50.021 Artificial Intelligence Project

Visual Question Answering

Single-Word-Answer Questions

Koh Hui Wen (1003593)
He Yuhang (1003775)
Chen Hongfang (1003759)
Zhang Shaozuo (1003756)

Introduction

Visual Question Answering (VQA) is a computer vision and natural language processing research problem which has been gaining popularity in recent years [1]. The goal of VQA is to understand the semantics of the image and to be able to answer open-ended, free-formed natural language questions. Such a system would greatly help the visually impaired, and may even help us to get a better understanding of the image [2].

We aim to create a neural network that takes in an image and a question, and outputs the answer to the question based on the image. Based on different categories, the answer could be binary(yes/no), single-word sequence and multiple-word sequence. This project targets a subset of the questions space with only single-word answers.

Related Works

VQA Efforts

Various papers have begun to study visual question answering. [2] explores “late fusion” - i.e, the Long Short Term Memory (LSTM) question representation and the Convolutional Neural Network (CNN) image features are computed independently, fused via an element-wise multiplication, and then passed through fully-connected layers to generate a softmax distribution over output answer classes. The VQA model consists of a 2-channel vision model using VQA that provides an embedding for the image, a question model using Bag-of-Words/LSTM/Deeper LSTM Question that provides an embedding for the question.

Question-guided Attention

[3] further explores the effect of the attention module. The question inputs are tokenized and represented using Global Vectors for Word Representation (GloVe) word embeddings and then passed through a Gated Recurrent Unit (GRU) to create the final question embedding. The image feature vectors along with the final question embeddings are processed by a one-way top-down attention module that computes the relevance of each object/vector to the current question embeddings. This paper treats the attention mechanism as the hyperparameters of the models and experiments with different types of attention modules (refer to Figure 1).

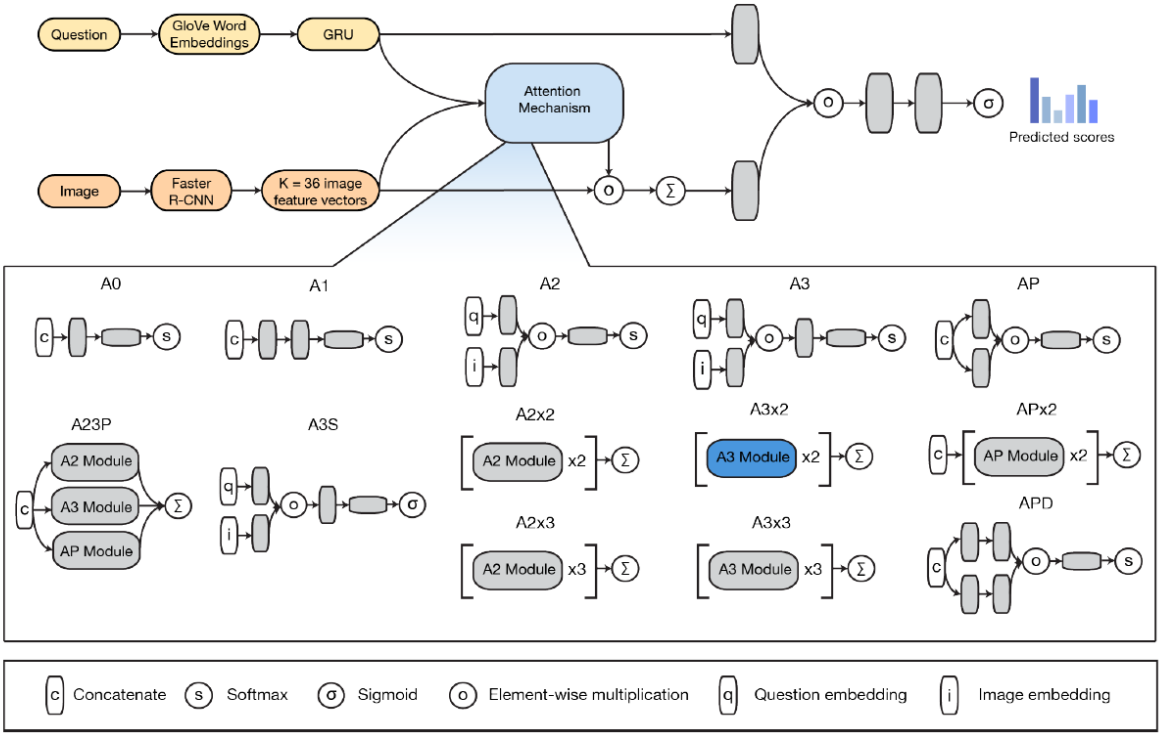

Figure 1. Visual representation of VQA model architecture and attention modules [3]

Methodology

Dataset

General Information of Data

We are using the VQA v2.0 Balanced Real Images Dataset, which contains open-ended questions about images and their respective answers [1]. However, we only used the data points that had single-word answers. The total number of single-word answer samples is 609,493, with 411,198 training samples and 198,295 validation samples. The VQA dataset does not provide test samples with labels, thus we split the validation samples into two sections: 30% as test set, 70% as the final validation set. In conclusion, the ratio train: validation: test is 67%:23%:10%(refer to Figure 2).

