Visual Question Answering — Attention and Fusion based approaches

A Preprint

Digbalay Bose
Department of ECE
University of Southern California
Los Angeles,CA
dbose@usc.edu

Nithin Rao Koluguri
Department of ECE
University of Southern California
Los Angeles,CA
koluguri@usc.edu

Aditya Mate
Department of CS
University of Southern California
Los Angeles,CA
amate@usc.edu

Namrata Tammareddy
Department of CS
University of Southern California
Los Angeles,CA
tammared@usc.edu

October 27, 2021

Abstract

Visual Question Answering is one of the challenging AI tasks, which involves both reasoning about the image content and understanding the question to provide open ended natural language answer. Here in this work, we explore different deep neural network based models for generating natural language based answers. We aim to develop baseline models along with modifications of the current state of the art VQA setups and evaluate its performance on the standard VQA dataset.

***K*eywords** Visual Question Answering · Deep Learning · Bilinear Pooling · Classification

1 Introduction

Recent years have witnessed huge leaps in the areas of computer vision and natural language processing due to the usage of deep neural network models. These advances have prompted researchers to look at challenging problems lying in the intersection of the above mentioned areas like image captioning, video description, visual storytelling etc. One such challenging problem which has attracted the attention of the researchers is visual question answering. The problem involves giving an image and natural language question as inputs and expecting a natural language answer as output.

General question answering approaches in NLP involve understanding the contents of both question and answer. For example, given a question "how many cars in the garage?", a NLP system must understand that its a counting question and extract the object ("cars") to count and associate with the answer(textual answer). In case of VQA, the challenge lies in inferring the context as well as the answer from the contents of the associated input image. When the above question is asked in case of VQA, a system must recognize the garage in the image (scene recognition) along with the cars (object detection + classification) to arrive at a proper answer.

2 Pipeline of VQA system

The VQA system consist of three major segments:

- ***Feature Extraction:*** The feature extraction block acts separately on the input question and image to extract the relevant features. A CNN is usually used for extracting the image feature after removing the last fully

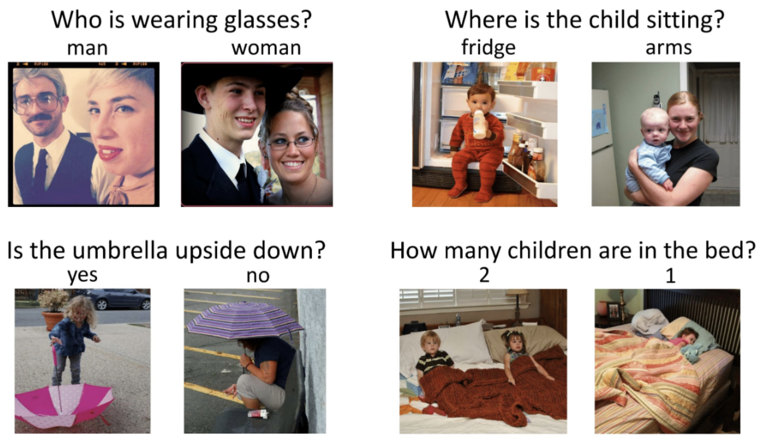

Who is wearing glasses?
man
woman
Where is the child sitting?
fridge
arms
Is the umbrella upside down?
yes
no
How many children are in the bed?
2
1

Figure 1: Sample examples from the VQA dataset (The image, associated question and the answers are listed)

connected layer for classification. This is often replaced by **pooled feature map outputs** from the last convolutional layer in the CNN. The question feature is usually extracted by passing the **question tokens** (details of the tokenization process given in the dataset preparation section) through an embedding layer followed by **LSTM** or **GRU**. The final hidden state from the LSTM or GRU is taken as the question feature representation.

- ***Feature Fusion:*** The key element of the entire VQA pipeline is the fusion block. The fusion block takes both the question and image features as inputs and provides the **fused multi-modal feature** as an output. Fusion is done using different methods like **elementwise multiplication of image and question feature vectors** and **bilinear pooling**. The elementwise multiplication method is constrained by the fact that the dimensions of image and question vectors must be the same. In order to fuse feature vectors from image and text modalities having different dimensions, **bilinear pooling** is often used to obtain a fused embedding. Suppose the image vector is x and the question feature vector is y, then the fused output z is computed as follows:

$$z = x \odot y \tag{1}$$

$$z_i = x^t W_i y \tag{2}$$

The first equation shows elementwise multiplication between two vectors x and y of same dimensions and second equation shows bilinear pooling between x and y of different dimensions. For example, if the dimensions of x are 60 and 40, then W_i has dimensions of 60×40. z_i is the ith element of the fused vector z.

Other potential techniques for feature fusion can be concatenation, attention-based pooling, Bayesian-based methods, compositional approaches etc.

