



Vision is Language: Visual Understanding via LLM

Xiangyu Liu

jxiangyu@stanford.edu

Department of Computer Science, Stanford

Introduction and Problem Statement

When we hear “sky is blue”, we can immediately picture a beautiful blue sky above with a few white clouds in our mind. Similarly, when we see a dog chasing after a ball, we can effortlessly describe which is doing what. Hence, intuitively, there is a large overlap between vision understanding and language understanding. Indeed, in the recent scientific studies, it was found that brain neural overlap across visual and auditory modalities. Therefore, to have human-like visual understanding, the machine must also have human-like language understanding.

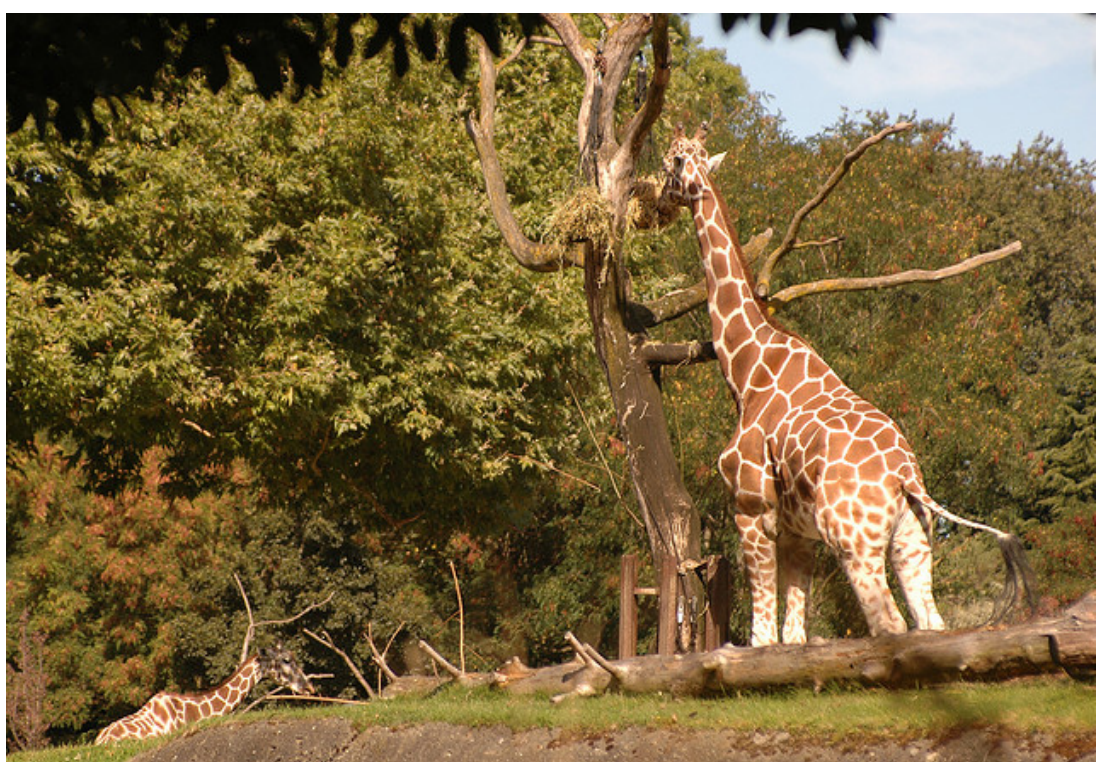
One important area of vision and language understanding is Visual Question Answering (VQA), where a machine is given an image and a question related the image and its task is to produce an answer to the question. Many previous works have leveraged vision-language pretraining and have achieved very high accuracy in VQA tasks. However, most of them treat the VQA as a classification problem and have the model to predict the correct answer from a closed vocabulary. The current state-of-the-art model has reached 84.3% accuracy on the VQA test set. It is obvious that there is still room for model improvements here.

In this project, I explore leveraging both pre-trained language and vision models in my model to perform the VQA task.

Dataset

Input: Images, Captions, and Open-ended Questions

The input I use is directly from visualqa.org. It contains 82,783 training images with 400K+ questions and 81,434 test images with 400K+ questions. Here is an example of the question, answer, image tuple.

	Question	Answer
	What is in front of the giraffes?	tree
	What do these giraffes have in common?	eating
	Where is the giraffe?	near tree

Captions		
	A giraffe eating food from the top of the tree.	
	A giraffe standing up nearby a tree	
	A giraffe mother with its baby in the forest.	
	Two giraffes standing in a tree filled area.	
	...	

Table 1. A VQA Training Data Example

Each image can contain different number of captions and different number of question-answer pair.

Output: Open-Ended Answer prediction

The output of the model is to predict the answer for a given image-question pair.