The raw dataset consists of 6 files:

1. Training and validation annotation files in JSON format that store the question-answer pairs.
2. Training and validation question files in JSON format that store the question-image pairs.
3. Training and validation image sources.

Class Distribution

Each sample is a tuple of one image, question and single-word answer. There are three types of answers: number, yes/no and other. Figure 2 shows the distribution of class.

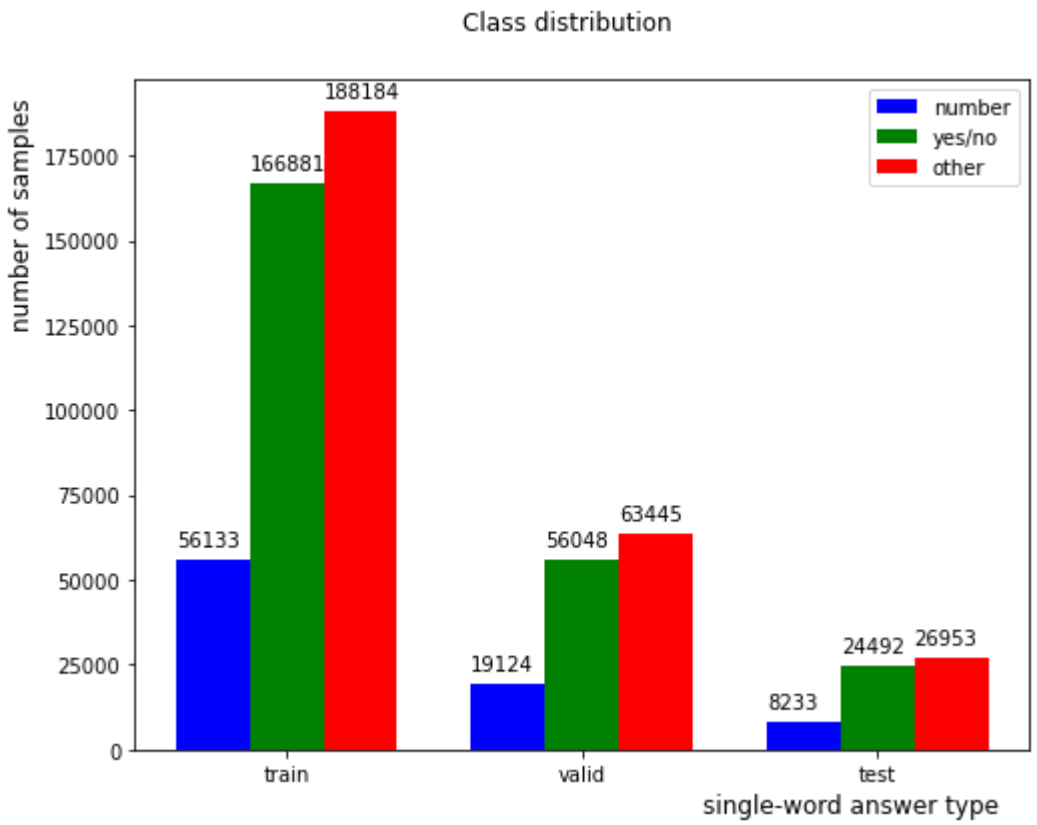

Figure 2. Class distribution of the dataset

We observed that in both training, validation and test dataset, the three classes are almost balanced.

