Vision is Language: Visual Understanding via LLM

Xiangyu Liu
Stanford
jxiangyu@stanford.edu

Abstract

Many vision tasks can benefit significantly by leveraging language. Some tasks, such as Visual Question Answering (VQA) for open-ended questions, require the vision model to understand language. In this project, I designed a model leveraging both frozen image model and frozen large language models to achieve high accuracy VQA results.

1. Introduction

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.

As the Large-Language Model has exhibited promising capability recently, research to leverage LLM to improve image understanding has witnessed a rapid advancement. One fundamental capability of visual understanding is to perform visual question answering (VQA). Many different approaches [such as (12), (4)] that leverage vision encoder and LLm for VQA tasks have been explored to improve the overall accuracy of VQA tasks.

However, the open-ended VQA remains a very challenging task, because it not only involves many traditional vision tasks, such as object segmentation, but also requires the natural language understandings. More importantly, it needs to be able to identify the connection between vision tasks and the language in order to answer vision related questions.

In this paper, I plan to use the VQA and CoCo datasets to train and evaluate the accuracy of the neural network for VQA performance, with the focus on the open-ended questions instead of the True/False questions or multiple-choice questions. More explicitly, the input to my model is an image and a question related to the image, the output will be the answer to that question.

2. Related Work

Transformer (11) has become a very popular and powerful method for language tasks. For example, both GPT (10) and BERT (5) used transformer as its architecture and achieved great performance. Because of its effectiveness, many explored using transformers in vision tasks and had seen similar success (6). As a result, there are various approaches to use transformer to jointly model vision and language (1), (12).

With Transformer's success, pre-training for vision-language tasks has become popular. One effective approach for vision-language pre-training is contrastive learning (9), which allows joint-training a model for both vision and language (12).

The current state-of-the-art approaches for Visual Question Answering leverage the vision-language pre-training and have achieved very high accuracy in Vision Question Answering tasks (4), (12), (3). However, they all treat the task as a classification problem, which still limits the model's capability to achieve human-like results.

3. Methods

3.1. Architecture

For a model to properly answer the open-ended questions for any given image, it needs to understand both language and image, and how they overlap and interact with each other. Given that the Large Language Model is extremely effective in encoding the semantic meaning of language, projecting the vision features onto the same space can align the vision feature and the language embedding and further allow the output prediction.

Hence, I've design a simple model, as shown in Figure 1. It uses a vision backbone to extract the image features including spatial features, then those features will pass through a MLP layer to project onto the text embedding space. Then a multi-head transformer decoder layer is used to predict the answer based on the question embedding and the projected image embedding.

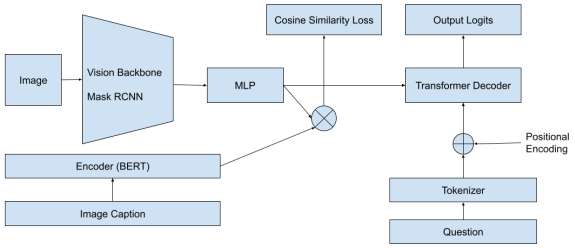

Figure 1. Model Architecture

3.2. Vision Backbone

In order to understand the full context encoded in the image, the vision model needs to not only classify the image, but also detect and segment objects. Therefore, I use a frozen Mask R-CNN network (7) in my model.

Since the vanilla Mask R-CNN network also includes Region Proposal Network (RPN) and PoI layer to detect the bounding boxes of object, which are not necessary for QA tasks as long the features are extracted, I removed both RPN and RoI layers and only use the image backbone of the Mask R-CNN network as the image backbone in my model.

3.3. Image Projection

The frozen vision backbone produces a $256 * 13 * 13$ output. It is then flattened to a 1-D tensor and passed through a Multi Layer Perception (MLP) layer to project onto the text embedding space.

The MLP layer consists of 2 linear layers with 2048 as the hidden layer size and LeakyReLU as the activation function.

3.4. Caption Embedding and Similarity Loss

I used the "bert-base-uncased" BERT model (5) to define text embedding space. For the model to correctly capture the semantic meaning of a given image, its text embedding projection must have large cosine similarity with the embeddings of the captions from the same image.

Because the closer to 1 the cosine similarity is, the more aligned the embeddings are, to make sure it works with the learning algorithm, the cosine similarity is inverted and shifted by 1:

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

With that, the similarity loss function is defined as the following:

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

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

3.5. Open-Ended QA, Transformer Decoder, and QA Loss

The QA task is structured as the next token prediction task performed by a Transformer Decoder layer. The projected embedding from the image is used as the memory (i.e. key) of the Transformer Decoder. The question and answer are processed and fed into the transformer, which will produce the next output logit.