Methods

Model

In order to incorporate both image and language understanding into the model, I leveraged a pre-trained Mask R-CNN model to extract image features and a BERT to extract text embeddings for captions associated with the image. A MLP layer is then used to project the image features onto the text embedding space. Its output is treated as the memory to the transformer decoder layer to predict answers based on the given question. The overall architecture is shown in figure 1

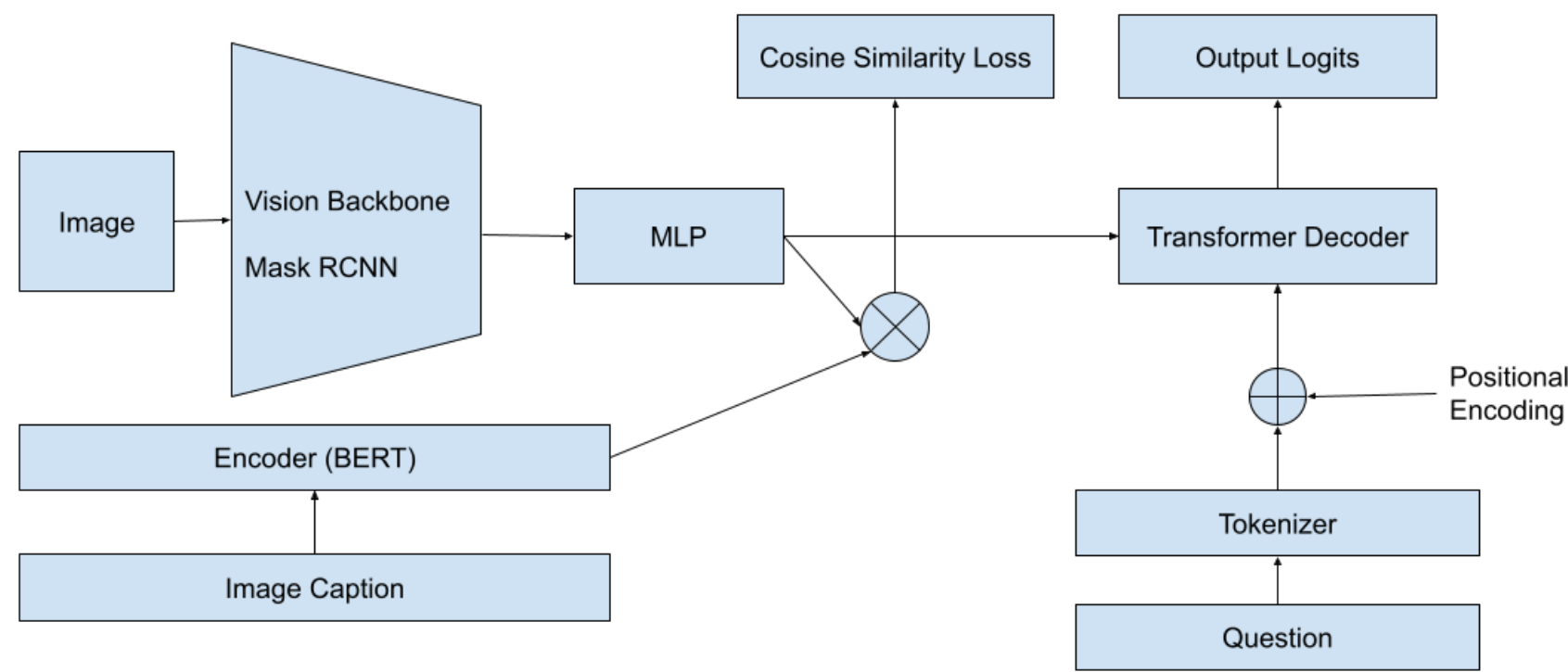


Figure 1. VQA Model Architecture

Loss Functions

Caption Embedding Similarity Loss: In order to ensure the image feature projection alignment with the text embedding, I leveraged a similar idea as the contrastive learning loss. But instead of providing both positive and negative pairs, I use a cosine similarity to ensure that the image feature projection aligns all the caption embeddings.

$$\text{sim}(x_1, x_2) = 1 - \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon)} \quad (1)$$

$$L_{cap}(img) = \frac{1}{M_c} \sum_{c \in caps} \text{sim}(Embed_c, Embed_{img}) \quad (2)$$

where M_c is the number of captions associated with the image. $Embed_{img}$ is the embedding projection of the image.

QA Loss: Since the answer prediction is structured as next token prediction tasks performed by the transformer decoder. The QA Loss is simply the cross-entropy loss.

$$L_{qa}(img) = \frac{1}{M_{qa}} \sum_{qa \in qas} \text{cross_entropy}(logits, target) \quad (3)$$

where M_{qa} is the number of question-answer pairs associated with the image.