Distribution of Top 50 Frequent Words in Questions space

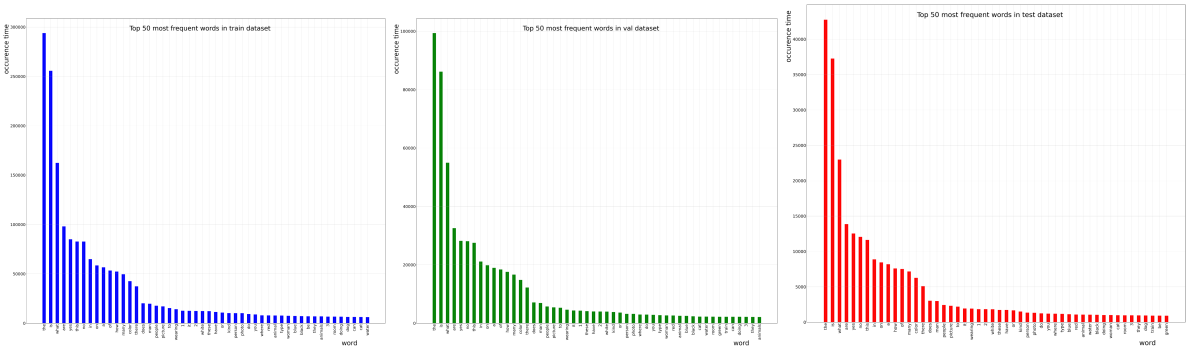

Figure 3. Distribution of top 50 frequent words for train (left), val (middle), test (right) dataset

It is observed that “this”,”is”,”what”,”are”,”yes”,”no”,”a”,”of”,”how”,”many” are the most frequently appeared words in training, validation and test dataset. In addition, the frequency distributions are almost the same.

Preprocessing of Data

Firstly, we preprocessed the images by resizing them into a 256x256px image, and then doing a center crop to get a 224x224px image. The reason why resizing is applied is that the

input images can be in different sizes, and the central crop ensures that the image is focused on the middle, where the important details of the image would be. The images are then normalized to allow faster convergence during the training of our model since large values can make the learning process more computationally expensive.

Next, to preprocess the question text data for our approach using a Recurrent Neural Network (RNN), we created a tokenizer that would clean a string of words by converting it into lower case and removing punctuations, and then splitting the sentence into a list of words. The vocabulary for the question text data can be built using the lists of words, and each word in the questions can be converted to numbers based on its index in the question vocabulary. The question can thus be represented by a tensor. The size of the question vocabulary is 13457.

Similarly, we built a vocabulary for the answer text data and converted the answers into tensors. The size of the answer vocabulary, i.e. the number of classes, is 1823.

On the other hand, for our approach using Bidirectional Encoder Representations from Transformers (BERT), a pre-trained BERT tokenizer provided by the transformers library is used. After tokenization, questions are padded to the longest sequence length in a batch and converted to tensor.

Evaluation Metrics and Loss

We approached the VQA task as a multi-class classification problem. Since for single-word answers, the model output is either correct or wrong. Thus, one evaluation metric we will be using is **accuracy**, which is calculated as in Equation (1).

$$Accuracy = \frac{No.\ of\ correct\ predictions}{Total\ no.\ of\ samples} x100\% \text{ ---- (1)}$$

Due to a large number of possible classes, accuracy alone may be too strict to evaluate the model, since any random guess only has 0.05% probability of getting the right answer out of the 1823 classes. Hence, we decided to include the **top-5 accuracy** in our evaluation metrics as well. The top-5 accuracy is the accuracy where the true class is one of the top 5 most probable classes predicted by the model.

Since our task is a classification problem, we used the **cross-entropy loss** as our loss function, allowing us to get the probability for each class for each prediction.

Experiments

Architectures

Our neural network architecture will be multimodal due to the different input formats of image and text. It consists of an Image Model, a Question Model, and a Fusion scheme that combines the outputs of the two models to predict the final answer (refer to Figure 4).