Similarly, the QA Loss is structured as the next token prediction loss using cross entropy loss function:

$$L_{qa}(img) = \frac{1}{M_{qa}} \sum_{qa \in qas} cross\_entropy(logits, target) \quad (2)$$

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

3.6. Final Loss Function

The final loss function can now be defined as the following:

$$L = \frac{1}{N_b} \sum_{i \in N_b} \gamma * L_{cap}(img_i) + L_{qa}(img_i) \quad (3)$$

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.)

4. Dataset and Features

The dataset I used to train the model is directly from Coco (8) images with questions and answers annotation from VQA (2). The training data set with over 100K images, 400K questions, a test data set with 40K images and 107K questions. Each image can contain different number of questions and answers. The following image in Figure 2 is an example of what 1 training data might look like:

4.1. Image processing

The images in VQA come with different sizes. To make sure the images can be processed in batch, all images are padded to size 640 X 640 and uniformly scaled down to 320X320, mostly to reduce the computational resource usage.

```
q/a: What is in front of the giraffes? tree
q/a: What do these giraffes have in common? eating
q/a: Could this photo be from a zoo? yes
q/a: Are the animals eating? yes
q/a: Where is the giraffe? near tree
q/a: Is there a zebra? no
q/a: What is the giraffe standing behind? tree
q/a: Is the giraffe eating the tree? yes
q/a: Are both giraffes standing? no
q/a: Are they at a zoo? yes
q/a: What is on the ground next to the giraffe on the right? log
q/a: Are some of the trees dead? yes
q/a: Are any of the animals eating? yes
q/a: Is the giraffe in the shade? no
q/a: Are these giraffes living free range? yes
q/a: How many giraffes are there? 2
q/a: Is there a rock near the giraffe? no
q/a: How many animals are in this photo? 2
```

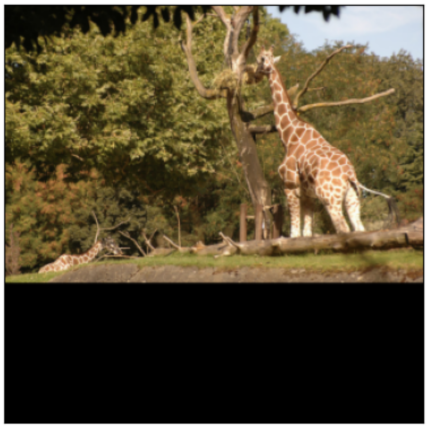

Figure 2. A training data example

4.2. Tokenization

I used the frozen Bert tokenizer with *google- bert/bert-base-uncase* weights with a vocabulary size of 30,000. All texts are tokenzied with 'max_length' as the padding option and 32 as 'max_length'

All captions of the images in the same batch is padded to the same length for tokenization.

In order for the transformer to identify the question and the answer parts, 3 special tokens are added to the tokenizer:

- *[QUESTION]* token: indicating the beginning of the question.
- *[ANSWER]* token: indicating the end of the question and as well as the beginning of the answer.
- *[END]* token: indicating the end of the answer.

During training, all questions and their corresponding answer are processing into 1 sentence in the following format: "*[QUESTION]* question *[ANSWER]* answer *[END]*".

All questions in the same batch are tokenized with padding to ensure the same shape in the output tensor.

4.3. Batch Size

I use a mini batch of 32 images, mostly due to the resource constraints. However, each images might contain different number of captions and different number of qa tuples, the text encoder and transformer decoder have different input sizes.

4.4. Question Answering

After training, the caption embedding part of the model is no longer used. To answer a question for a given image, I sample the model output logits until the special token *[END]* is sampled.

5. Experiments and Results

5.1. Training Losses and Hyper-params

I've compared different optimizers' and different γ's impact on training. For each experiment, I plotted per batch training loss for L_{cap}, L_{qa}, and L.

5.1.1 Different Optimizers

I started an Adam optimizer with $1e^{-3}$ as the learning rate, 0.9 as β_1, and 0.99 as β_2, using $\gamma = 0.9$ in the final loss function 3. I also tested a SGD optimizer with 0.1 as the learning rate, 0.9 as the momentum, with the same $\gamma = 0.9$ value. Both uses batch size of 32.

However, the Adam optimizer was not able to converge. The loss became nan after 1000 steps of training.

