, and the difference caused by architecture/parameter changes between train and evaluation performance. Because of our limited resources, we were not able to train each model to the full extent required. However, as the model mostly converges after a short time, this is not a major issue. We also worked on the Baseline Model provided by the VQA website [1] to better understand the baseline structure and its possible shortcomings. In the next section, we will first give information about the baseline VQA model, then the main model that we used.

2 Description of the methods

2.1 Baseline Model

In general for the Baseline VQA, we need to develop 2 channels [1]:

1.Image channel an embedding for the visual part, and extracts the features out of them. In the baseline model we used method I. In this method we used VGGNet and the activations from the last hidden layer of VGGNet are used as 4096-dim image embedding.

2.Question Channel provides an embedding for the questions. We experiment with the LSTM Q. An LSTM with one hidden layer is used to obtain 1024-dim embedding for the question. The embedding obtained from the LSTM is a concatenation of last cell state and last hidden state representations (each being 512-dim) from the hidden layer of the LSTM. Each question word is encoded with 300-dim embedding by a fully-connected layer + tanh non-linearity which is then fed to the LSTM. The input vocabulary to the embedding layer consists of all the question words seen in the training dataset.

At last, both the final question and image features are transformed into the same space and fused via element-wise multiplication. This will be passed through a fully connected layer followed by a softmax layer over 1000 most frequent answers as possible outputs.

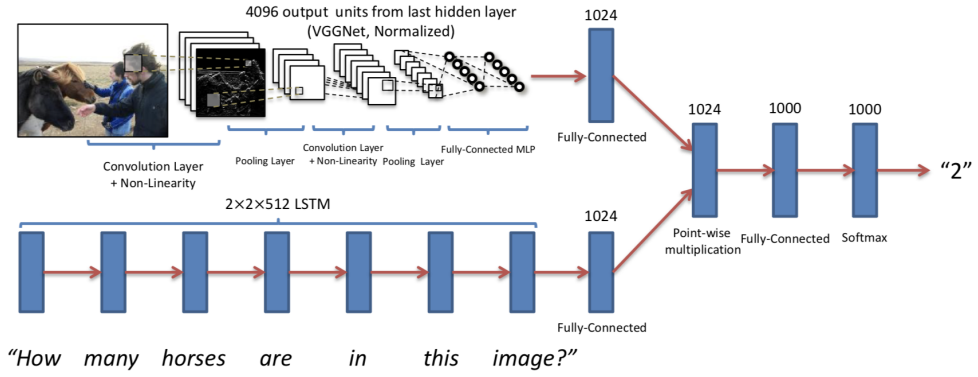


Figure 1: Baseline model of the VQA using LSTM Q and I [1]

2.2 Bottom-Up and Top-Down Attention Mechanism

In this section, we will explain the main method we have based on our project. We start by explaining the relevant concept: Visual Attention in Networks [3], Faster R-CNN [4], and the Bottom Up Attention Mechanism [2]. In the Methodology section we explain how the Bottom Up and Top Down attention models are combined, and details of the overall method. Finally, we will mention the specific changes that were made to the method in the code of this project.

2.2.1 Visual Attention in Networks

The mechanism of attention is used in Neural Networks to focus on specific parts or subsets of the input, giving less priority to the rest. Visual attention is a subset of attention, in which the network focuses on specific areas of an image. Visual attention can be considered in two main ways, Hard and Soft Attention. Briefly, Hard Attention multiplies features of the image by a mask composed of 0s and 1s. Parts of the image are either included with full intensity or completely excluded. This is similar to cropping the image, where the cropped part(s) can be of any shape. Soft Attention, on the other hand, uses a mask of values between 0 and 1 (or possibly other ranges, but not just 0 or 1). In this case, the image is not actually cropped, but the intensity of different parts is changed. An example of Soft Attention can be seen below (in the context of image captioning, but it can also be considered for VQA).

Figure 2: Examples of Soft Visual Attention applied to captioning. [3]

For both types of Attention, the ability to focus on different parts of the image is a very good tool for Visual Question Answering. Rather than focusing on each part of the image, the network can focus on the section that is related to the question. For example, consider the first image above. In the case of a question such as "What is being thrown?" the given attention distribution in the image (focusing on the frisbee) would greatly help the network in giving the correct answer.

2.2.2 Faster R-CNN

Bottom-Up Attention Mechanism which will be described below is based on the Faster R-CNN method. Faster R-CNN is in turn based on R-CNN [5] , Fast R-CNN [6]. We will first briefly explain the R-CNN methods, and then describe the overall Bottom-Up Attention Mechanism in the next section.