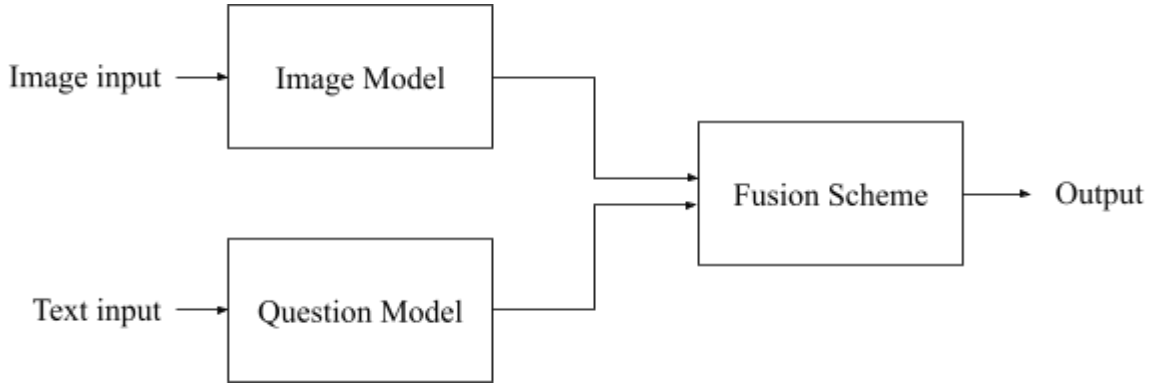

Figure 4. Multimodal structure

The **Image Model** will be a CNN that performs feature extraction on the image, detecting colours or objects that could potentially answer the given question. Due to a wide range of pre-trained CNN models available, we used transfer learning for the Image Model and experimented with 4 different CNN architectures: VGG16, ResNet-152, GoogLeNet, and AlexNet.

On the other hand, we experimented with 2 different approaches for the **Question Model**. Firstly, we used an RNN consisting of GRUs to process the text sequence data and extract the underlying context of the question. Next, we also attempted to use transformers for the Question Model. In particular, we used a pre-trained BERT for its natural language processing capabilities.

The **Fusion scheme** takes in the output of the Image Model and the Question Model, and combines them to produce an output that answers the input question according to the input image. There may be many ways to combine these outputs, and we experimented with element-wise multiplication and concatenation.

We also experimented with the addition of attention layers before the fusion, similar to the structure in Figure 1. The **Attention** mechanism enables deep neural networks to extract localized features from input data. It is expected that question-guided visual attention would highlight regions in the image which are highly relevant to the answer. We first tile the embeddings of the question overall spatial positions, then obtain the attention layer by concatenating the image features and new question features. After that, we apply the attention layer on the image features and perform softmax. The attention softmax layer indicates the weights of each pixel in the image. Larger attention weights mean higher relevance between the pixels and the answer.

The results of our experiments are recorded in the table below.

Table 1. Architecture experimental results

Question Model	**Image Model**	**Fusion scheme**	**Attention layer**	**Validation Accuracy**	**Validation Top-5 Accuracy**
1 GRU layer	VGG16	Element-wise multiplication	No	0.391	0.807
1 GRU layer	ResNet-152	Element-wise multiplication	No	0.418	0.818
1 GRU layer	GoogleNet	Element-wise multiplication	No	0.391	0.805
1 GRU layer	AlexNet	Element-wise multiplication	No	0.441	0.814
2 GRU layers	AlexNet	Element-wise multiplication	No	0.409	0.810
1 GRU layer	AlexNet	Concatenation	No	0.386	0.798
1 GRU layer	Alexnet	Element-wise multiplication	Yes	0.438	0.812
BERT	ResNet-152	Element-wise multiplication	No	**0.498**	**0.856**
BERT	AlexNet	Element-wise multiplication	No	0.479	0.836

As expected, models with pre-trained models for both the Question Model and the Image Model performed significantly better than the ones without. Element-wise multiplication also appeared to work better than concatenation, which could possibly be because the multiplication matches the features from the image and the question while concatenation simply puts the two sets of features together. This finding is consistent with the result in [2], where element-wise fusion performs better for both open-ended tasks and multiple-choice tasks. The experiment also proved that the addition of attention layers did not improve the validation performance of the model, and thus is not necessary for the final architecture since it only consumes more computational resources.

From the results, the architecture using BERT, ResNet-152, element-wise multiplication fusion scheme, and no attention layer has the best validation accuracy of 49.8% and top-5 accuracy of 85.6%. Hence, we will use this for our final architecture.

Hyperparameter Tuning

We conducted another experiment to find the best hyperparameters for the final architecture by tuning the learning rate, batch size, and optimizer. Each experiment was run for 4 epochs, and the results are recorded in the table below.

Table 2. Hyperparameter tuning results

Learning rate	**Batch size**	**Optimizer**	**Validation Accuracy**	**Validation Top-5 Accuracy**
1e-5	32	AdamW	0.498	0.856
1e-5	16	AdamW	0.497	0.855
1e-5	8	AdamW	0.510	0.852
1e-4	16	AdamW	0.223	0.539
1e-6	16	AdamW	0.426	0.797
1e-5	16	Adam	**0.547**	**0.875**

We found that learning rate of 1e-5, batch size of 16 and Adam optimizer achieves the highest validation accuracy of 54.7% and top-5 accuracy of 87.5%. Therefore, we will use these hyperparameters in training our final model.