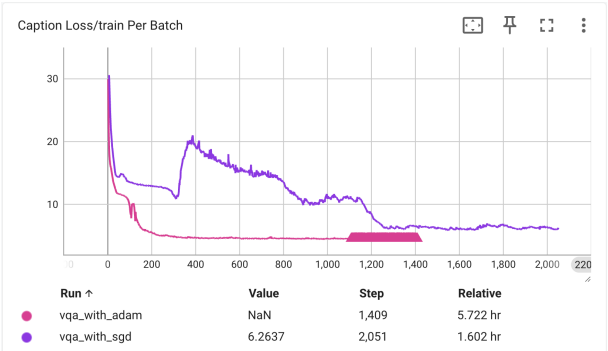

Figure 3. Caption Loss Comparison between Adam and SGD

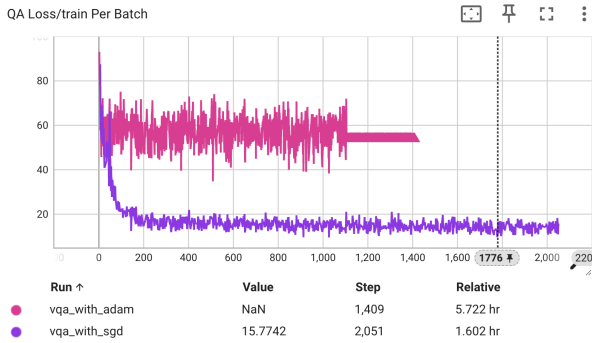

Figure 4. QA Loss Comparison between Adam and SGD

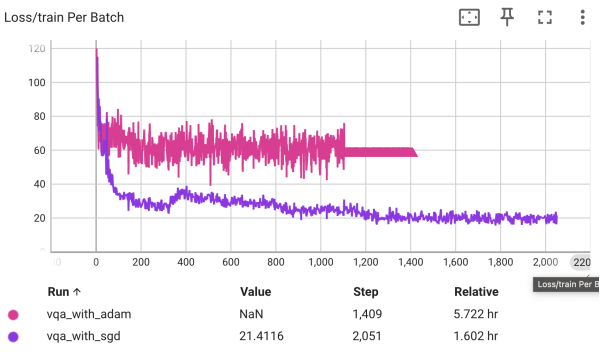

Figure 5. Total Loss Comparison between Adam and SGD

Interesting, the Adam optimizer perform very well on caption loss in the beginning, but does very poorly on QA loss. I believe this is because Adam optimizer accumulates the past momentum in order to update the step size. However, in the large language model and vision space, large step size often tends to overshoot. Furthermore, for caption prediction, the Adam works better because the gradient of the caption embeddings tend to align, but for QA, the question embedding can often be different from answers embedding, thus Adam optimizer tends to under-perform.

5.1.2 Different γ

To further understand the impact of the caption embeddings loss, I trained the model with 3 different set of γ values: 0.9 (with "vqa_with_caption" label), 0.5 (with "vqa_with_0_5_caption" label) and 0 (with "vqa_no_caption" label).

All are trained with batch size of 32 and a SGD optimizer with 0.1 as the learning rate, 0.9 as the momentum.

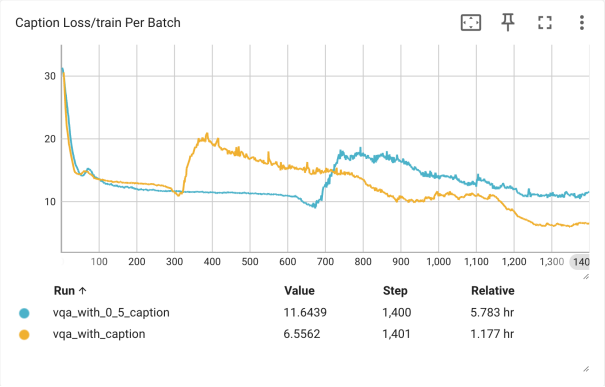

Figure 6. Caption Loss Comparison between different gamma values

Because when $\gamma = 0$, there is no caption loss, the caption loss in figure 6 does not have any data for "vqa_no_caption".

Recall that in 3, the total loss is defined as: $L = \frac{1}{N_b} \sum_{i \in N_b} \gamma * L_{cap}(img_i) + L_{qa}(img_i)$. The loss of interest is the loss for the question and answers, i.e. L_{qa}. The L_{qa} when $\gamma = 0.9$ is lower than both L_{qa}s when $\gamma = 0.5$ 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.