R-CNN This method is developed for object detection in images. Rather than just classifying an image as a certain object, it detects single or multiple relevant objects in the image. These images are specified by drawing a bounding box around them. The standard type of classifier CNN networks is not able to do this in a straightforward manner. In practice, many different sections of the image could be fed to the network separately, then the classification result would be the detected image. However, this approach would not be practical, as we can't know how many objects there are, their size and aspect ration, and where they are in the image.

R-CNN overcomes this problem by using the "Selective Search" method [7], which is a fixed algorithm that is not learned. This method extracts a certain number of (default 2000) proposed regions from the image. Then the network can just focus on these regions, decide which ones to keep and classify. The main disadvantage of R-CNN is that it is quite slow because, for every single image, it needs to feed forward 2000 candidates regions through the network. Also, the fixed nature of the Selective Search means that the candidate region selection may not work with every dataset.

Fast R-CNN This method is an improvement over R-CNN to make it faster. It has overall the same structure with R-CNN but one main difference. Instead of the 2000 proposal regions, only the original image is fed to the CNN. The proposed regions are formed from the feature maps outputted

by the CNN. Again, Selective Search is used in this step. Fast R-CNN greatly increases the speed as it reduces the required number of convolutions. However, it still uses the Selective Search method which is somewhat time intensive and cannot be learned from the dataset itself.

Faster R-CNN This method is a further upgrade over Fast R-CNN. Faster R-CNN combines Fast R-CNN with a Region Proposal Network (RPN) [4]. The increased speed comes from replacing the Selective Search algorithm by the RPN. The outputs of the initial CNN layers (mentioned in Fast R-CNN) are fed to the RPN. RPN learns how to make good proposals from the dataset, and it is quite faster compared to Selective Search. After the RPN, the rest of the structure is very similar to the rest of the Fast R-CNN architecture. The final output is a classification of each region proposal, including corrections to their bounding box coordinates.

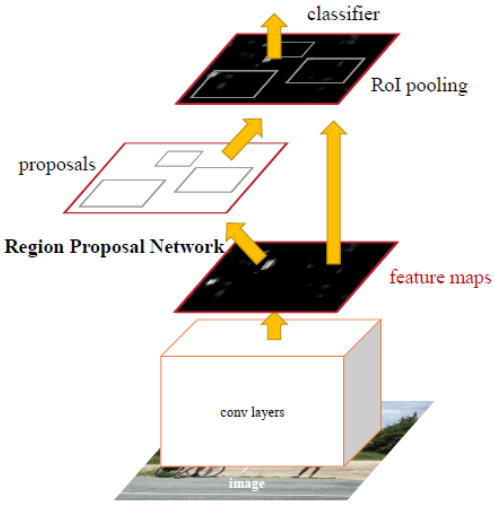


Figure 3: Overview of the Faster R-CNN method [4]

2.2.3 Bottom Up Attention Mechanism

The Bottom-Up Attention Mechanism is used to propose relevant sections of the image as attention sources (region proposals). This process is independent of the corresponding question of the image. These relevant sections of the image are then fed to the Top Down Mechanism and the rest of the network, which will be explained later. Faster R-CNN is used as the main component of the Bottom Up Attention Mechanism. It can be considered as a Hard Attention Model as it selects a specific set of bounding boxes, from the total number of possible bounding box configurations. For the generation

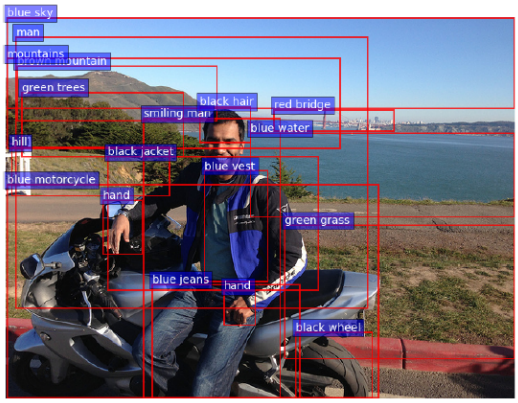


Figure 4: Example outputs from the Bottom Up Attention Model using Faster R-CNN. [2]

of the model, firstly the Faster R-CNN is initialized with the ResNet-101 Network that is pre-trained using the ImageNet Dataset [8]. Then, further training is done on the Visual Genome Dataset [9]. The network at this stage learns and outputs both object classes (dog, tree, car, etc.) and attribute classes (small, large, red, etc.) to further improve feature representations.

