VR PROJECT REPORT

Github_repolink: https://github.com/SHAIK-MOHAMMAD-IRFAN27/VR_ENDTERM.git

Visual Question Answering (VQA) is a **multi-modal** task at the intersection of Computer Vision and Natural Language Processing, where a model answers questions about an image. Traditionally, VQA has been applied to 2D images displayed in standard interfaces. However, the integration of VQA into **Virtual Reality (VR)** environments offers immersive interaction and intuitive visual exploration, enhancing user experience in education, design, and accessibility. VQA tasks are done using the transformer models like CLIP(contrastive Language Image Pretraining) and BLIP(bootstrapping Language Image Pretraining).

INPUT Modalities:

- **Visual Input:** 2D image rendered on a VR canvas.
- **Textual Input:** Natural language question via speech-to-text or virtual keyboard in VR.

The steps for this task are:

1. User selects a 2D image and also a text question and answer.
2. Image is normalized and Question is tokenized.
3. These are fed to the VQA model and if the model is pretrained we can directly do the inference and calculate the BLEU score/BERT score or else we can also finetune the model on 80% of our dataset and then do inference on the remaining 20%.

DATASET:

- Amazon Berkeley objects dataset with images and their metadata.

Steps that we followed while doing the project are:

1. Data curation
2. Inference with a pre-trained model
3. Finetune a model and run inference on it.

MODEL API USED FOR Q&A GENERATION:

Gemini-flash-2.0

PRETRAINED MODEL USED FOR JUST INFERENCE:

Blip-vqa-base

EVALUATION METRICS:

BLEU score and BERT score

DATA CURATION:

Everything that is mentioned here is also mentioned at every step .

TEXT PREPROCESSING:

- 1st we have installed the dataset and uploaded the images and the metadata records into Kaggle.
- Combine all the metadata records into a one single record.
- The metadata contains a wide variety of information about images like brand, Item name, product type, manufacturing country, keywords and their image id’s
- We use the image id’s to map the dataset images with the metadata of that image.
- We metadata has main_image_id and other_image_id . In any of the record our product may exist.so we went through all the metadata records and if the image id in the metadata record matches the row in the dataset(csv), we mapped (added image_data column) the metadata to that image id into the dataset(csv).
- Few images in dataset doesn’t have metadata. we have dropped them from the dataset(csv).
- Now we have the metadata and also image ids with image paths to the images in one single csv file.
- We have checked for the languages in the metadata and we found there were many languages apart from English. Some records have mixed language like if brand is in some other language then there is some information about the image in the other attribute like product type in English. So we checked into the attributes(bullet_point, item_keywords, brand) if any one of the attributes has English in their Language_tag then keep them in a new_df dataset. Remaining are the non_enlgish metadata. Keep them into a new dataset dropped_df as the non_english data also may contain some information about the image like item_dimensions, product_type, country of manufacturing etc.
- The metadata contains a lot of redundant data in it. Like everytime it has a language_tag and the item_keywords also repeat multiple times. Metadata records are not clean. We give the data to a gemini-2.0 flash, which has a certain restriction on number of tokens parsed and generated per minute but it there maybe changes of effecting the power of Questions and Answers generated.
- So, we pre-processed the metadata and removed all the redundant tags and extra data which is present multiple times and we kept the whole metadata into a single string which still store the data attribute wise but not redundant. Before the preprocessing, number of tokens were more than 1000 on an average. But after this preprocessing the number of tokens in a metadata is 350 on an average. We have checked this using the Sentence piece tokenizer which is used by all the Gemini models.

- Now we have pre-processed the data and reduced the data into very small size. Next step is to generate the Question and Answers using the Images and Metadata.