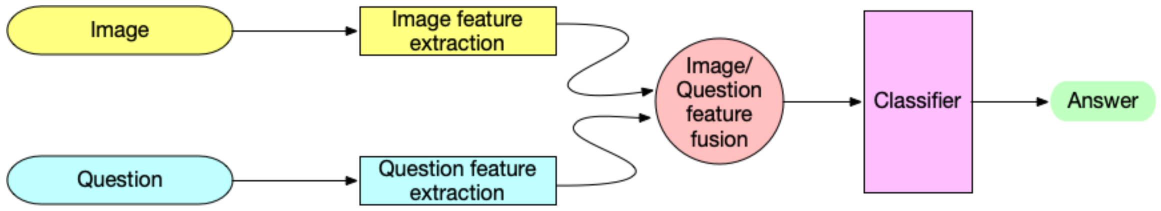


Figure 2: Overall pipeline for the VQA system

- ***Classifier:*** The classifier is the last block in the pipeline which takes the fused feature vector as input and tries to classify it into the given **answer** labels. The single word answers for the questions are converted into labels and the entire VQA is thus cast as a **classification problem.** While training a deep neural network for VQA, usually the problem is cast as a multi-label problem. This is because a single open-ended question can have multiple correct answers(labels). For example, if a question is asked: **"What is the color of the shirt?"**, then there can be multiple correct answers: **"red"," dark red" etc.**

3 Methods Explored

3.1 Baseline Method

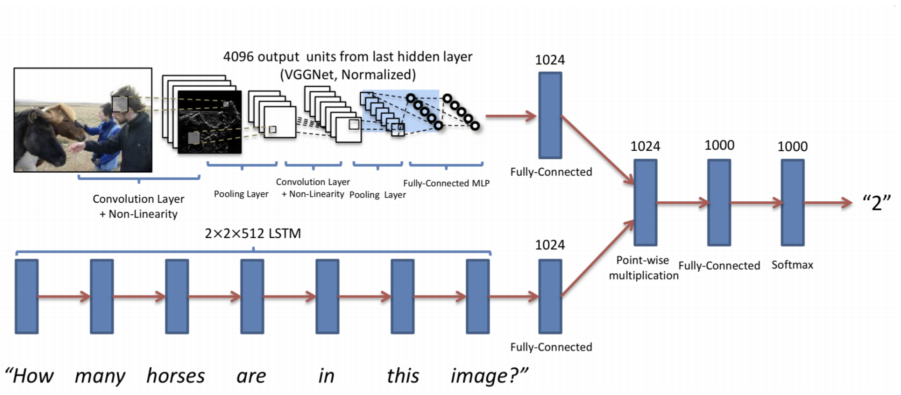


Figure 3: Baseline method based on multiplicative fusion of question and image feature vector

We considered the baseline method discussed in [2], where the image feature extraction is done using **VGG network**. The question was passed through a **2 layer LSTM** and the final hidden state is considered as the question feature vector. The question feature vector was combined with the image feature vector using point wise multiplication. The fused feature vector was then passed through fully connected layers to obtain the final answer class.

3.2 Question Guided Attention Mechanism

We implemented the model given in [10] from scratch where a question guided attention mechanism was used for combining the image features from different regions. As shown in the figure, the question feature vector q was concatenated with the image feature vector from the i^{th} region v_i and passed through a non-linear layer (f_a) followed by a linear layer whose weights are denoted by w_a. Then, the attention weights were normalized over all locations using a softmax function to obtain the final weights values (α) which sum to 1. The combined image feature vector(using attention weights) for all the locations is given by $\hat{v}$.

$$a_i = w_a f_a\left([v_i, q]\right) \tag{3}$$

$$\alpha = \mathrm{softmax}(a) \tag{4}$$

$$\hat{v} = \sum_{i=1}^{K} \alpha_i v_i \tag{5}$$

Since the combined image vector $\hat{v}$ is to be combined with the question vector q using multiplicative fusion, both the feature vectors are passed through appropriate **FC** layers to obtain same sizes for the embeddings. For example, in our experiments, the combined image feature vector had a dimension of 2048 and the question feature vector had a dimension of 512 or 1024 or 1280. Hence before combining the two feature vectors, their respective dimensions should be made equal. After the multiplicative fusion of the question and image vectors, a classifier composed of fully connected layers was used for final prediction. Instead of relying on the two-stream (text+image) classifier for final answer prediction as used in [10] , we used two fully connected layers for final answer classification.

3.3 Question guided attention mechanism + Multi modal Factorized Bilinear Pooling

Bilinear pooling[11] as mentioned in the pipeline for the VQA system is another way of fusing the feature vectors from different modalities. The motivation of using bilinear pooling in place of multiplicative fusion of feature vectors is that it removes the restriction of using same dimensions for the text and image feature vectors. It also captures richer pairwise interactions among the feature dimensions in the different modalities.