2.2.4 Methodology

There are already several Top Down methods that are used for image captioning and VQA as stated in [2]. The novelty here comes from the combination of Top Down and Bottom Up attention mechanisms. As explained before, the Bottom Up Model isolates proposal regions from the image that possibly contain objects. Then, the Top Down Attention mechanism assigns weights to each of these regions, considering the question asked about the image.

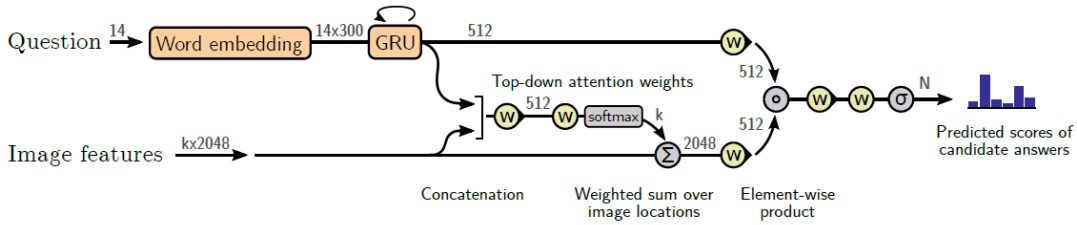


Figure 5: Overall structure of the Top Down and Bottom Up Attention Mechanism. The yellow elements represent parts with learned parameter. Gray elements represent operations that do not include learning. Gray numbers represent the feature sizes of corresponding image features or question encoding. [2]

Similar to the other methods that are used on the VQA Dataset [10], the architecture has two main components. These are the processing of the question and the image. For the question, firstly each word the question contains is transformed to 300 dimension word embeddings. The word embedding is initialized by pre-trained GloVe vectors introduced in [11]. After this step, the embedded words are put through a Gated Recurrent Unit (GRU) [12]. Each question is represented by a hidden state of the GRU (with dimension is 512). The encoded question is fed both to the Top Down (Soft) Attention mechanism and the joint multimodal embedding of the question and the image.

The image features coming from the Bottom Up Attention model are of size $k \times 2048$ where k is the number of proposed regions. The Top Down Attention model decides on the normalized attention weight for each region. For this decision, it uses a concatenation of the $k \times 2048$ image features and 1×512 question embedding. The concatenated input is passed through two layers and the output layer contains a softmax function. The softmax function's output is the normalized k attention weights. Then, according to these attention weights, a weighted sum is applied to the k regions (of size 1×2048 each), forming a final vector of size 1×2048. This vector is passed through a learned layer reducing it to size 512.

The final size of both the image and question components is 512 , which enables element-wise product leading to the multimodal embedding mentioned before. At the final stage, the embedded result is passed through two hidden layers and the final output function (nonlinear activation). The output is the predicted scores of the candidate answers set. From this set, the candidate answers with the highest scores are chosen as outputted answers.

In the original methodology of the paper, the non-linearities that are learned in the network (the yellow elements represented in Figure 7) use gated hyperbolic tangent activations [13]. These were shown to have advantages over the more common ReLU and tanh activations in the empirical context [1]. However as explained in the next section, the implementation that we based our project upon takes a different approach. Also, in the original model the adaptive gradient descent function AdaDelta [14] is used along with early stopping.

2.2.5 Modifications

There are deviations from the methodology in the codebase [15] that we based our project upon. Most of these changes are not major changes from the overall structure, but usually small implementation differences. The modifications fall into four categories as below.

Feature Extraction and Pre-training

(1) Visual Genome dataset is not used for the pre-training of the Bottom Up Attention Model. The original pre-training complicates the whole process. However, the final accuracy results do not seem to be affected by this pre-training, hence it was removed. (2) Keeping the number of proposed regions in the RPN fixed at (K=36). This decision comes from the original paper which mentions that using the top 36 proposed regions for each image worked quite well for their purposes.

Network Changes

(1) Instead of a two-stream classifier which classifies object and attribute classes, a single stream classifier is used, which does not have pre-training. (2) Instead of the gated tanh activations, easier ReLU activations are used. (3) The number of neurons in the layers is doubled throughout the architecture. This alleviates the impact of some of the modifications.

Training Changes

(1) Dropout is added during training, which makes the network better resist overfitting. (2) Batch Normalization is used during training. (3) Instead of the AdaDelta optimizer Adamax [16] is used as the baseline. However, we try additional optimizers in our experiments. (4) As ReLU activations are being used, Gradient Clipping is added to prevent exploding gradients.