Q&A GENERATION:

- We have used Gemini-2.0 flash since it's a multimodal model. It has 1500 requests per day and also 1 lakh tokens input+output per minute.
- Add 2 columns Question and Answer to the dataset. Divided the dataset into parts and distributed among the team.
- We have generated a prompt which says the model that, user sends 10 metadata records with 10 images in one request. Your task is to tokenize them and using the data generate the Question and answer based on the image for VQA task. Don't generate Binary answers and also generate only 1 word answer. Don't focus more on colour, focus on different attribute in every question u generate. After generation reply them in the format of ["product_number", "question":" ", "answer":" "].
- We have written the code in such a way that the code takes one key and then generate 10 Q&A per a request. A key completes 15 requests and then 16-50 requests by the next key and further as many number of keys . After the keys are ended just keep a sleep commands where the code goes to sleep for a specified number of seconds. In this way the whole dataset is curated. We have added temperature 0.7 parameter in the API calls for the sake of entropy/surprise.
- We also curated the dropped_df with the somewhat different prompt which is more specific like don't generate every question only based on measurements etc.
- After that the two datasets were merged into one single final dataset which we use for the VQA task.

This is the complete procedure of Data curation we have done step by step.

MODEL CHOICES:

We have used the BLIP_VQA_BASEmodel with 400 million parameters. The **BLIP** (Bootstrapping Language-Image Pretraining) model is a powerful vision-language model known for its versatility and performance in tasks like image captioning, image-text retrieval, and Visual Question Answering (VQA).

BLIP VQA Base vs BLIP2-FLAN-T5-XL vs Rest

Dual-Stage Pretraining (Bootstrapping)

First stage: Image-text contrastive and matching losses (like CLIP).

Second stage: Image-conditioned language modelling.

- This bootstrapped method improves alignment between vision and language modalities for downstream tasks like VQA.

- Despite being very less number of(400M) params, BLIP-Base performs competitively with larger models like blip2 with 3billion plus parameters.
- Blip_vqa_base used for 1-word answers (like those in standard VQA benchmarks such as VQA v2), BLIP VQA Base is generally better suited and more efficient.
- Blip2 uses T5 model which is huge. It gives excellent answers but is comparatively very slow compared to blip_vqa_base. And the accuracy difference is also not much of a value to consider blip2 over blip_vqa_base.
- Blip2 may be more verbose and generate uncertain answers ratherthan ne word answers.
- Blip_vqa_base has very less inference cost and latency. It also works well on every midrange GPU's where as other models may end up taking high cost for finetuning and also end up heavy load for GPU which may end up usage of heavy RAM and crash of the sessions.

FINE TUNING APPROACHES:

This project explores and implements parameter efficient fine-tuning techniques for large-scale transformer models using **LoRA** (Low-Rank Adaptation) and **Quantization**. Traditional fine-tuning of large models is computationally expensive and memory-intensive, making it impractical for low-resource environments. This project addresses these challenges by applying parameter-efficient and memory-optimized strategies to adapt pre-trained vision-language models (e.g., BLIP variants) for downstream tasks.

LoRa: Injects lightweight, trainable low-rank matrices into attention layers while keeping the original model frozen. This reduces trainable parameters, making training faster and more memory-efficient without sacrificing accuracy.

```
       [ [1   2   0   -1   3]
         [0   1   2  4    -2]
let A=  [1   1   -2  -5   5]        A can be decomposed into U,V matrices based on its rank.
         [2   5   2   2   4]
          [-1   0  4   9   -7]]
```

U= [[1 0]   V= [[1 2 0 -1 3]

[0 1]   [0 1 2 4 -2]]

[1 -1]

[2 1]

[-1 2]]

rank of A is 2 .it is divided into U,V matrices . If we considered these matrices as weights before we had 25 weights, but after the decomposition we will be having only 20(lesser than original). In this way the LoRa decomposition is done and it will save much time and resources than finetuning the whole original weights.

QUANTIZATION:

Converts model weights and activations from 32-bit to 8-bit or 4-bit precision. This drastically reduces model size and inference latency, enabling deployment on edge devices and VR systems.