Bilinear pooling which is denoted by $z_i = x^i W_i y$ involves learning separate matrix W_i for each output entry z_i. This results in high computational cost and overfitting. In order to circumvent this problem, the matrix W_i is written as the low rank factorization of two matrices U_i and V_i. The bilinear pooling fusion can then be written as follows:

$$z_i = x^T U_i V_i^T y = \sum_{d=1}^{k} x^T u_d v_d^T y = 1^T \left(U_i^T x \circ V_i^T y \right) \tag{6}$$

If the fused vector $z \epsilon R^o$, then the matrices U_i and V_i can be grouped together to form $\tilde{U} \epsilon R^{m \times ko}$ and $\tilde{V} \epsilon R^{n \times ko}$ respectively. Here k is the latent dimensionality of the factorized matrices U_i and V_i. Thus the above entire operation can be rewritten as:

$$z = \text{ SumPooling } \left(\tilde{U}^T x \circ \tilde{V}^T y, k \right) \tag{7}$$

Here $SumPooling(z, k)$ means summation over a one-dimensional window of k. Here $SumPooling(z, k)$ for $z \epsilon R^{ko}$ results in an output vector of $z \epsilon R^o$. The entire operation is considered as a part of MFB(Multimodal factorized bilinear pooling) module which is displayed as below:

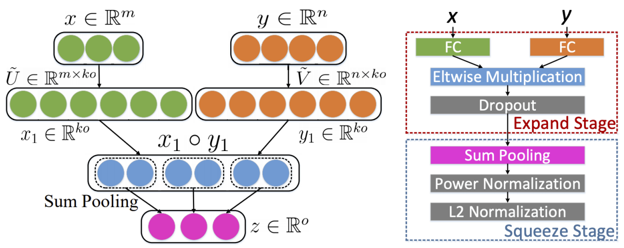


Figure 4: Multimodal factorized bilinear pooling model and the MFB module

Since the MFB block offers easy implementation of the bilinear pooling of text and image feature vectors, we used it instead of the multiplicative fusion in the question guided attention approach as discussed above. We used two MFB modules instead of one and finally concatenate their output representations to obtain the final fused representation. The fused representation was passed again to FC layers based classifier.

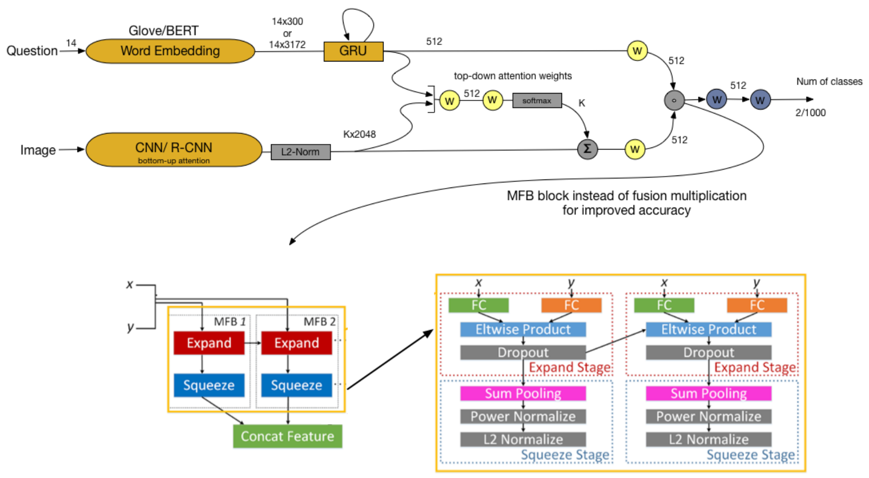
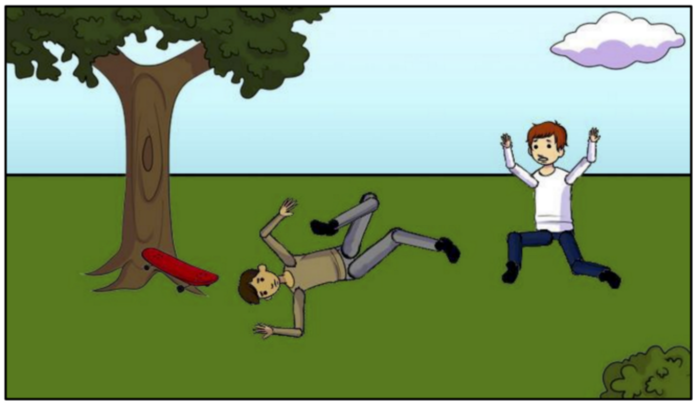


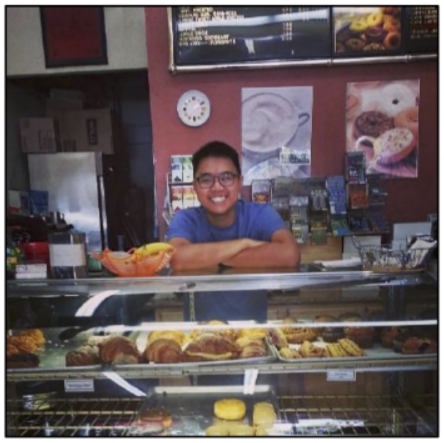
Figure 5: Question guided attention for the image regions followed by MFB based fusion followed by fully connected layers for classification

4 Dataset

The dataset considered for evaluating the proposed VQA algorithms is given in https://visualqa.org/. The dataset contains open-ended questions about images. There are primarily two types of images upon which the dataset is further subdivided: **Abstract and Real Images.**



(a) Cartoon-like abstract image where the question asked: Do you think the boy on the ground has broken legs?