Results and Analysis

Evaluation on Test Set

Overall, our best model consists of ResNet-152 as the Image Model, BERT as the Question Model, element-wise multiplication as the Fusion scheme. Using a learning rate of 1e-5, batch size of 16 and Adam optimizer with cross-entropy loss, we were able to achieve a test accuracy of **49.7%** and top-5 accuracy of **86.0%**. This means the model was able to correctly answer the question based on the image almost half of the time, and has quite a high probability of correctly answering the question if it outputs the top 5 answers.

We trained the model for 4 epochs and validated on a subset of the validation set every 200 batches. Figure 5 below shows the training and validation curves, which include loss, accuracy, and top-5 accuracy against training epochs.

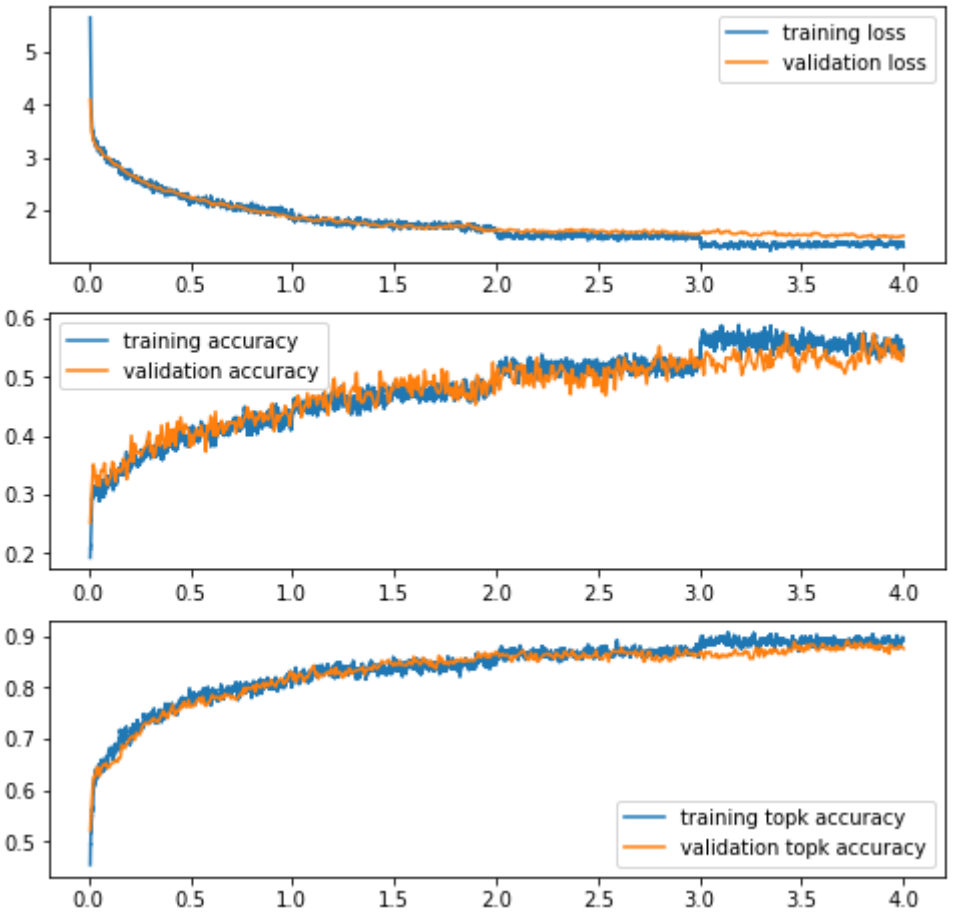

Figure 5. Training and validation curves

From Figure 5, we can see that the learning curves converge at around the 4th epoch, implying that the model has been sufficiently trained. The training and validation curves are also very close to each other, thus the model is not overfitting. Regularizations such as Dropout layers are therefore not necessary in our architecture.

Figures 6, 7, and 8 below showcase our model's outputs to some samples of the test set. The question focuses on different areas of the images to show that our model is generalized such that it is able to extract features from various parts of the image and not just the center. For example, Figure 7 shows an image of a baby girl with an ear-piercing at the top right of the image, and Figure 8 shows a baby with a tie around their neck at the bottom right of the image. For these images and questions, the model is able to give the correct answer.

Q: [CLS] what does the train say? [SEP] [PAD] [PAD] [PAD] [PAD] [PAD]
A: <unk>
Model: <unk>

Figure 6. Correct prediction to “what does the train say?”

Q: [CLS] are the girls ears pierced? [SEP] [PAD] [PAD] [PAD] [PAD] [PAD]
A: yes
Model: yes

Figure 7. Correct prediction to “are the girls' ears pierced?”

