Developing and Exploring Visual Question Answering

Sashank Kakaraparty
George Mason University
skakarap@gmu.edu

Madhukar Reddy Vongala
George Mason University
mvongala@gmu.edu

Sai Karthik Dindi
George Mason University
sdindi@gmu.edu

1 Introduction

Visual Question Answering (VQA) is a research field that combines Computer Vision and Natural Language Processing to create systems that can answer questions based on visual content. VQA has gained significant attention in recent years, with the potential to enhance various applications such as image and video captioning, scene understanding and robotic navigation. The ultimate goal of VQA is to develop algorithms that can understand visual content and accurately answer questions about it, similar to how humans interpret visual information. We propose the task of free-form and open-ended VQA. Given an image and a natural language question about the image, the task is to provide an accurate natural language answer.

Despite the recent advancements in VQA, the problem remains challenging due to the inherent complexity of both computer vision and natural language processing. Mirroring real-world scenarios, such as helping the visually impaired, both the questions and answers are open-ended. Visual questions selectively target different areas of an image, including background details and underlying context. As a result, a system that succeeds at VQA typically needs a more detailed understanding of the image and complex reasoning than a system producing generic image captions. Therefore, we aim to develop a robust, scalable and accurate VQA model that can understand and reason about visual content and natural language to answer questions with a high degree of accuracy and reliability.

2 Related Work

A Neural-Based Approach to Answering Questions about Images; (Malinowski et al., 2015) uses a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) with attention mechanisms to learn to attend to relevant regions of the image and words in the question. The model learns to represent the image and question features in a joint embedding space and uses the attention mechanism to attend to the most relevant parts of the image and question. The proposed model achieves state-of-the-art results on the VQA dataset and outperforms previous approaches that do not use attention mechanisms. The authors also demonstrate the interpretability of the model by visualizing the attended regions of the image and words in the question.

Visual Question Answering; (Agrawal et al., 2016) introduced the task of VQA and proposed a model that combines Convolutional Neural Networks (CNNs) for image processing and and a Long Short Term Memory (LSTMs) network for language processing. This model achieved state of the art results on the VQA dataset.

Hierarchical Question-Image Co-Attention for VQA; (Lu et al., 2017) proposed model selectively focuses on different regions of the image and words in the question in a hierarchical manner to improve the performance of VQA. The model uses co-attention mechanisms to learn the correlation between the image and question features and adaptively attend to relevant regions of the image and words in the question. The hierarchical attention mechanism improves the interpretability of the model by allowing it to attend to different levels of granularity in the image and question. The proposed model achieves state-of-the-art results on the VQA dataset and outperforms previous approaches that use attention mechanisms.

Transformer based models for VQA; (Gupta et al., 2022) proposed model, called GPV-1, uses a two-stream transformer-based architecture that processes both the image and the question independently. The image stream uses a pre-trained Faster R-CNN model to extract visual features, which are then fed into a transformer-based encoder. The question stream also uses a transformer-based encoder to process the textual input. The two streams are then fused using a cross-modal transformer-based decoder that generates the answer.

3 Evaluating GPV on VQA

As we are new to multi models, to get a better understanding of how these models work, We used the pre-trained GPV model and ran inference over it using the COCO and CLEVR dataset. The use of general purpose models to a single purpose model has always been a debate .
The idea of a single architecture being a solution to different problem statements fascinated us to choose the GPV. So, How does the model know what task it is supposed to do?, GPV always take in a image and text as input to output an image, text, bounding boxes, relevance score of bounding boxes and answer probabilities. The model uses text input to understand the task and generate output accordingly.

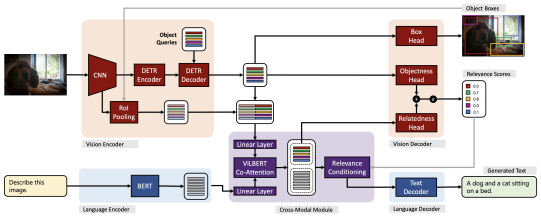


Figure 1: GPV Architecture

3.1 GPV Architecture

The GPV model has three major components : vision modules, language modules and cross contextual module.