(b) Real image where the question asked: What kind of store is this?

Figure 6: Examples of abstract image/question pair along with real image/question

Our experiments were carried out on the real images in the VQA dataset. The details are listed as follows:

Mode	Training	Validation	Testing
Images	82783	40504	81434
Questions	443757	214354	447793
Answers	443757	214354	-

Table 1: Table showing the number of questions and images available as a part of VQA dataset

Some salient features of the dataset:

- The images were taken from **MS COCO dataset** `http://cocodataset.org/`because of its diversity in terms of objects and rich contextual information. The train/val/test splits for the images are the same as the **MS COCO** dataset.
- On average, three questions from unique workers were gathered for each image. While a worker was writing a question, he/she was shown the previous questions in order to increase diversity.
- The answers considered were a **single word** in nature. Examples include yes/no, color like yellow, numbers like "1" or "2".
- Diversity of answers was also handled while tackling open-ended questions. Since a single question can have multiple answers depending on the person's choice, **10 answers** from unique workers were considered for each question. Further, it was also ensured that the worker answering the question did not ask it.

4.1 Preprocessing steps

The major preprocessing steps followed for training the neural network architectures are listed as below:

- **Image feature extraction:** For MS COCO images we use two types of features: feature maps extracted from the last convolutional layer in **resnet152** and the bottom up image region features provided by [10]. For the bottom-up image region features, we use the precomputed train/val features provided in `https:`

//github.com/peteanderson80/bottom-up-attention. For each image, the bottom up region feature has dimensions of 36×2048. This indicates 36 salient image regions with each region having a corresponding feature vector of dimension **2048** (l2 normalized to 1). In case of resnet152, the pooled feature map of dimension $2048 \times 7 \times 7$was resized to 49×2048.

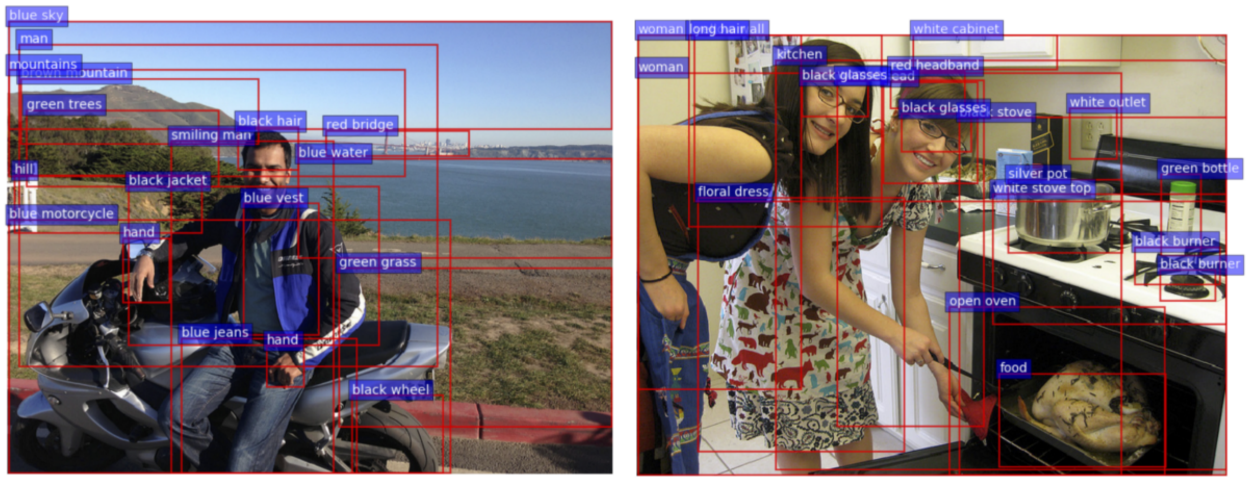


Figure 7: Examples from [10] showing salient image regions(with class and attribute labels) which were used to extract region-specific features.

- **Question token length selection:** For preprocessing of the questions, we first tokenize the questions by **splitting them into individual words and removing the punctuations, followed by conversion to lower case.** In order to pad the question tokens to ensure that the input sequences to **LSTM/GRU** are of the same length, we first plot the distribution of the question lengths. Since very few questions have lengths exceeding 14, the maximum question length was fixed at 14. Any question having length less than 14 was padded, before being passed as an input to the embedding layer.

