VR Mini Project 2 : Multimodal Visual Question Answering and Finetuning with LoRA

IMT2022017 Prateek Rath
IMT2022076 Mohit Naik
IMT2022519 Vedant Mangrulkar

May 18, 2025

Contents

1 Introduction

This assignment involves creating a Visual Question Answering (VQA) dataset using the Amazon Berkeley Objects (ABO) dataset, evaluating baseline models such as BLIP and ViLT, fine-tuning using Low-Rank Adaptation (LoRA), and assessing performance using standard metrics. The Github repository containing all code and data, along with steps to run the code and reproduce the results, can be found here.

2 Dataset

We are using the Amazon Berkeley Objects Dataset, which contains around 150k Amazon product listings with multilingual metadata and around 400k catalog images.

- The listings metadata contains information like the product name, category, brand, color, keywords associated with the product etc.
- The images are variable-sized catalog images with a single item on a white background.

3 Preprocessing and Curation

Around 85% of the 398212 images are mobile phone cases and covers. The rest 15% come from a variety of categories such as clothing, shoes, home items, electronics, furniture, accessories, kitchen and groceries etc., as shown in the image below.
We first made a dataset of selected images along with cleaned up metadata, for which we filtered the data (images + listings) as follows:

- Ignore all products which have non-English non-ASCII metadata.
- Ignore all products which do not have metadata for product name, type, color and keywords.
- Consider all remaining products which are **not** mobile phone covers (around 9500), and then consider mobile phone covers to bring the total number of images to 10000. This was done to offset the huge imbalance in the number of mobile phone cover listings compared to other products.
- Resize all images to 256×256 for consistency, sort by filename and save the images to a separate folder, while saving the filename, name, product type, color and keywords in a csv. We then get a dataset of 10000 images, each with their product name, product type, color, and keywords.

To create the VQA dataset, we used the Google Gemini API to generate question-answer pairs for each image. Specifically, we used the `gemini-1.5-flash` model, to which we passed the product image, along with its metadata, and a prompt to generate questions, with instructions as follows:

```
prompt = f"""
    You are given an image, some metadata about that image, and a set of instructions.
    Follow the instructions exactly.
    Image: {img}
    Metadata:
    Name: {name}
    Product Type: {product_type}
    Color: {color}
    Keywords: {keywords}
    Instructions:
    1. Generate 5 distinct questions.
    2. The questions can be answered by looking at the image or can be inferred by thinking.
    3. Difficult questions are preferred.
    4. Do not use quotation marks anywhere.
    5. The answer should exactly be 1 word.
    6. Provide the output strictly in the format given below.
    """
```

This prompt encourages the model to generate a mix of questions; ones which can be directly answered by looking at the image such as the product color or product name, and ones which require knowledge and thinking such as product functionality or fine details. We set the `temperature` parameter to 0.5 for a balance of conservative and creative questions.

After the entire dataset processing and VQA generation process, we get a set of 50000 question-answer pairs, 5 pairs each for 10000 selected images. Each question has a single-word answer.

4 Baseline VQA models

We worked with 2 off-the-shelf VQA models provided by HuggingFace :

- **Salesforce/blip-vqa-base**: A simple BLIP-base model with a ViT backbone and a text transformer. Pretrained on VQA tasks. We chose it because of ease of implementation and smaller footprint (∼400M params), along with good support for finetuning with `LoRA` and `peft`.
- **dandelin/vilt-b32-finetuned-vqa**: A smaller, lighter and faster model (∼100M params), composed of a ViT along with a BERT encoder. Pretrained on masked image modeling and image captioning tasks, and finetuned on VQA tasks. Unlike BLIP, this model is discriminative; it generates a distribution over the vocabulary instead of decoding text tokens. We chose it due to its small resource consumption and fast inference.