Final Loss Function:

$$L = \frac{1}{N_b} \sum_{i \in N_b} \gamma * L_{cap}(img_i) + L_{qa}(img_i) \quad (4)$$

γ is a hyper-parameter to control how much weight the caption embedding loss should be counted for in the final loss. N_b is the batch size. (Worth noting that the N_b , M_{qa} , and M_c are different.)

Experiments and Results Discussion

Hyperparam: I've tested 3 different values for the γ : 0.9 (with "vqa_with_caption" label), 0.5 (with "vqa_with_0_5_caption" label) and 0 (with "vqa_no_caption" label).

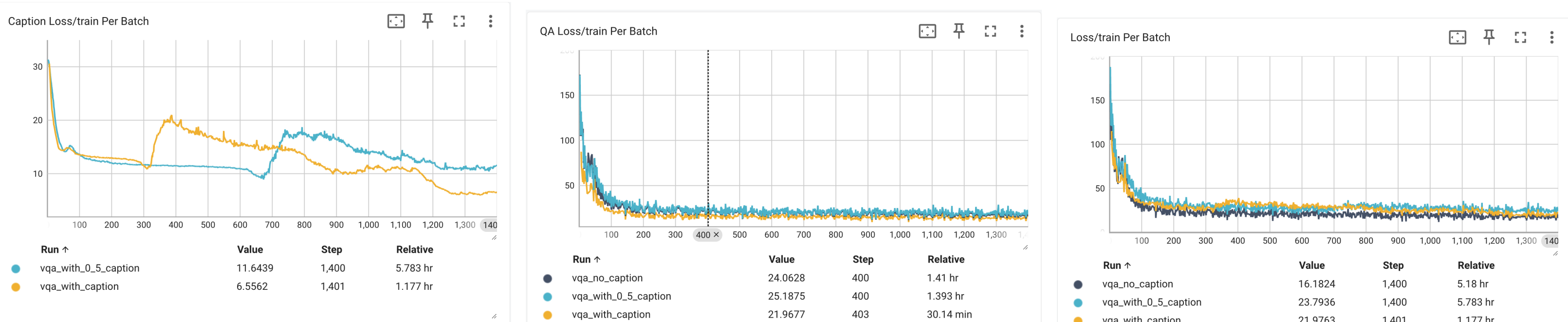


Table 2. Gamma hyperparam tuning

When $\gamma = 0.9$, L_{qa} is lower when compared with the L_{qa} s from $\gamma = 0.5$ and $\gamma = 0$. This clearly indicates using the caption embedding as part of the loss to train the network has a significant advantage in obtaining lower training loss.

Pretrained Models and Convergence: For the training size of 100K images, it takes about 3700 steps to process 1 epoch with batch size of 32. However, by leveraging the pretrained Mask R-CNN and Bert models, training converges after step 1400, as the train loss no longer decreases. This again proves that reusing the pretrained foundational models is very effective in training and significantly reduces the training time.

Validation Accuracy: The validation set contains 26333 total questions. The accuracy on the validation set is computed based on naive string comparison. The result is listed in the table below:

γ	Total Questions	Accuracy (in %)
0.9	26333	99.28
0.	26333	99.28

Table 3. Validation Set Accuracy

Surprisingly, the model trained with $\gamma = 0.9$ achieves the same accuracy as the model trained without using the caption embedding loss.

test-dev Accuracy: I was able to submit my test-dev results using $\gamma = 0.9$ to eval.ai and obtain an official score for my model:

Question Type	yes/no	number	other	overall
Accuracy (in %)	61.72	29.31	7.22	32.18

Table 4. eval.ai test-dev Accuracy for 0.9 gamma

Loss Examples: To understand the inaccuracy, I further examined the inaccurately predicted answers and compared them with the expected answer from the validation set. Here are some examples:

Expected	Predicted
.25	. 25
6:56	6 : 56
keep elephants out/in	keep elephants out / in

Interestingly enough, all the losses in validation are about the format of the output. The model doesn't seem to be able to understand the semantics of the formatting based on the questions. Upon further investigation, it was caused by the tokenizer. My model uses a frozen Bert tokenizer with "google-bert/bert-base-uncased" weights. When it decodes the token, it adds white spaces after symbols like ".". If I have the tokenizer to tokenize ".25", and immediately pass the output to the tokenizer to decode, it returns ". 25" instead.

Future work

- Different Tokenizer:** The tokenizer used by BERT model was not able to identify the language nuances during the decoding process. To properly capture the right formats, a new tokenizer needs to be implemented.
- Loss Analysis:** Given the low accuracy on the "number" and "other" category, it's important to further analyze some examples and identify additional gaps.
- Larger Image Backbone and/or Decoder Layer:** "yes/no" question accuracy suggests that either the image backbone cannot extract enough image features or the decoder layer does not have large enough parameters. Using a larger model should improve the accuracy for "yes/no" questions.