Modified Attention

(1) The concentration based attention in the top-down attention model is changed by a projection based model. This new attention model is a modified version of the one described in the paper [17]. This model focuses on the relationships between the attended objects / regions.

3 Experimental setting and Results

3.1 Dataset Explanation

In this project, we used the VQA v2.0 dataset [10]. This dataset minimizes the effectiveness of learning dataset priors by balancing the answers to each question. It contains $82K$ training, $40K$ validation and $81K$ test images from the MS COCO dataset.[18]

MS COCO dataset is designed with several objects and rich contextual information to give visual complexity to the images. This helps in making more interesting, diverse and comprehensive questions [1]. Simple questions like asking the color of an object may require a low-level computer vision knowledge, while we also want questions that require commonsense knowledge about the scene and require the image to correctly answer to some extent. With a wide variety of question types and difficulty, we are able to measure the continual progress of both visual understanding and commonsense reasoning. The dataset contains $443K$ questions for the train, $214K$ questions for validation and $447K$ test set that come in 2 formats of multiple choice and open-ended questions. We focus on the open-ended task.

In open-ended questions, for every image, 3 free-form natural-language questions with 10 concise open-ended answers are collected along with their confidence level.

3.2 Experimental Setup

As mentioned in section 2.2.3 we used a bottom-up mechanism with a pretrained Resnet101 network to extract feature vectors of VQA dataset's images. The output of the image sub-network is a spatial feature tensor and a bounding boxes tensor for each image in the training and validation set. The spatial feature is a 36×2048 tensor in which 2048 is the dimension of feature vectors and 36 is the number of detected objects. The bounding boxes contain 4 number for each of 36 boxes that indicate the position of the corresponding box in the original image.

In the question embedding sub-network, the model performs standard text preprocessing and tokenization. For decreasing computational complexity questions are trimmed to a maximum of 14 words. In this network, we will compare using LSTM or GRU units while the number of hidden units in each layer is fixed to 1024 in both cases. In addition, in preprocessing tools there is a tool that creates a dictionary of correct answers in the training set that appears more than 8 times. Any

member of this dictionary which includes 3129 words or combination of words will be a candidate answer for questions.

To evaluate answer quality, the standard VQA metric [1] has been used to compute the accuracy percentage. In this metric we use:

$$\min(\frac{\#human\,labels\,that\,match\,that\,answer}{3}, 1)$$

this metric basically gives full credit to the answer when three or more of the ten human labels match the answer and gives partial credit if there are fewer matches.

The original implementation Adamax optimizer with default learning rate (2×10^{-3}) and without weight decay has been used for training, however, in our experiment, we tweaked the learning rate and weight decay to avoid overfitting and increase the efficiency of the model. Furthermore, we exchanged Adamax with other optimizers such as SGD and Ada-delta to compare their performance. Gradient clipping threshold is another parameter we have changed to see its impact on model performance.

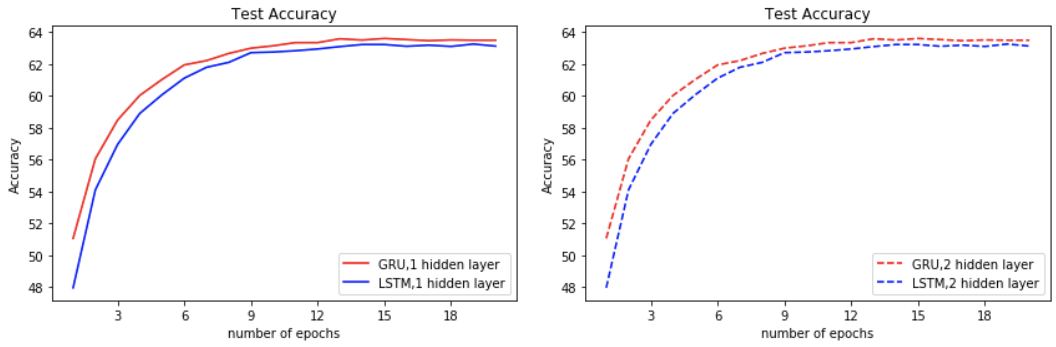
The network with the mentioned setting takes about 570 seconds per epoch for training on DSMLP server.