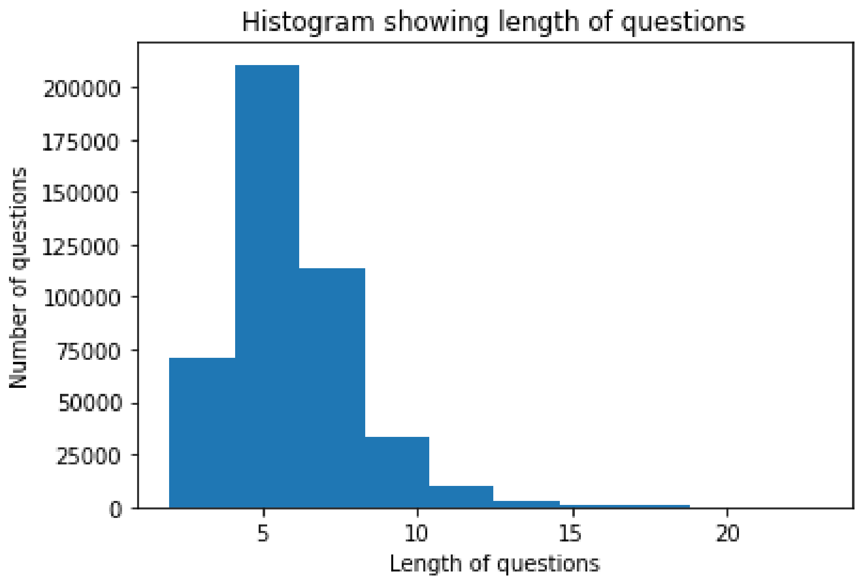


Figure 8: Distribution of the length of the questions from the training set

- **Text feature extraction:** We utilized both **Glove embeddings** (dim =300) for individual words and **document pool embeddings from BERT** as a part of the text processing pipeline. While utilizing BERT document embeddings, we extracted sentence level representation **(dim = 3072)** using the **flair framework** (https://github.com/zalandoresearch/flair).

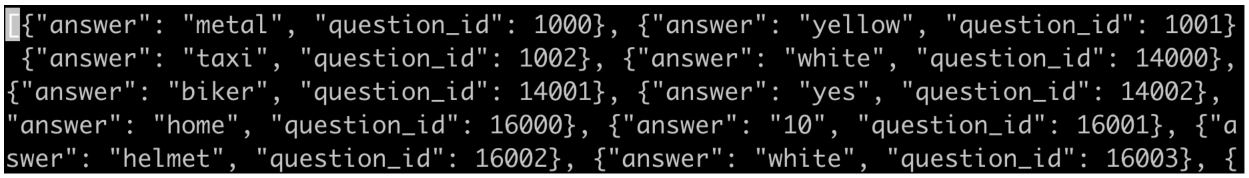
- **HDF5 storage:** In order that the data loading latency from the CPU is minimal, we store the precomputed image features (**resnet152/bottom up R-CNN**) and **BERT** features for questions in hdf5 files. The major advantage of using HDF5 storage is that the entire set of features can be stored as a dataset, which allows indexing like numpy arrays, without having the need to load it into memory.

5 Experiments:

The codebase developed for this project is hosted at https://github.com/nithinraok/VisualQuestion_VQA. Our experimental framework details are listed as follows:

- **Framework :** PyTorch 1.0.0
- **Python Version:** 3.6
- **Batch size:** 512
- **Optimizer:** Adam (lr =1e-3) for 2 class and 1000 class problem. Adamax(lr=2e-3) for 3129 classes.
- **GPU:** 2 NVIDIA GeForce GTX 1080 Ti, 1 NVIDIA K80
- **Activation:** Leaky ReLu and Tanh

The accuracy metric we use for comparing different models is listed at https://visualqa.org/evaluation.html. For computing accuracy, we first generated a JSON file for the validation questions where each entry has two fields: **question id** and the **single word answer.**



```
[{"answer": "metal", "question_id": 1000}, {"answer": "yellow", "question_id": 1001}
 {"answer": "taxi", "question_id": 1002}, {"answer": "white", "question_id": 14000},
{"answer": "biker", "question_id": 14001}, {"answer": "yes", "question_id": 14002},
"answer": "home", "question_id": 16000}, {"answer": "10", "question_id": 16001}, {"a
swer": "helmet", "question_id": 16002}, {"answer": "white", "question_id": 16003}, {
```

Figure 9: Sample entries from the JSON file used for validation.

For each question id, the single answer provided in the JSON file was matched with the given answer annotations. The number of matches was then used for computing the accuracy as follows:

$$\text{accuracy} = \min\left(\frac{\#\text{ humans that provided that answer}}{3}, 1\right) \tag{8}$$

For our experimental results we created 3 subsets of our dataset:

- **Yes/No Subset:** Yes/no subset consists of only questions with yes/no as answers. This subset had 81293 validation and 168562 training questions.
- **Top 1000 classes subset:** The top 1000 classes in terms of answer occurences (labels) were considered while subsampling the entire dataset. This subset had 402651 training and 193597 validation questions.