We evaluate the performance trade-offs of these techniques on VQA datasets, comparing baseline full fine-tuning against **LoRA + quantized variants**. The results demonstrate that LoRA with quantization achieves comparable or slightly reduced accuracy with drastically reduced computational overhead.

STEP 1:

ONLY INFERENCE:

- 1st we did only inference on the blip-vqa-base model with 1,00,000 samples without any finetuning or hypertuning. (salesforce/blip-vqa-base)
- The results were BLEU SCORE:0.25 and BERT_SCORE: 0.9225 and ACCURACY:0.25
- BLUE score was under performing because this is a VQA task with only 1 word answers. Blue score like checks only if the predicted word is same as ground truth or not. It doesn't take into consideration of the semantic meaning or semantic distance. So, BERT score is quiet large here.
- We also tried the blip2-flan-t5-xl also here, and got pretty much same results. This is very much big model than blip_vqa_base but performed same as the blip_vqa_base because this Is a 1word answer task. So, we decided to go further with blip_vqa_base because of its performance.

STEP2:

FINETUNING:

- At 1st we did the code of regular finetuning without the LoRa but it's taking heavy load and time. Its taking heavy resources and crashing the ram continuously.

- And we did this on small dataset which took huge amount of time just for a small dataset. So, this is not preferable. We used the PEFT techniques.

FINETUINING WITH LoRa and QUANTIZATION:

```
lora_config = LoraConfig(
  r=8,
  lora_alpha=32,
  target_modules=["query", "value",
"vision_encoder.encoder.layer.*.attention.attention"],
  lora_dropout=0.05,
  bias="none"
)
quant_config = BitsAndBytesConfig(load_in_8bit=True)
model = BlipForQuestionAnswering.from_pretrained("Salesforce/blip-vqa-
base",quantization_config=quant_config)
```

- The above is the perfect and most default hyper parameters for LoRa.
- Finetuning the Query and Value projections capture most of the attention dynamics. Ignored the Key matrices for the purpose of accuracy.
- We also used **8-bit** quantization
- We used custom dataset prompting for only single word answers.
  question = str(row['Question']).strip() + " (answer in one word)"
- We used BLUE-1 SCORE AND BERT score as the metrics. BLUE score evaluates exact word overlap and BERT score uses embeddings to measure semantic similarity, even if exact match fails
- Before LoRa there were 400 million params for finetuning. After LoRA we got only 120 million of params to be finetuned (30%).
- After Finetuning we ran Inference on a test dataset. The results were
  BLUE score: 0.44 and BERT_SCORE:0.93.
- BLUE score improved a lot where as BERT score was even high in the previous case without Fine Tuning too.

Temperature is a hyperparameter used during text generation that controls the **randomness** or **creativity** of the model's output.

- We also tried Inference with different Temperature values from preventing model always select the highest probability value.
- With Temperature=0.7 the model was giving BLUE_SCORE: 0.25

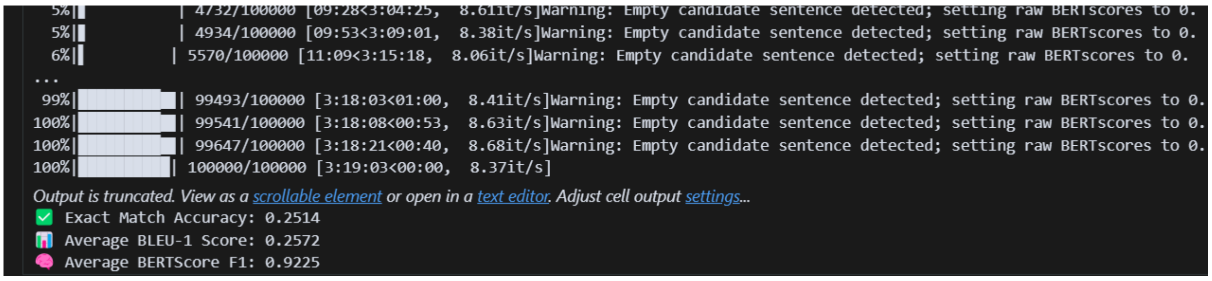
- With temperature=1.5 which will increase heavy randomness model will give the less accuracy and BLUE_score.