3.3 Results

The model introduced in [2] uses a network with one hidden layer of GRU units in question embedding sub-network. Our contribution is comparing the performance of the model while using LSTM units instead of GRU units as well as the impact of the number of hidden layers in the final accuracy of the model. Thus we modified the question embedding part of the model in order to exploit the LSTM units. Experimental results in Table 1 and Figure.6a show that GRU units will slightly outperform LSTM units in this task. Furthermore, in the second part of Table 1 as well as Figure. 6b, we can observe that networks with one and two hidden layers have similar performance. Thus increasing number of hidden layers in question embedding proves ineffective on the performance of the model. Hence, the best performance according to test accuracy is 63.61% for GRU units.

Table 1: Accuracy of using LSTM VS GRU

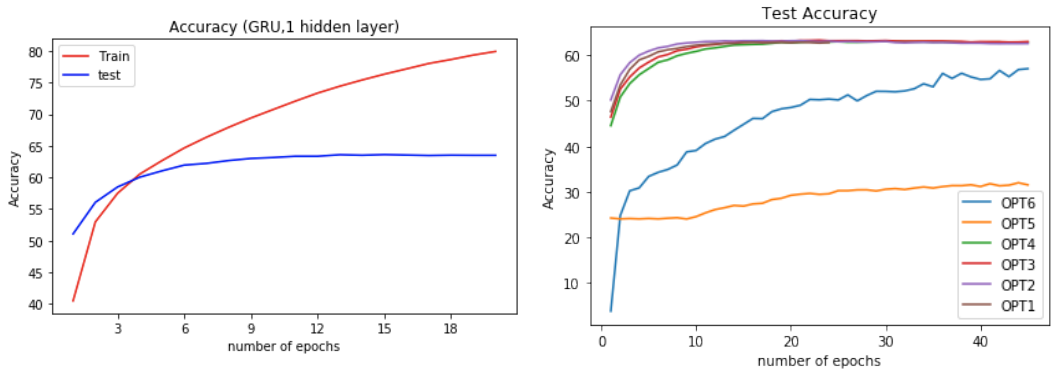
	1 hidden layer		2 hidden layer	
	GRU	LSTM	GRU	LSTM
train	**82.66%**	79.39%	**82.66%**	79.39%
test	**63.61%**	63.25%	**63.61%**	63.26%



(a) **(b)**

Figure 6: Comparing test accuracy of a network with LSTM units (red) and a network with GRU units (blue). a) one hidden layer b) two hidden layer

The increasing trend of training and validation accuracy in Figure .7a shows that with current parameters the network suffers from over-fitting. Therefore six different settings of optimizers have been tested to alleviate the over-fitting problem and as a result improve the model performance. We present the optimizer type and corresponding parameters for these settings in Table 3. By investigating the validation accuracy for these optimizers which are shown in Figure. 7b, and mentioned in Table



(a) **(b)**

Figure 7: a) train and test accuracy of the model while using default Adamax optimizer with default parameters, b) test accuracy of using 6 different setting for optimizer

3, we can observe that SGD optimizer without mini-batches does not perform well for the VQA task. Adadelta's convergence time is longer than desired. In addition, Adamax has the best performance among the different optimizers. Tweaking its parameters such as learning rate and weight decay mostly affect the convergence time and have a small impact on final test accuracy. The best validation accuracy is 63.61% which is achieved while using the third optimizer.

Table 2: Optimizers and parameters

	optimizer	learning rate	weight decay	momentum
OPT1	Adamax	10^{-2}	10^{-9}	-
OPT2	Adamax	4×10^{-3}	10^{-8}	-
OPT3	Adamax	6×10^{-4}	10^{-7}	-
OPT4	Adamax	2×10^{-3}	0	-
OPT5	SGD	2×10^{-3}	10^{-9}	0.9
OPT6	AdaDelta	1	10^{-8}	-

Table 3: Accuracy of different optimizers

	OPT1	OPT2	OPT3	OPT4	OPT5	OPT6
train	78.60%	85.91%	**87.37%**	82.66%	31.32%	59.44%
test	62.87%	63.20%	63.32%	**63.61%**	31.99%	57.06%

Comparison with reference paper:
The best result on validation set reported in the paper [19] is 63.15%. The reported result without training on extra data from visual genome is 62.48%, the result using only 36 objects per image is 62.82%, the result using two steam classifier but not pretrained is 62.28% and the result using ReLU is 61.63%. These numbers are cited from the Table 1 of [19]. Considering modifications that are mentioned in section 2.2.5 and hyper parameter adjusting which have been presented in Table 3, our network has reached 63.61% accuracy which is better that previous result.