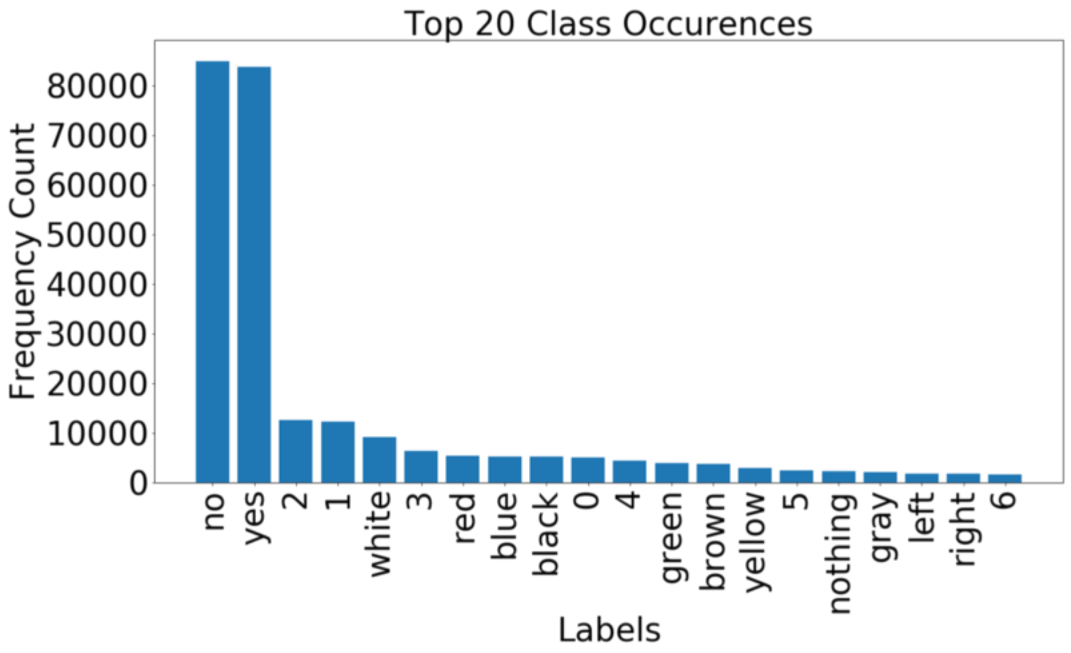


Figure 10: Histogram of answer occurrences for the top 20 classes

- **Complete dataset:** The complete dataset has 3129 classes (answer labels). The entire dataset has 443757 training questions and 214354 validation questions.

For each question, we had a **question id** and an associated **image id**. The image id was used to retrieve the exact image from the **MS COCO dataset**.

We first performed a feature analysis on the Yes/No subset for the question guided attention model. **Bottom-up RCNN** image features showed superior performance when compared with **resnet152**. For question representation, Glove word embeddings showed better performance than BERT document embeddings. The reason can be attributed to the mean pooling of the BERT word embeddings, which may not be the optimal way of representing questions. Using the findings on Yes/No subset, we fix the image feature set to be **bottom-up RCNN** and **word embeddings** are extracted from Glove for our subsequent experiments. The results are listed in Table 2.

For the 1000 classes subset, we implemented the baseline method shown in Fig 3 (implementation based on `https://github.com/anantzoid/VQA-Keras-Visual-Question-Answering`). But the baseline method based on the multiplicative fusion of question and image feature vectors relied on a condensed feature vector representation of the image, which was not influenced by the question representation. The next step that we decided was to use the question guided attention-based model where importance was assigned to different image regions based on the question asked. As seen in Table 3, the attention model (row 2) performed much better in comparison to the LSTM+VGG baseline for the top 1000 classes.

In order to improve the performance and remove the restriction of the same dimensions for fusing question and image feature vectors, we implemented the MFB module and used that in place of multiplicative fusion. We can see the improvement after including MFB in case of 1000 classes from Table 3. The best performing model in Table 3 has GRU hidden state representation of dimension 1280 and MFB based fused feature vector of size 1000. Since we use two MFB blocks as shown in Fig 4, the final fused feature vector is obtained by concatenating two 1000 dimension feature vectors.

For comparison with the validation results presented in [1] and [10], we trained the question guided attention + MFB pooling model on the entire dataset of 3129 classes. While training, we considered the problem of answer classification as a multi-label problem. The model variant with RCNN image features, Glove word vector embeddings, hidden state of 1280 (unidirectional GRU) and 1000 dim fused feature vector performed better than other models as shown in Table 3. In terms of activation functions, we experimented with both Leaky RelU and Tanh but Leaky ReLU was superior in terms of performance.

