Visual Question Answering (VQA)

Apoorv Khattar, 2016016 — Madhur Tandon, 2016053 — Madhav Thakker, 2016159
CSE641: Deep Learning 2020 Final Project Report Group 5

1 Motivation

Visual question answering is a task proposed to connect Computer Vision and Natural Language Processing (NLP). Computer vision is concerned with methods that teach computers to perceive images same as how humans do. On the other hand, NLP deals with interactions of computers with natural languages same as how humans do. Many significant developments in both these fields have happened separately but VQA requires simultaneous understanding of the visual content of images and the textual content of questions.

2 Problem Statement

VQA is a multimodal problem where we answer a question about an image, the answer could be a multiple choice question or a short phrase. There has been a huge interest to explore techniques that can fuse multimodal information, in this case the features of an image with the textual features of a question. In our project, we focus on some of the recent fusion techniques that have been proposed for bilinear models. Bilinear models consist of two networks: one for extracting visual features from an image and the other for extracting textual features from a question, which are then fused to generate an answer for a question.

3 Related Work

There have been many new methods for VQA that have been summarized by Srivastava et al. [7]. In this section, we will focus specifically on fusion techniques that have been proposed for VQA. Kim et al. [5] provide a simple fusion technique where given the image and the question feature vector of the same size, the combined feature vector can be determined by calculating the hadamard product. Yu et al. [8] state that for a given question not all regions in the image may be relevant and thus, instead of using the global features of an image, they propose a co-attention layer to determine the relevance of different regions in an image for a particular question. It is possible be that the i^{th} neuron of the image feature vector is related to the j^{th} neuron of the question feature vector. However, calclating the relation between every pair of neurons is not memory efficient so Younes et al. 2017 [3] use Tucker decomposition to find a compact representation for pair-wise relations between the neurons of the two feature vectors. Building on top of this approach, Younes et al. 2019 [2] perform Tucker decomposition on the two feature vectors in chunks.

4 Methods

Given an image and a question, we use two models to extract the image and question feature vector, v and q respectively. Now we need to fuse q and v to predict the final answer. In our project we focus on the following fusion techniques:

4.1 MLB

Kim et al. [5] explore simple tehniques for fusion such as element-wise sum, concatenation, hadamard product etc. They show empirically that hadamard product between q and v produces the best results. For the image vector v and question vector q, the fused feature vector, f is given by:

$$y_i = (q^T W_q)_i * (v^T W_v)_i,$$

where W_q and W_v are weight matrices that transform q and v to have the same dimensions.

4.2 MUTAN

The fused feature vector in MLB captures the relation between the neurons at the same index for the two feature vectors. However, the fused vector must capture relation between every pair of neurons. This fused feature vector is given by,

$$y_k = \sum_{i=1}^{d_q} \sum_{j=1}^{d_v} T_{ijk} q_i v_j,$$

$$y = T \times_1 q \times_2 v,$$

where T is a learnable 3D weight matrix and d_v, d_q are the dimensions of v and q. Since the dimensions of feature vectors are large, thus there are a lot of parameters in T. Younes et al. 2017 [3] use tucker decomposition to reduce the number of parameters in T. The feature vector y can be calculated as,

$$y = T \times_1 (q^T W_q) \times_2 (v^T W_v) \times_3 W_o,$$

where W_q and W_v are weight matrices to reduce the dimensions of q and v and W_y increases the dimensions of final fused vector. Figure 1 shows how y is determined.

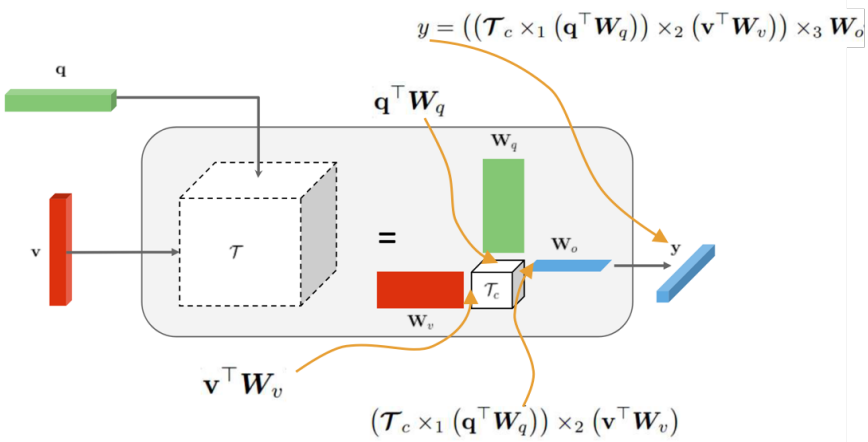

Figure 1: Tucker Decomposition in MUTAN

4.3 BLOCK

In MUTAN, we compress the feature vector q and v to reduce the number of parameters in T but this can lead to loss of information if the reduced sized feature vectors do not efficiently capture all the information in q and v. Younes et al. 2019 [2] propose to determine the fused vector y in chunks i.e. the feature vector q and v are divided into blocks and Tucker decomposition is done on each block. Finally all these blocks are concatenated to obtain the fused feature vector y. The BLOCK method can be seen in figure 2.