3.4 Successful and failed examples

As seen in the following images, answers which are more related to numbers and colors, with less detailed questions are more likely to be right. These types of answers are frequent among training data and are more likely to be successful.

On the other hand, failure examples, are related to questions with answers that do not belong to the dictionary and are less frequent. These types of questions would need a trained net that is more dedicated to extracting the related features. For example, the motorcycle's plate number could be recognized by a network that is trained for recognizing sequences of digits. Another failure example is the question of gym shoes' color. The gym shoes are somewhat hidden in the image, which makes

it harder to recognize the content. Even human observers would have a harder time giving the correct answer to such a question compared to others.

(a) What color is the lamp?
blue
blue
[1.]

(b) How many animals are in the picture?
two
two
[1.]

(c) Is this a motorcycle or bike?
motorcycle
bike, motorcycle
[0.4 0.6]

(d) Where is the dog laying?
sidewalk
outside, street, yes
sidewalk, ground
[0.1 0.1 0.1 0.6 0.1]

(e) What color are the gym shoes?
red
white, white and black
[0.9 0.1]

(f) What is the plate number?
yes
sv 6260, sv-6260, 6260
[0.1 0.7 0.2]

Figure 8: Examples of questions (black), Answers given using VQA (Blue), People's answers (Red), Confidence levels of people's answers (green)

4 Discussion

By investigating different hyper-parameters of network architecture and training we learned the following main points: 1) Although some methods are necessary for successful training, such as gradient clipping, tweaking their parameters does not necessarily have a significant impact on network performance. 2) For an RNN type network (LSTM or GRU), in the context of VQA dataset and the architecture we use, one hidden layer is enough. Increasing number of the hidden layers does not bring benefit. 3) Decreasing learning rate causes the network take more training time, but it cannot help to address the overfitting problem in this task. The reason for this happening might be that overfitting is tied to the architecture's limitations in generalizing, especially for harder examples.

One of the shortcomings of the VQA model we used is caused by the limited answer vocabulary that it uses. This is in fact the case for the baseline model and other proposed methods. Even though limited vocabulary contains most common words / phrases for the dataset, it cannot cover every possible answer. Some questions need more specific, dedicated feature extraction, such as the license plate example mentioned before.

To increase accuracy, we could add more complexity to the model, or add some external helpers. For example, using the MUTAN fusion method [20] would make the multimodal embedding of images and questions more complex and detailed. This could increase the accuracy for questions that have a more complex relation between the question an the image. Another method would be to add relational facts as a prior source of information to the model. For example, if the model knows relationships such as "sky is blue", "snow on ground" etc., it would not have to learn everything itself. These two methods could also be used together to complement each other.

Our challenges in this project mostly stemmed from being exposed to many concepts that are new to us, and understanding why they are strong or weak with the VQA dataset. The long training times

and limited resources required us to isolate the most significant parameters to modify. Preparing the demos and the github page, and making modifications to the code lead to many implementation challenges which further improved our knowledge. Finally, some of our tests yielded similar results, which forced us to rethink our approach and find where the real performance difference lies. We believe we that earned valuable practical and theoretical experience by overcoming such challenges of the project.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. *In ICCV, 2015.*

[2] Anderson, Peter, et al. "Bottom-up and top-down attention for image captioning and visual question answering." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.*

[3] Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *In International conference on machine learning, 2015.*

[4] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *In Advances in neural information processing systems. 2015.*

[5] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.*

[6] Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision. 2015.*

[7] Uijlings, Jasper RR, et al. "Selective search for object recognition." *International journal of computer vision 104.2 (2013): 154-171.*

[8] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *International journal of computer vision 115.3 (2015): 211-252.*

[9] Krishna, Ranjay, et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." *International Journal of Computer Vision 123.1 (2017): 32-73.*

[10] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *In CVPR, 2017.*

[11] 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.*

[12] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078 (2014).*

[13] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. "Language modeling with gated convolutional networks." *arXiv preprint arXiv:1612.08083, 2016.*

[14] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701 (2012).*

[15] https://github.com/hengyuan-hu/bottom-up-attention-vqa

[16] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980 (2014).*

[17] Hu, Ronghang, et al. "Modeling relationships in referential expressions with compositional modular networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.*

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. *In ECCV, 2014.*

[19] Teney, Damien, et al. "Tips and tricks for visual question answering: Learnings from the 2017 challenge." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.*

[20] Ben-Younes, Hedi, et al. "Mutan: Multimodal tucker fusion for visual question answering." *Proceedings of the IEEE international conference on computer vision. 2017.*