We ran inference for these 2 models on a fixed random sample of 10000 questions from the VQA dataset. The predicted answers for both models were recorded, and compared with the reference answers generated by the Gemini API, using the following metrics:

- **Exact Match Accuracy**: Proportion of examples where the model's prediction exactly equals (character-for-character) the ground-truth answer.
- **Substring Match Accuracy**: Proportion of examples where the prediction is a (contiguous) substring of the ground truth, or vice versa.
- **Bert F1**: Continuous similarity score between prediction and ground truth word embeddings. Indicates semantic closeness.
- **Levenshtein Similarity**: Word similarity based on edit distance i.e. minimum number of single-character edits (insertions, deletions, substitutions) needed to turn the prediction into the reference.
- **Meteor Score**: F1 score over exact matches, stem matches and synonym matches combined with a penalty for fragmented matches.

We did not use other metrics such as **BartScore**, **BLEU 1-4**, **Rouge Score**, since they are meant to compare sequences of words, and hence are of limited value when applied over single words, even though they can technically still be computed. Since the ground truth answers as well as most of the baseline predictions are single words, we chose the metrics which are easiest to understand and interpret, as well as give sufficient amount of information regarding the performance of the models, both structurally and semantically. This means that pairs like "orange" and "vermilion", "orange" and "Orange", "orange" and "tangerine" will be identified as 'close' or 'similar' by atleast one of the above metrics.

A 'good' model, i.e. one which gives predictions close to the reference answers, will have high values for all the metrics listed above. The performance of both models before finetuning is given below:
We can see that both models perform similarly, with poor exact match accuracies but high BERT scores. The ViLT model performs poorer than the BLIP model because the ground truth reference answers given by the Gemini API have a lot of uncommon words, which may be outside of its vocabulary.

Model	Exact Match	Substring Match	BERT F1	Levenshtein	METEOR
BLIP	0.3610	0.3787	0.8383	0.5117	0.2245
ViLT	0.2487	0.2551	0.8439	0.4058	0.1592

Table 1: BLIP and ViLT performance before fine-tuning

5 Finetuning

The purpose of finetuning is to take pretrained model and adapt it to a specific domain or task through training. LoRA or Low-Rank Adaptation is an efficient method of finetuning which reduces the number of trainable paramenters while fine-tuning large models. Rather than updating every weight, LoRA injects small trainable low-rank matrices into selected layers, and only these matrices are updated during training while the other weights remain frozen. By carefully selecting what weights to update and how to update them, we can achieve significant improvements in performance compared to the baseline, on the task/domain on which the finetuned model is trained. The basic equation is

$$W' = W + \frac{\alpha}{r} BA$$

where W' is the updated weights matrix, W is the original weights matrix, r is the rank of the injected matrices, α is a scaling factor, and B and A are the two rank-r up-projection and down-projection matrices.

The basic structure of our finetuning process is as follows:

- **Select the target modules**: The layers which will receive attention updates. We selected all 3 projections (q,k,v) for each attention head.
- **Configure LoRA**: Set the rank, alpha, dropout and bias.
- **Inject adapters**: Insert the LoRA matrices alongside existing weights.
- **Train adapters**: Finetune only the adapter parameters on the task-specific dataset, keeping the rest of the model frozen. In our case, the task is visual question-answering. The loss being minimized is token-level cross-entropy.
- **Inference**: At inference time, the model combines existing weights along with the adapter parameters to generate predictions in the case of BLIP, or a distribution over the vocabulary in case of ViLT.

6 Results

The performance of both models after finetuning, on the same data they were evaluated on before finetuning, is given below.

Model	Exact Match	Substring Match	BERT F1	Levenshtein	METEOR
BLIP	0.6928	0.7065	0.9038	0.7780	0.3540
ViLT	0.2809	0.2848	0.8664	0.4076	0.1425

Table 2: BLIP and ViLT performance after fine-tuning

We see that the fine-tuned BLIP model performs much better than the baseline BLIP model, as shown by the huge increase in exact and substring matches, and a significant increase in the BERT score. The ViLT model, on the other hand, did not show such improvements after fine-tuning, and infact shows decrease in some metrics like Meteor Score. This may be due to several reasons:

- The baseline model we used is already finetuned on VQA tasks, so further finetuning may not yield good results.