5 Data Acquisition

We use the publicly available VQA Dataset: VQA-v2 The dataset consists of around 200,000 images from the MS-COCO dataset with at least 3 questions per image. Each one of these questions is then answered by 10 annotators, yielding a list of 10 ground truth answers. Due to the large size of the dataset, we take 5000 samples from the training set containing questions with **yes/no** answers, 800 samples as validation set and 200 samples as test samples. We chose only **yes/no** questions since they had the most number of samples and all the papers report their accuracy, allowing us to compare the results of our model with those mentioned in the paper.

5.1 Preprocessing Techniques

During the training phase, each image is resized to (256,256) and a random patch of size (224,224) is passed through the network. The training set and validation set are used to determine the vocabulary and tokenize each question.

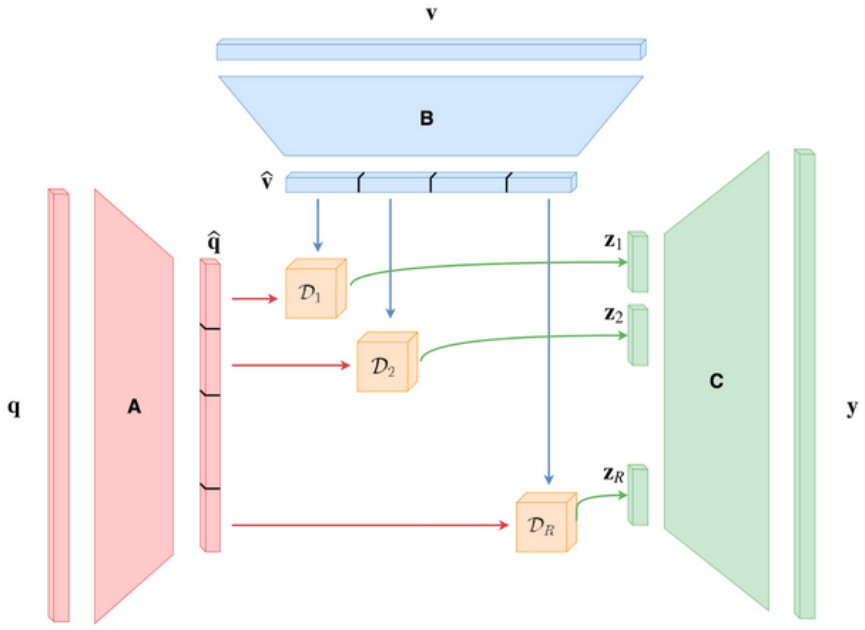

Figure 2: BLOCK overview (image taken from paper)

6 Experiments and Results

For our experiments, we use a pretrained VGG16 [6] as an image extractor and a bidreictional LSTM [4] as an question feature extractor. We also use attention, as explained by Bahdanau et al. [1] to compare the performance with vanilla LSTM.

6.1 Evaluation Metric

When the model predicts an answer for a given question, the VQA Accuracy is given by:

$$min(1, \frac{humans\ that\ provided\ that\ answer}{3}),$$

that is, if the predicted answer appears at least 3 times in the ground truth answers, the accuracy for this example is considered to be 1. Intuitively, this metrics takes into account the consensus between annotators. Since we reduce it to a **Yes / No** problem, accuracy is considered as the metric.

6.2 Results

Table 1 summarizes the accuracy of the baselines on the training and validation sets:

Model	**Train Acc.**	**Val Acc.**
MLB	0.83647	0.63399
MUTAN	0.89959	0.67369
BLOCK	0.91224	**0.68733**

Table 1: Accuracy on training and validation set for baseline

Table 2 summarizes the accuracy of the VQA model with attention on the training and validation sets:

Model	**Train Acc.**	**Val Acc.**
MLB	0.83779	0.62735
MUTAN	0.85926	0.68323
BLOCK	0.86139	**0.70103**

Table 2: Accuracy on training and validation set for all models with attention

The training and validation plots for all models have been provided in the supplementary material 8.

6.3 Inference

6.3.1 Comparison between our results and results of the paper

Using our code, we obtained the best performance with the BLOCK fusion technique. This is in accordance with the paper, which achieved 82.86 percent accuracy for **Yes/No** questions. The training set we used is around 100 times smaller. Thus, it is possible that some questions available in the validation set can contain tokens not present in the training set at all. This can lead to inadequate feature representations, and thus, our results fall short from the paper with an accuracy of around 70.1 percent.