Q: [CLS] what is around the baby's neck? [SEP] [PAD] [PAD]
A: tie
Model: tie

Figure 8. Correct prediction to “what is around the baby’s neck?”

However, the model also outputs wrong answers. For example, from Figure 9, the image of a pasta salad is given, and the question is “Is this served hot or cold?”. The model answers “hot” when the answer should be “cold”. While the answer is wrong, the model was able to pick the other possible logical answer out of the 1823 possible classes, as opposed to answering the question with “yes” or “tie” which would not make any sense. A possible reason for the model malfunction may be that the model has seen more hot foods during training than cold foods, thus it became biased towards answering “hot” when asked if the food is hot or cold, since there are no visual cues in the image about the temperature of the object. This behaviour would be similar to humans, since someone who is used to hot foods and has never seen pasta salad would also think that the dish is served hot.

Q: [CLS] is this served hot or cold? [SEP] [PAD] [PAD] [PAD] [PAD]
A: cold
Model: hot

Figure 9. Incorrect prediction to “is this served hot or cold?”

Comparison with State-of-the-Art

Table 3. Comparison with the original VQA paper [2]

Models	**Test Accuracy**
Deeper LSTM Q + norm I	58.16%
Our model	49.7%

As mentioned in the Related Works section, the “late fusion” is explored by [2] and the paper proposed their best model with l_2 normalized activations from the last hidden layer of VGGNet for their image channel, and a deeper LSTM with 2 hidden layers for the question channel.

Comparing this with our approach, our model’s performance is worse by about 10% accuracy despite having similar architecture structures and using pre-trained models for the Question Model. This could be due to our model lacking the normalization of the activations from the Image Model, making the feature extraction of the images weaker and perhaps resulting in the model giving less weight to the Image Model. It may be that our model is more dependent on the Question Model, detecting keywords such as “colour” to decide

possible answers. This could be a reason to explain why the top-5 accuracy is significantly higher than the top-1 accuracy, as the question may ask “what colour is this?” and the top 5 answers produced by the model could just be 5 different colours, regardless of what colour the object in the image is.

That said, we are unable to definitively say that our approach is much worse than the model proposed by [2] due to the differences in the dataset used. Firstly, we are using the VQA v2.0 dataset, an upgrade of the original VQA dataset used in [2]. The new dataset is more “balanced” such that there are similar images to the same question which results in different answers [1], thus our model would have to be able to interpret the small differences between the images. Otherwise, the test performance of the model would be penalized more heavily, as compared to evaluating the model on the dataset used in [2].

Table 4. Comparison with state-of-the-art models on VQA v2 test-std

Model	**Test Accuracy**
MSR+MS Cog.Svcs.,X10 models	77.45%
Baidu NLP DU-SG Team	76.9%
MSR+MS Cog. Svcs	76.63%
InterBERT Ensemble	76.1%
StructVBERT-base Ensemble	76.01%
Our best model	49.7%

The state-of-the-art (SOTA) models in Table 4 [6] are trained on VQA 2.0 dataset as well. Instead of focusing on single-word-answer samples only, these models cover all the categories: number, yes/no and other. The top 5 models show very promising performance in all three categories. By 2021, the highest performance reaches 77.45% test accuracy on test-std which greatly outperforms our best model.

The SOTA models used the training dataset that covers all three categories, thus the model learns more information and potentially has a more coherent understanding of the general answer pattern. These models also used more advanced algorithms and are not as limited by the capacity of pre-trained models and GPU memory compared to our development. Last but not least, our model is not tested on VQA v2 test-std but the test data we split from the original validation set, which may be slightly different than if we test on VQA v2 test-std.

User Interface

A web page is built to demonstrate the model. The model is hosted as a backend service using Flask, and the frontend is built with React. The web page allows users to upload an image from local files or take one with the phone camera first, and then the user can ask any arbitrary question regarding the uploaded image. The question asked will always be about the last image uploaded.