- We did not tune any hyperparameters for this model, unlike we did for BLIP, due to time constraints. That may impact the fine-tuned model's performance
- The BLIP model was fine-tuned for 7 epochs, while the ViLT model was fine-tuned only for 3 epochs, again due to time constraints. Maybe more epochs would have led to better results.

7 Experiments

One of the things we experimented with a lot was the prompt given while generating the VQA dataset. It heavily influences the type of questions asked. A very basic prompt would have led to all questions being easily answerable just by looking at the metadata or image, which would have led to a very high baseline model performance itself, which leaves little room for finetuning. On the other hand, explicitly telling the prompt to only generate questions which cannot be answered by looking at the image or metadata is also no good, since it creates very big questions and the ground truth answers are themselves wrong in most of the cases. The final prompt given above offers a mix of easy and difficult questions which is suitable for visual question-answering.

Fine-tuning, along with generating inference for the fine-tuned model, is a very time-consuming task. Hence we had little time to do any good experiments. We did check a wide variety of hyperparameters for fine-tuning the BLIP model. The parameters, with their checked values and selected values are given below.

Hyperparameter	**Range**	**Selected**
Learning Rate	(1e-4, 5e-5, 1e-5)	5e-5
Weight Decay	(0, 0.01, 0.1)	0.01
LoRA Rank	(4,8,16)	16
LoRA Alpha	(16,32,64)	32
Optimizer	(Adam, AdamW)	AdamW

Table 3: Hyperparameters for fine-tuning BLIP

8 Observations

Overall Trend: The fine-tuned model generally demonstrates a significant improvement in its ability to understand and answer visual questions compared to the model before fine-tuning. This is evident in increased accuracy across various question types.

No.	**Question**	**Answer**	**Baseline Pred.**	**Finetune Pred.**
1	How many earring pendants are shown?	two	2	two
2	What is the apparent texture of the case material?	smooth	leather	smooth
3	What is the type of earring?	studs	blue	stud
4	How many colors are in the clock's design?	four	6	four
5	What is the shape of the planters?	square	square	rectangular
6	What is the main color of the sandal straps?	white	brown	beige
7	What is the primary color of the bag?	beige	white	beige
8	What is the texture of the mat?	linen	soft	linen
9	What type of shoe is shown?	sneaker	tennis	running
10	What is the pendant's design?	state	cat	california
11	How many binders are shown?	four	5	four
12	How many chairs are shown in the image?	two	3	two

From the above table it is clear that fine-tuning leads to an improvement in the following tasks:

- **Exact matches**: Q1
- **Comprehension and question understanding**: Q2,Q3,Q9,Q10
- **Color identification**: Q5, Q6
- **Fabric and material decriptions**: Q8
- **Refined object classification**: Q9, Q10

Sometimes, as seen in Q5, we see that the fine-tuned prediction fails to identify the shape of the planters precisely. This is not a major problem, as squares and rectangles are very similar shapes.

9 Challenges

The biggest challenge was undoubtedly the amount of resources needed (time, memory, data and compute) to create the VQA dataset and fine-tune both the models. Doing fine-tuning locally is almost impossible, so we worked on Kaggle Notebooks with Nvidia T4 $\times$ 2 GPUs. Even with that, it takes around 5 hours to fine-tune the baseline BLIP model completely, and 3 hours for the ViLT model. Running inference on them for a set of 10000 questions takes an additional 20-30 minutes. Kaggle only offers 30 hours of GPU time a week, so we had to work carefully.

10 Contributions

- IMT2022017 Prateek Rath: Wrote code for curating the data and generating the VQA dataset.
- IMT2022076 Mohit Naik: Wrote code for fine-tuning the BLIP model along with code for inference.
- IMT2022519 Vedant Mangrulkar: Wrote code for fine-tuning the ViLT model along with code for inference.

The other tasks of setting up the environment, managing requirements and writing the report were done by all 3 members equally.