Vision modules: The input image is first convoluted using the RESNET-50's convolution architecture and the feature maps obtained are fed into DETR's encoder, whose output along with the R (=100) object queries (These are learnt positional encodings that look for different objects in the image) are passed into the decoder to obtain the contextual representations. These object queries eliminate the need for non maximum supression as they guide the model in where to look for, to get the representations.
GPV uses DETR's box head to find R bounding boxes which are used for grounding and detection tasks as well as for RoI pooling from the CNN backbone. The DETR transformer features and the RoI pooled features are concatenated to get the final encodings.

Language modules: The language encoder is used to encode the task description. WordPiece tokenizer to obtain sub-word tokens for the language input and a pretrained BERT model to compute representations. The language decoder outputs words to classify, describe, or answer the input. At every step of the generation, the words generated so far are used to generate the next possible word.The output for a vocabulary word is obtained by taking dot product between the embedding vector output by the decoder and a linearly transformed BERT encoding of the word.

Cross contextual module: Region descriptors from the vision and language modules' sub-token representations are turned into equal-dimension vectors via linear layers before being supplied into ViLBERT's co-attention layers for cross-contextualization. The relatedness head predicts logits that show relevance of regions to the job description using the coattended region attributes. These logits are combined with the logits from the objectness head, and then a sigmoid activation converts them into region-relevance ratings. These relevance scores are applied to regions or bounding boxes to rank their importance for completing the assignment.

4 Building a VQA Solution

Once we explored the existing VQA models and took a deeper look at the General Purpose Vision (GPV) model's performance on VQA tasks, we headed into building our own VQA multimodal model from scratch to better understand the nuances and interactions behind VQA tasks.

In the implementation of this multimodal model for VQA, we employ two key components (Image and Text Encoder) and one choice (Fusion Strategy):

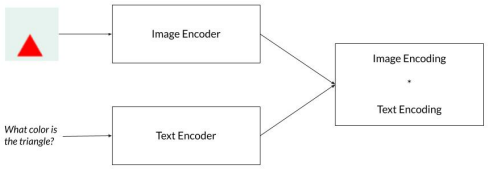
- For image encoding, popular choices included convolutional neural networks

(CNNs) like ResNet or VGG, and transformer-based models like ViT.

- For text encoding, popular choices included LSTM, GRU, or transformer-based models like BERT and GPT.
- Another key choice to make was the fusion strategy - Early fusion combines image and text features early in the model's processing pipeline, right after encoding, while late fusion combines the features at a later stage, usually just before the final prediction layer.

We initially built the network and tested it on the EasyVQA dataset, which consists of images containing simple colored shapes accompanied by simple questions. This allowed us to evaluate the model's performance on a relatively less complex visual question-answering task and refine its architecture accordingly.

In terms of fusion choices, we only attempted early fusion models, where once we encoded the image and text, we combined these embeddings before any training occurs.

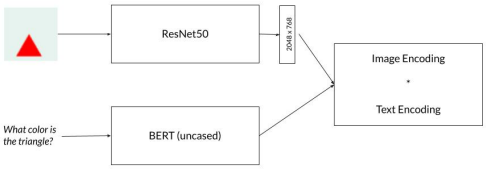


As a Text encoder, we use BERT uncased, which is a popular and powerful pre-trained model for natural language processing tasks. BERT is based on the transformer architecture and has been pretrained on a large corpus of text, enabling it to efficiently generate context-aware embeddings for input sentences. This allows our model to better understand the semantics and structure of the text input. However, in terms of image encoders, we explored both ResNet50 and ViT-21K, which required small changes in the architecture, and we'll detail them below.

4.1 ResNet50 as the Image Encoder

In our initial architecture, we experimented with using ResNet50, a well-known convolutional neural network, as a feature extractor for the images. ResNet provided a way to add more convolutional layers to a CNN without running into the vanishing gradient problem. ResNet50 extracts features from images by progressively reducing their spatial dimensions while increasing the number of channels. The resulting high-level feature representations are then combined with the text embeddings using the early fusion strategy as described earlier.

Here's the architecture when using ResNet50 for image encodings -



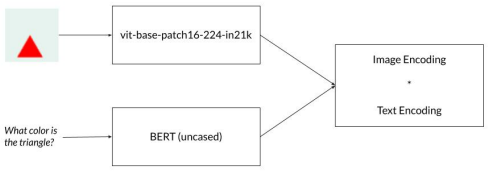
As we observe, there is an additional fully connected layer between the ResNet embeddings and the combining stage, that is because of the output dimensions of our image and text encoders.