The models are coded using the following convention: **Att**: Attention baseline, **MFB**: Multi-Modal Factorized Bilinear Pooling, **Resnet152**: Image features ($2048 \times 7 \times 7$) **R-CNN**: Bottom up Image features (36×2048), **mult** :

Multiplicative fusion of image and question embeddings , **2** :2 classes , **1000** : 1000 classes, **Uni**: Unidirectional , **Bi**: Bidirectional, **GloVe**: GloVe embeddings, **GRU**: Gated Recurrent Unit, **BERT**: Bidirectional Encoder Representations from Transformers, **hid**: hidden state size of GRU

Models	Validation Accuracy
Attn_hid_512_resnet152_Glove_uni_GRU	72.25
Attn_hid_512_resnet152_Glove_bi_GRU	73.05
Attn_hid_512_RCNN_Glove_bi_GRU	74.70
Attn_hid_1280_RCNN_Glove_bi_GRU	**74.93**
Attn_hid_1024_RCNN_BERT	74.70

Table 2: Results on the VQA v2 validation dataset(Open -Ended questions) for Yes/No classes. The 4 rows indicate results from our model variants (Here the models are trained with single class softmax loss)

Models	Yes/No	Number	Other
Att_hid_1280_R-CNN_Glove_Uni_GRU_mfb_fusion_Leaky_Relu	**80.11**	**45.23**	**60.95**
Att_hid_1024_resnet_GloVe_Uni_GRU_multiplication_Leaky_Relu	76.29	39.55	54.11
LSTM_hid_1024_vgg_1024_Tanh[2]	75.36	35.56	57.45

Table 3: Results on the VQA v2 validation dataset(Open -Ended questions) for topmost 1000 classes. The first 2 rows indicate results from our model variants and the last row results are taken from [2]

Models	Overall	Yes/No	Number	Other
Attn_hid_1280_RCNN_GloVe_UNI_GRU_MFB_(1000 dim proj)_Leaky_Relu	64.44	80.67	45.54	57.25
Attn_hid_1280_RCNN_GloVe_UNI_GRU_MFB_(1000 dim proj)_Tanh	64.03	80.58	42.68	56.7
Attn_hid_1280_RCNN_GloVe_UNI_GRU_MFB_(500 dim proj)_Leaky_Relu	**64.73**	**81.08**	**44.24**	**57.33**
Attn_hid_1280_RCNN_GloVe_UNI_LSTM_MFB_(500 dim proj)_Leaky_Relu	64.53	80.81	44.01	57.19
Tips and Tricks paper reference model [10]	63.15	80.07	42.87	55.81
Up Down model (with bottom up RCNN features) [1]	63.2	80.3	42.80	55.8

Table 4: Results on the VQA v2 validation dataset(Open -Ended questions) for 3129 classes. The first 4 rows indicate results from our model variants. The last two results are taken from [3] and [2]. Here the models are trained using multi-label loss.

6 Visualizing Examples

We used Grad-CAM [9] to find out success and failure cases of our models. Below we show examples highlighting misclassified and correctly classified example from the validation dataset in Fig 11. In the left image, our model failed to predict the color of the red sandal as it focused on black sandal and on the right image, our model predicted that there is a TV and a remote in the image and correctly predicted No for the question, **"Are these women sitting on train?".**

7 Challenges faced

- **Class imbalance:** Yes/No class labels comprise the majority of the dataset. Need batch balancing of the minority classes to improve performance
- **Gradient explosion/saturation:** Experienced gradient explosion which was tackled through gradient clipping in PyTorch
- **Computational Bottleneck:** Due to lack of sufficient storage/computing resources and time constraints, extra data from Visual Genome was not used for training
- **Huge training time for current models:** Current models take enormous time to train on a single GPU. For Murel [5] network, we initially tried to train the whole network for 1000 classes. Even with multi-GPU setup, each epoch was taking 3 hours to complete the training. Therefore, due to time constraints, we decided to focus more on the attention models (+ fusion with MFB) instead of Murel.