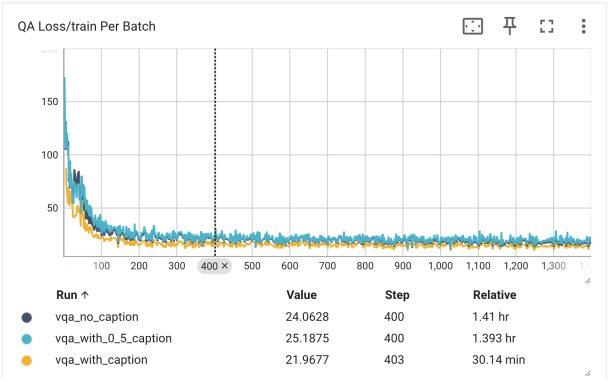

Figure 7. QA Loss Comparison among different gamma values

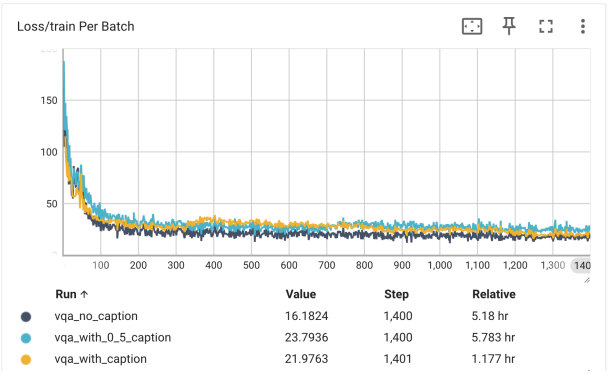

Figure 8. Total Loss Comparison among different gamma values

5.1.3 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.

5.2. Accuracies

Due to the resource constraints, running the entire test set was not feasible. Because the online eval rule in eval.ai has changed, partial results are not accepted. Therefore, I have to fall back to use the VQA validation set as my test set and examine the accuracy based on a naive string comparison between the predicted answer and the real answer.

I use a batch size of 32 images using the validation dataset, which includes 26333 total questions. (Recall that

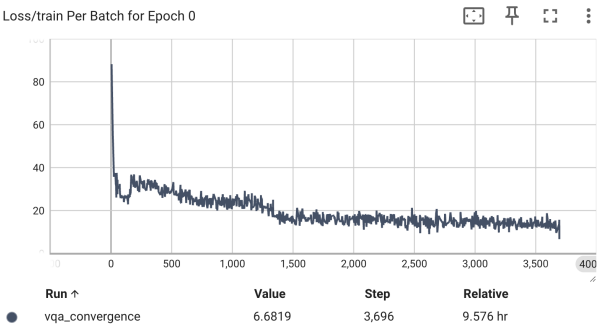

Figure 9. Total Loss per batch for epoch 0

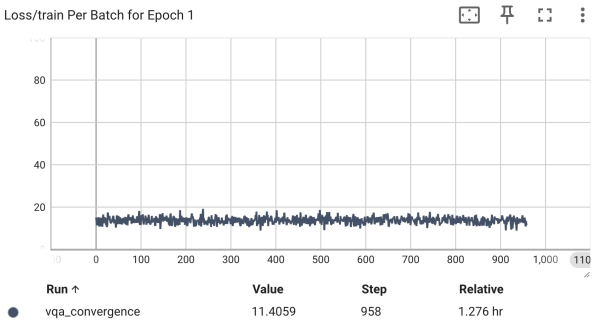

Figure 10. Total Loss per batch for epoch 1

each image can have different number of questions.) The accuracy result is listed in the table below:

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

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

5.2.1 Losses

To understand the inaccuracy, I further examined the inaccurately predicted answers and compared their with the expected answer. 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 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. E.g. when answering "how much is this donut?", it needs to be ".25", not ". 25". Similarly, when answering "what time it is", the model should output "6:25", not "6 : 25".

It turns out that this behavior is caused by the tokenizer. The model uses a frozen Bert tokenizer with "google-bert/bert-base-uncased" weights. When it decodes the token, it adds whitespaces after symbols like ".". If I simply have the tokenizer to tokenize ".25", and immediately pass the output to the tokenizer to decode, it returns ". 25" instead.

6. Conclusion and Future Work

As a conclusion, leveraging the pretrained vision model brings significant benefit in training, and using caption embedding as part of the loss improves the training losses. However, the model trained without using the caption embedding loss achieves the same accuracy as the model trained with $\gamma = 0.9$. It would be interesting to further investigate why the caption embedding loss didn't help.

Moreover, the main loss seems to be from the token decoding. It would be important to further experiment how to improve the tokenizer in order to achieve 100% accuray on the validation data.

Lastly, due to the resource constraint and the technical limitations, I weren't able to use any LLM with larger parameter size, such as Llama 2 or Gemma. Nor was I able to finish running over the entire test set. It would be exciting to try how using decoder-only LLMs can improve the accuracies of VQA and how the model compares to the other state-of-the-art models on the VQA performance.

References

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[3] H. Bao, W. Wang, L. Dong, Q. Liu, O. K. Mohammed, K. Aggarwal, S. Som, and F. Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint arXiv:2111.02358*, 2021.

[4] X. Chen, X. Wang, S. Changpinyo, A. Piergiovanni, P. Padlewski, D. Salz, S. Goodman, A. Grycner, B. Mustafa, L. Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn.

In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[10] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[12] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*, 2022.