- ResNet50's feature extractor outputs 2048-dimensional vector.
- BERT uncased outputs a 768-dimensional vector.

Hence, we needed to add a fully connected layer at the end of the image encodings, to reduce the dimensionality to 768 and match those of BERT, since it would make the combining process smoother, as will be described in detail in the join feature representation subsection.

4.2 Visual Transformer as the Image Encoder

Another variant of our multimodal architecture was to use a Visual Transformer as the image encoder, where we use the ViT-base-patch16-224-in21k model, a variant of Google's Visual Transformer (ViT) architecture with 21k pre-trained weights. This version of ViT has a base configuration and processes images by dividing them into non-overlapping 16x16 patches, flattening them into vectors, and feeding them into a transformer architecture with 224 tokens. This transformer-based approach enables ViT to capture both local and global information in images, which is crucial for answering questions based on visual content.

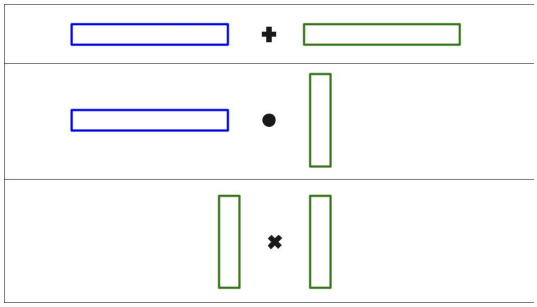


As we observe that the fully connected layer is out, that's because the ViT's output is a 768-dimensional vector as well, which means no need to reduce/increase dimensionality.

We will use the ViT variant for most of the experiments and a part of the best performing models, because it seems to perform better in our use cases, which will also be shown in the results section.

4.3 Joint Feature Representation

Now that we have the image features and tokenized question features, there are a few options to fuse the vectors together.



One of the most basic ways of handling different feature vectors is to concatenate the two, allowing subsequent layers to determine appropriate weights for each. When the features are of the same length, we can apply element-wise addition or multiplication. (Shih et al., 2016) experiments with a dot product of the two vectors, and (Battaglia and Prato, 2018) adopt a hybrid method that concatenates the outcomes of both element-wise multiplication and element-wise addition.

However, (Malinowski et al., 2016) experimented with these methods and found that element-wise multiplication gives the best results.

Hence, in our architecture, we chose to fuse the feature vectors using element-wise multiplication, resulting in a new 768-dimensional vector.

5 Datasets

We have used three datasets: MS COCO VQA, CLEVR, Easy VQA.

5.1 MS COCO VQA

MS COCO stands for Microsoft Common Objects in Context. MS COCO VQA consists of 205k images of everyday objects and humans. Each image is has three human generated questions and their corresponding answers. The entire 205k images are split into 82,783 train, 40,504 validation and 81,434 test images.

Figure 2: Example of MS COCO VQA dataset

5.2 CLEVR

CLEVR (Compositional Language and Elementary Visual Reasoning) is a diagnostic dataset. It contains images of 3D rendered objects and tests a range of visual reasoning abilities. It contain minimal biases and has detailed annotations describing the kind of reasoning each question queries. It has total 100k images with 10 questions per image and their corresponding answers. The entire dataset is split into 70k train, 15k validation and 15k test images.

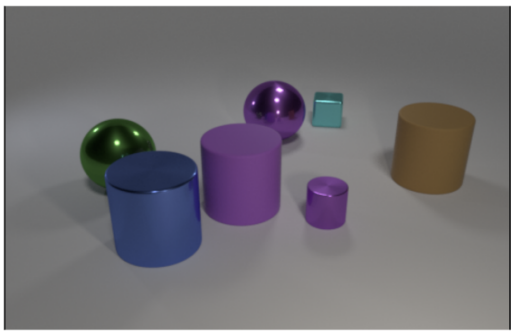


Figure 3: Example of CLEVR dataset

5.3 Easy VQA

Easy-VQA contains 4000 train images with 38,575 train questions and 1000 test images with 9673 test questions. There are only 13 possible answers. Out of all the available train questions 28,407 questions are yes/no and out of all the test questions, 7,136 questions are yes/no.