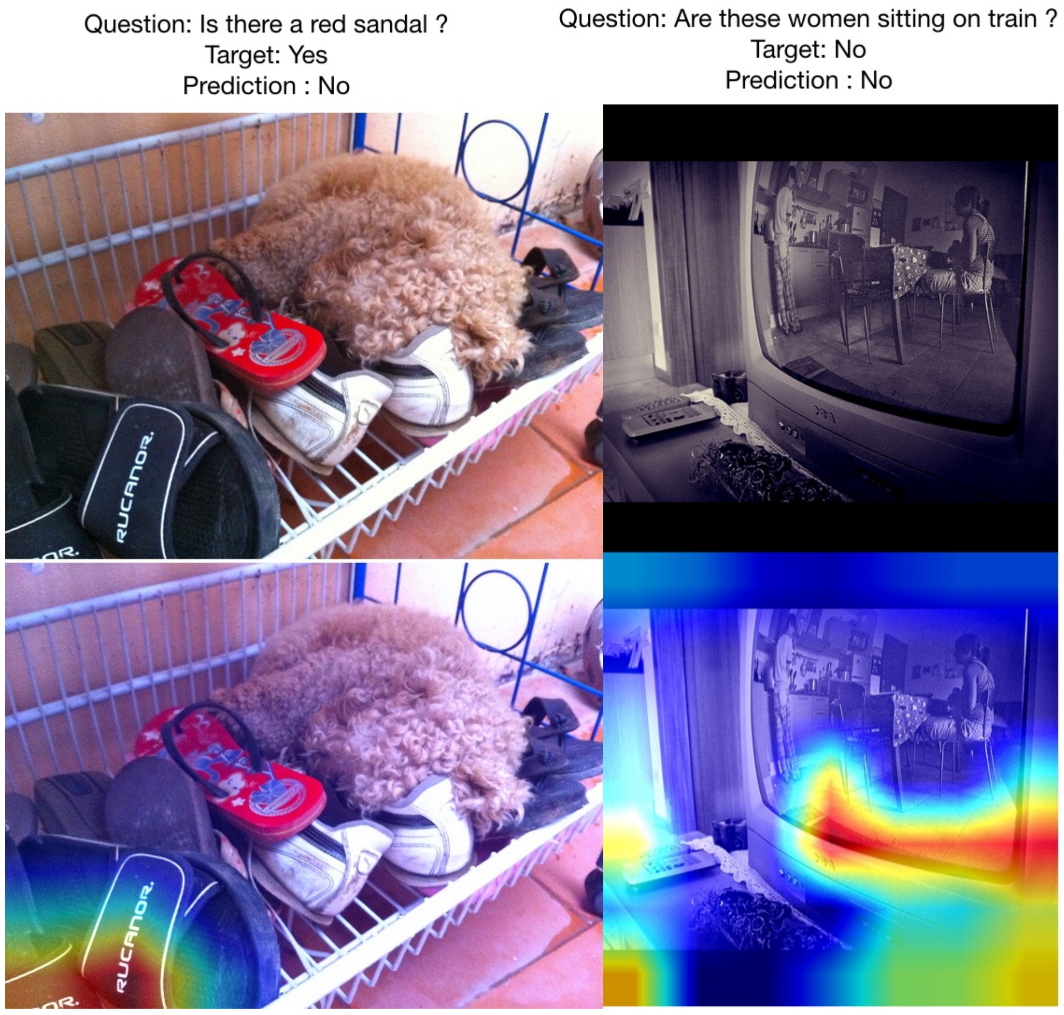

Question: Is there a red sandal ?
Target: Yes
Prediction : No
Question: Are these women sitting on train ?
Target: No
Prediction : No
RUCANOR.

Figure 11: GRAD-CAM examples

8 Future Work

Future work involves the usage of Visual Genome dataset (https://visualgenome.org) to improve the training set. Further, during the batch generation, we did not resort to class balancing. Since there is a huge class imbalance (most of the questions being yes/no type), batch balancing will help in tackling those questions that have answers having very low occurrences. Since the performance of the VQA system relies on multiple components, improved features for both image and text modalities can boost the performance. For image, instead of Faster-RCNN based features, RetinaNet[6] and YOLOv3[8] can be considered as alternatives. In case of textual representation, instead of simple mean pooling of the BERT word embeddings, RNN based document embeddings and skip-thought vectors for representing entire question can be used. Further other contextualized word representations like ELMO[7] can be potentially used for extracting question representations. In the case of fusing text and image embeddings, techniques based on tensor decomposition like BLOCK[4], MUTAN[3] can be explored for improved results.

9 Acknowledgement

We would like to thank Prof. Joseph Lim for his guidance during the duration of course project along with the lectures which helped us in understanding the state of the art deep neural network architectures. We would also like to thank TAs Hanpeng and Shao-Hua for their technical help and discussions during the duration of course project.

10 Individual Contribution

- **Digbalay**: Attention Baseline architecture and hyperparameter tuning, MFB implementation and tuning, Grad-CAM
- **Nithin**: VGG based baseline network implementation, Attention Baseline architecture and hyperparameter tuning, BERT implementation for attention baseline
- **Namrata & Aditya**: Data Preprocessing and Hyperparameter tuning of networks for Yes/No data subset

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and VQA. *CoRR*, abs/1707.07998, 2017.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[3] Hedi Ben-younes, Rémi Cadène, Matthieu Cord, and Nicolas Thome. MUTAN: multimodal tucker fusion for visual question answering. *CoRR*, abs/1705.06676, 2017.

[4] Hedi Ben-younes, Rémi Cadène, Nicolas Thome, and Matthieu Cord. BLOCK: bilinear superdiagonal fusion for visual question answering and visual relationship detection. *CoRR*, abs/1902.00038, 2019.

[5] Remi Cadene, Hedi Ben-Younes, Nicolas Thome, and Matthieu Cord. Murel: Multimodal Relational Reasoning for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2019.

[6] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[7] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[8] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[9] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[10] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *CoRR*, abs/1708.02711, 2017.

[11] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. *CoRR*, abs/1708.01471, 2017.