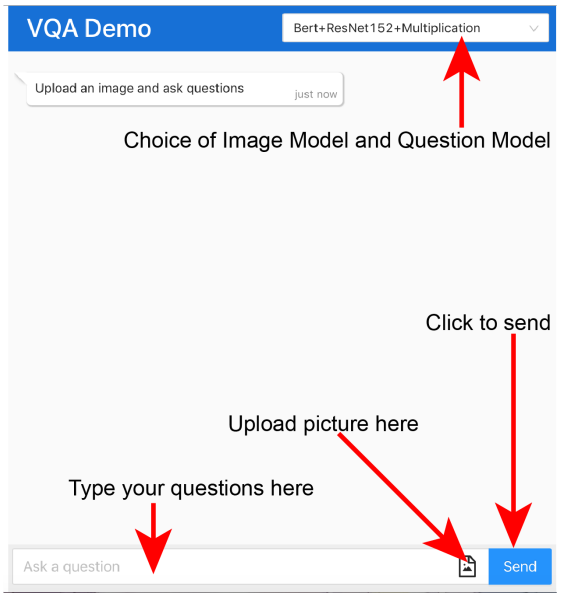

Figure 10. User Interface Instructions

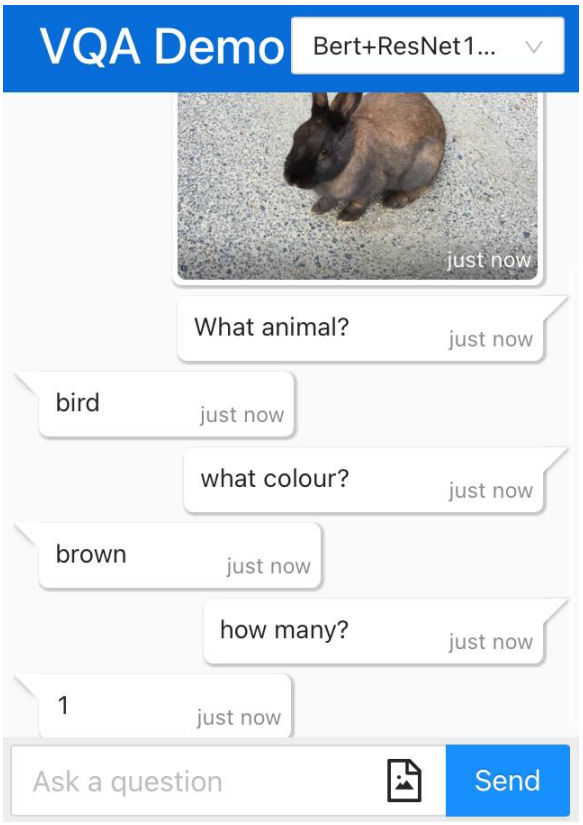

Figure 11. User Interface Screenshot

However, the model is not always able to give the correct answer, as shown in Figure 11.

The link to the GUI demo is here: https://vqa.hyh0.com/.

Conclusion

In conclusion, our solution to the VQA problem consists of using the pre-trained ResNet-152 and BERT to extract features from the images and text questions respectively, and then combining them in the Fusion scheme via element-wise multiplication. While it has achieved decent performance, with test accuracy of 49.7% and top-5 accuracy of 86.0%, there is still room for improvements.

Future Works

Our development is constrained by the memory limit of GPU server resources. The maximum batch size allowed is 32, which leads us to fewer choices of certain parameters. Usually, one complete training epoch took around ten thousand seconds, which constrains the number of epochs for training. Thus, there is still room for us to seek better results.

Due to the time limit, we are not able to do experiments on other possible algorithms. With more time allocated, one possible model we can try for our Question Model is Bag-of-Words (BOW). Another potential way for improvements is to augment the dataset and search for other public datasets. There are many existing public datasets besides VQA 2.0, such as Visual Genome [4] that contains 6 types of “W” (what, where, how, who, where, when) questions. These may help us better understand this topic.

As mentioned in the section of Comparison with SOTA, there are still lots of possible architectures that have been broadly used for VAQ problems. One example is Models Using External Knowledge Base, meaning to answer the question “How many eggs are there in the image”, we should know what an egg is first. The mechanism is shown in Figure 12:

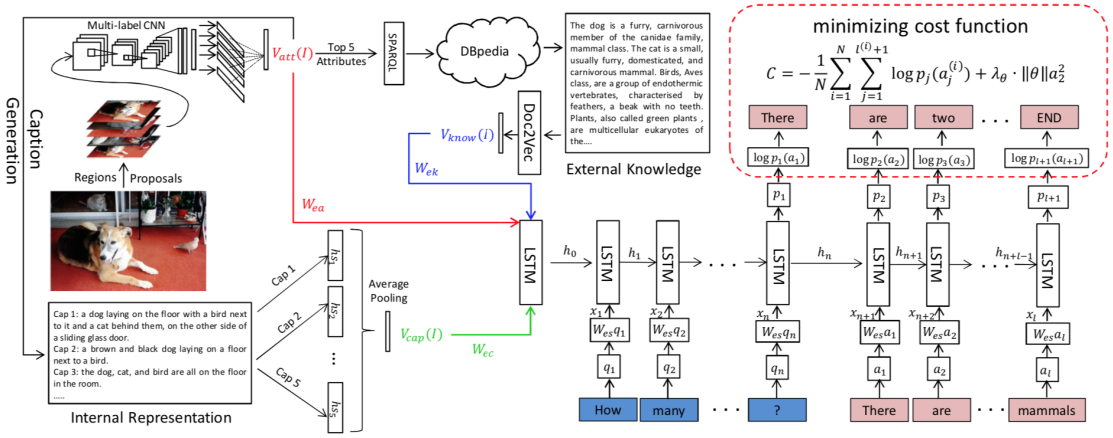

Figure 12. Model Using External Knowledge Base[5]

This model[5] first interprets the image in several attributes, then selects the top 5 related attributes to extract the corresponding data in Large-Scale Knowledge Bases, DBpedia, to further encode the answer using Dov2Vec, and also the model will generate and encode 5 captions to describe the attributes. Next, the model will put encoded attributes, encoded data from external knowledge and encoded captions to the Seq2Seq model together as initial states, to do the encoding and decoding and trained with MLE method. This method is able

to reach about 55.96% accuracy at the end and also provides an interesting view of the VQA problem that is worth trying.

Last but not least, the current natural language questions and answers are in English. In the future, we can explore some datasets encoded in other languages such as Chinese, or multiple languages, so that we can furtherly find out whether there is any difference in learning efficiency between different languages.

References

[1] Goyal, Y., Khot, T., Agrawal, A., Summers-Stay, D., Batra, D., & Parikh, D. (2018). Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *International Journal Of Computer Vision*, *127*(4), 398-414. doi: 10.1007/s11263-018-1116-0

[2] Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C., Parikh, D., & Batra, D. (2016). VQA: Visual Question Answering. *International Journal Of Computer Vision*, *123*(1), 4-31. doi: 10.1007/s11263-016-0966-6

[3] Singh, J., Ying, V., & Nutkiewicz, A. (2018). "Attention on Attention: Architectures for Visual Question Answering (VQA). doi: 1803.07724", *ArXive*, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.07724.

[4] Ranjay, K., Yuke, Z., Oliver, G., Justin, J., Kenji, H., Joshua, K., Stephanie, C., Yannis, K., Li-Jia, L., David A, S., Michael S, B., and Fei-Fe, L. (2017) Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int J Comput Vis* 123, 32–73. doi:10.1007/s11263-016-0981-7

[5] Qi, W., Peng, W., Chunhua, S. Anthony, D. Anton van den H. (2016) Ask Me Anything: Free-form Visual Question Answering Based on Knowledge from External Source. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Available: https://arxiv.org/abs/1511.06973

[6] Papers with Code - VQA v2 test-std Benchmark (Visual Question Answering). (2021). Retrieved 8 August 2021, from https://paperswithcode.com/sota/visual-question-answering-on-vqa-v2-test-std

Instructions to Run Code

Models

1. To train a new model, run `BERT_ResNet152_Mul.ipynb` (skipping cell "`Load pre-trained weight`") to create the model, train it on the VQA v2.0 dataset, and evaluate the model on the test set.
2. For reproducibility of our results, download the model weights from this link and save it to the `./models` folder: https://sutdapac-my.sharepoint.com/:u:/g/personal/yuhang_he_mymail_sutd_edu_sg/EbuC0ops4GFDu-Cp6QKKSMMBWBPk0X8a_ICa9YlPxzejww?e=tEvw3a
3. Run `BERT_ResNet152_Mul.ipynb` (skipping cell "`Train model`") to load the pre-trained weights of our best model before evaluating the model.

Dataset Visualization

1. Run `Dataset_Visualization.ipynb` to print the sizes of the dataset and visualize the dataset by getting graphs for class distribution and top 50 most frequent words in train, val, and test dataset.

User Interface

1. Download the model weights from this link and save it to the `./models` folder: https://sutdapac-my.sharepoint.com/:u:/g/personal/yuhang_he_mymail_sutd_edu_sg/EbuC0ops4GFDu-Cp6QKKSMMBWBPk0X8a_ICa9YlPxzejww?e=tEvw3a
2. Run `python -m ui.backend` to launch the backend service.
3. Open frontend folder: `cd ui/frontend`
4. Install frontend dependencies: `yarn install`
5. Start frontend: `yarn start`
6. go to http://localhost:3000 to see the demo web app.

Appendix

Please access our code files from our GitHub repository:
https://github.com/ZhangShaozuo/Artificial_Intelligence_VQA