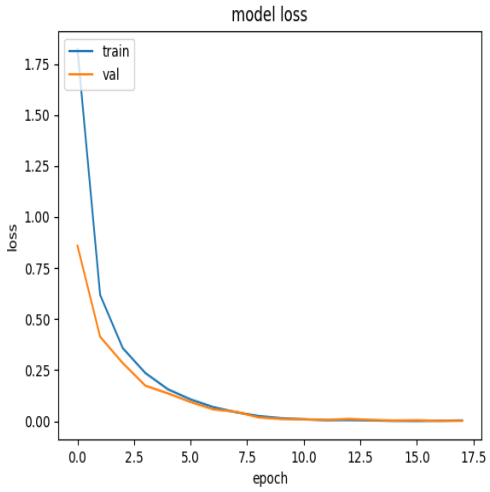
Figure 4: Example of Easy VQA dataset

Questions:
What color is the rectangle?
Does the image contain a triangle?
Is no blue shape present?
What shape does the image contain?

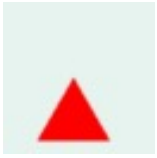
6 Experiments and Results

6.1 EasyVQA Results

EasyVQA was the dataset we initially built our network, refined our architecture and code. We hit a 98% test accuracy on the VQA dataset with our best architecture. Here's a loss plot of the best model, which is Experiment 3 on the chart.



Input Image:



Input Question: What is the color of the triangle?

Output: Red

6.2 MS-COCO x GPV Results

We have obtained quite interesting results. When we tested images from COCO with questions manually entered by us, the model performed very well. To test out the explainability of the model i.e to see what parts of images is it using to generate the answer. To our surprise, although the model answers the question perfectly, the bounding boxes generated are way off of what was expected of it.

Our intuition as to why this is happening is, during training they used hungarian loss for localization task to optimize the bounding boxes and negative log-likelihood loss for other tasks like VQA, captioning and Categorization. Incorporating the Hungarian loss into other tasks during training might solve this problem.

Input Image:

Figure 5: Input image to GPV

Input Question: What is the color of the horse?

Output: Brown with 0.998 confidence

Figure 6: Bounding box from GPV

Figure 7: Bounding box from YOLOv5

From the figures 6 & 7, we can observe that the bounding boxes obtained from GPV are very different from the bounding box from YOLO to detect the horse. But still GPV is able to answer it correctly because there is brown in the detected portion.

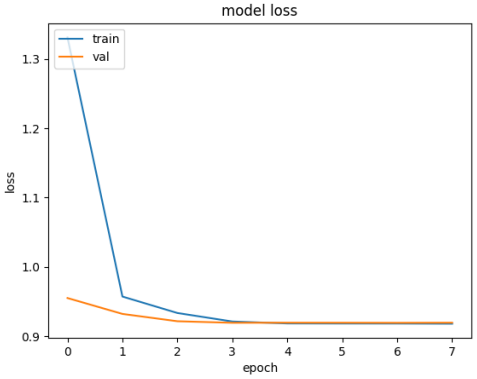
6.3 CLEVR Results

This is where things get into serious VQA problems, as we trained our best EasyVQA architecture on the CLEVR Dataset, which is more complex and requires more knowledge of the image to answer the questions.

We used the architecture and parameters in Experiment 3 of Table 1. And trained it on the 800,000 records of the training section of CLEVR, which accounts to around 80K images with each image having 10 questions. Our validation dataset contained 150,000 records (15K images). Each epoch took 6 hours to run and was a time taking endeavor.

Our best accuracies on the training and validation set end up at: 53% on Training 52% on Validation

Attaching a plot of the loss on the complete dataset:



Based on the results from the epochs that we saw unfold, we kept making changes to our model, and started running them on a sample of our data which was 100,000 records which is 1/8th of the original data. We have reported the accuracies on those datasets, but it's clear it seems to perform just as well, which could be because it is not exposed to the entirety of the data.

6.3.1 GPV vs Our Model on CLEVR

We created a subset of the testset, just 1000 questions, 100 images to test the performance of our model vs GPV.

Here are the results from our model's performance: 52% accuracy on the test set, which is in line with the validation accuracies we saw on the data.