6.3.2 Convergence of our models

From the training and validation loss plots in section 8, it can be seen that all the different techniques we used, have converged successfully.

7 Individual Contribution

Each member worked on the following approaches:

- Apoorv Khattar: MUTAN
- Madhav Thakker: MLB and Attention
- Madhur Tandon: BLOCK

8 Supplementary Material

The following are the training and validation plots for all the three models.

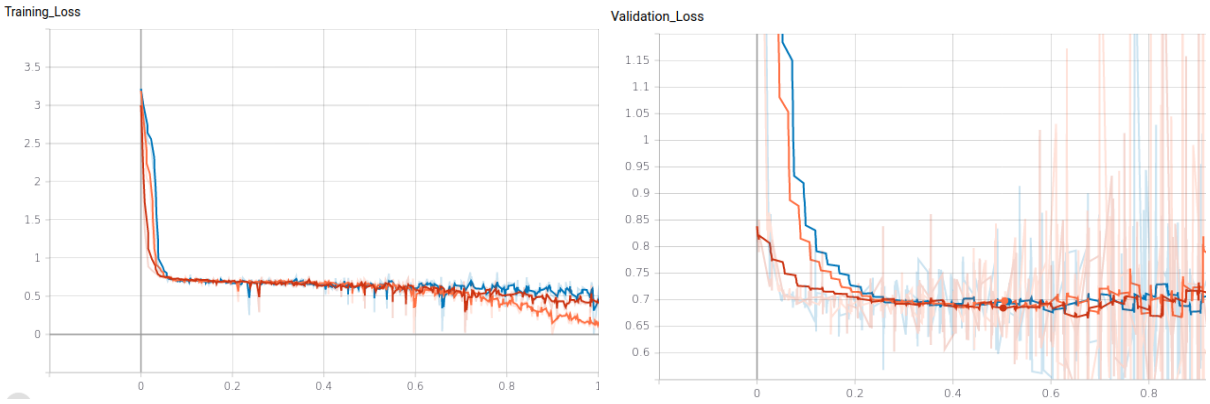

Figure 3: Training and validation loss plots for MLB (orange), MUTAN (blue), BLOCK(red)

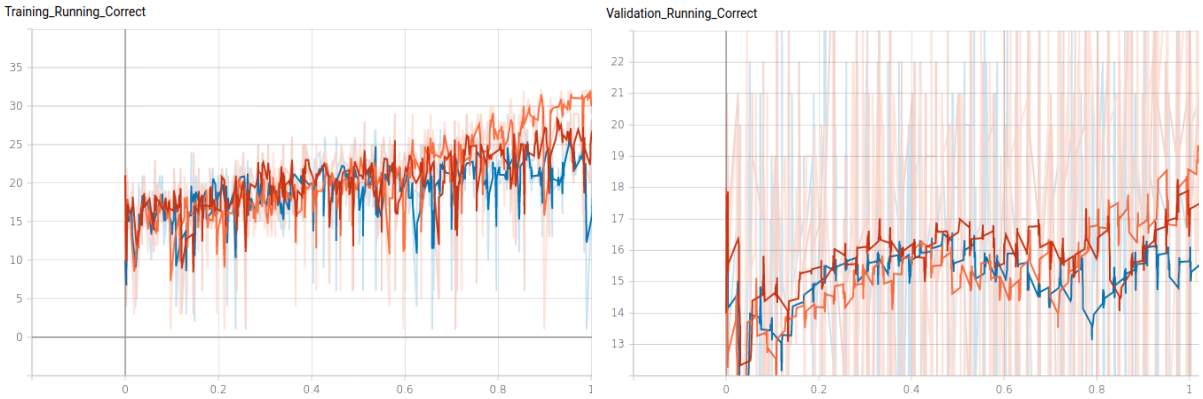

Figure 4: Training and validation batch-wise correct answers for MLB (orange), MUTAN (blue), BLOCK(red)

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Hedi Ben-Younes et al. "Block: Bilinear superdiagonal fusion for visual question answering and visual relationship detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 8102–8109.

[3] Hedi Ben-Younes et al. "Mutan: Multimodal tucker fusion for visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2612–2620.

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

[5] Jin-Hwa Kim et al. "Hadamard product for low-rank bilinear pooling". In: *arXiv preprint arXiv:1610.04325* (2016).

[6] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[7] Yash Srivastava et al. "Visual Question Answering using Deep Learning: A Survey and Performance Analysis". In: *arXiv preprint arXiv:1909.01860* (2019).

[8] Zhou Yu et al. "Multi-modal factorized bilinear pooling with co-attention learning for visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1821–1830.