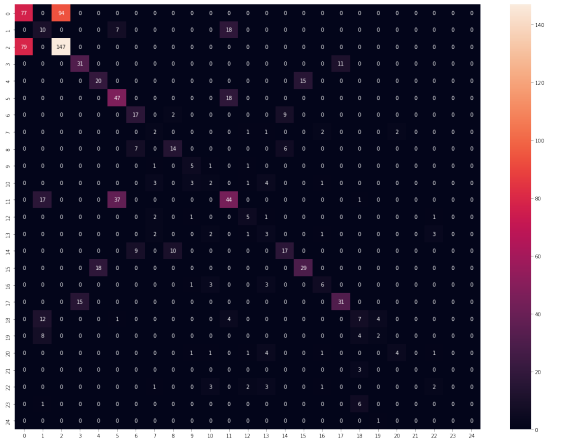


Figure 8: Confusion Matrix of Our model on the test set

We ran GPV on the same sampled test set, and it returned an accuracy of 29.6%, with as many as 63 unique answers.

Hence, we started digging into the predictions to see what was going on and observed that in some cases GPV had responded with predictions which could be synonymous, but not the exact same as our labels. Hence, we worked on replacing these synonymous labels to give it a fair shot at improving accuracy. For instance, we used logics such as ["circle", "ball", "round", "oval"] all referred closely to the label "sphere", so we converted them to "sphere". We applied similar logics to ["big", "giant"] being "large" and also colors like "silver" meaning "gray" in our labels.

After making such changes, we were able to reduce the unique answers down to 53. We focused on replacing answers that had occurred more than

20 times (2% of our test data), which contributed to 935 of the 1000 predicted labels. One of the labels that was extremely tricky to replace was “plastic” which was an answer to questions in CLEVR that asked for the “type” of the material. We had to replace “plastic” with either “metal” or “rubber”, so we went down to manual analysis and learned that, ”plastic” occurs 69 times, out of which the ground truth is ”metal” 35 times, rest 34 times it is ”rubber”. Which meant, the test results would suffer anyways, so we just converted it to “metal”.

After all this clean up, GPV solutions had an accuracy of 35% on the test set sample, behind our model which had 52%.

Exp	Architecture Details	CLEVR Acc.	EasyVQA Acc.
1	ResNet50, FCNN: 3 Layers, 2 BatchNorm, ReLu	-	Tr: 84%, Val: 82%
2	ViT, BERT using avg. pooling layers	-	Tr: 89%, Val: 88%
3	BERT using CLS Token	Tr: 53%, Val: 51%	Tr: 99%, Val: 99%
4	Reduced image size by 8 and only 100,000 records.	Tr: 52%, Val: 48%	-
5	Deeper Network, 50,000 records, T5 as encoder	Tr: 51%, Val: 50%	-

7 Conclusion and Future Work

Although our model performs better on the CLEVR set, it is because of the specificity of the dataset, and the discriminative/classification nature of our model versus the generative nature of GPV. Since our model is discriminative in nature, and is trained on the dataset, learning the specificity of the task has an internal advantage. We notice that GPV particularly struggles with questions about the “material” of the objects, which are close to 10% of our test set. In our brief analysis, we also noticed that it seemed to struggle with “relational question” such as “Does the object beside the green cube, of the same size, blue in color?” which is one of the complexities of CLEVR. Furthermore, during our testing and exploration of GPV, we observed that although GPV responded with the correct answers it failed in detecting the bounding boxes perfectly. Hence, arises a question about the explainability of GPV.

In conclusion our model performs up to 52% on unseen data, which is quite decent, but not anywhere close to the best performing model on CLEVR, which boasts close to 95% accuracy.

Future work on this also includes experimenting with newer approaches in embedding techniques, fusion techniques and the classifier, which all could improve performance, along with an error analysis of our model, to understand where it might be straying. Overall, this was a great learning experience, working with multimodal models.

References

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. 2016. Vqa: Visual question answering.

Fiammetta Battaglia and Elisa Prato. 2018. Nonrational symplectic toric cuts. *International Journal of Mathematics*, 29(10):1850063.

Tanmay Gupta, Amita Kamath, Aniruddha Kembhavi, and Derek Hoiem. 2022. Towards general purpose vision systems.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2017. Hierarchical question-image coattention for visual question answering.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2016. Ask your neurons: A deep learning approach to visual question answering.

Kevin J. Shih, Saurabh Singh, and Derek Hoiem. 2016. Where to look: Focus regions for visual question answering.