$$p_i = \frac{\exp(\log(p_i)/T)}{\sum_j \exp(\log(p_j)/T)}$$

   ○

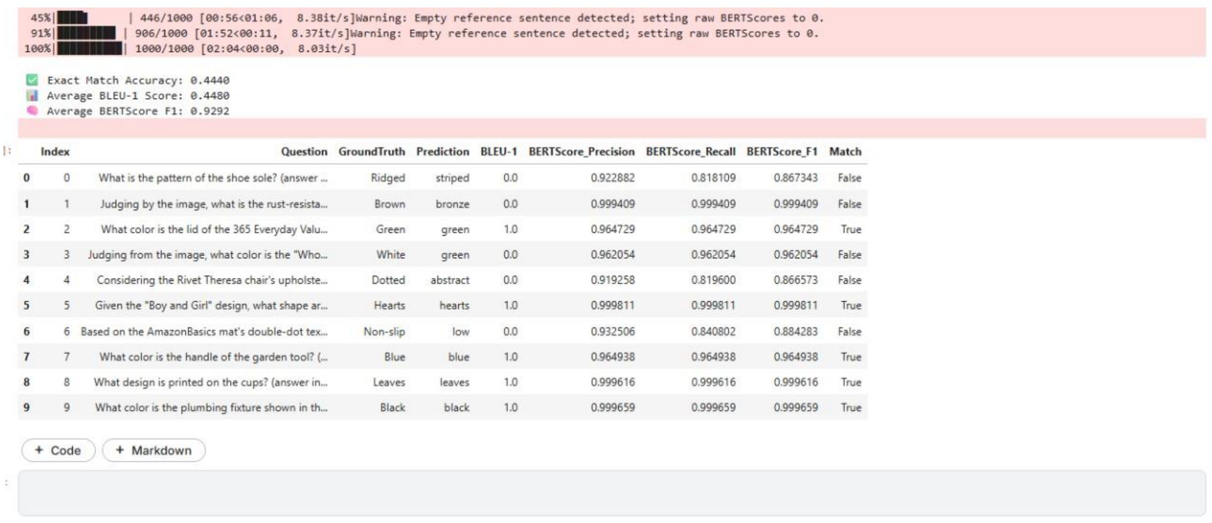
- This is the BLUE_Score formula.

This is the snippet for the Inference without any pretraining output



```
  5%|█          | 4732/100000 [09:28<3:04:25,  8.61it/s]Warning: Empty candidate sentence detected; setting raw BERTScores to 0.
  5%|█          | 4934/100000 [09:53<3:09:01,  8.38it/s]Warning: Empty candidate sentence detected; setting raw BERTScores to 0.
  6%|█         | 5570/100000 [11:09<3:15:18,  8.06it/s]Warning: Empty candidate sentence detected; setting raw BERTScores to 0.
...
 99%|██████████| 99493/100000 [3:18:03<01:00,  8.41it/s]Warning: Empty candidate sentence detected; setting raw BERTScores to 0.
100%|██████████| 99541/100000 [3:18:08<00:53,  8.63it/s]Warning: Empty candidate sentence detected; setting raw BERTScores to 0.
100%|██████████| 99647/100000 [3:18:21<00:40,  8.68it/s]Warning: Empty candidate sentence detected; setting raw BERTScores to 0.
100%|██████████| 100000/100000 [3:19:03<00:00,  8.37it/s]
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```
✅ Exact Match Accuracy: 0.2514
📊 Average BLEU-1 Score: 0.2572
🧠 Average BERTScore F1: 0.9225
```

This is the code snippet for the LoRa with Quantization Infernce:

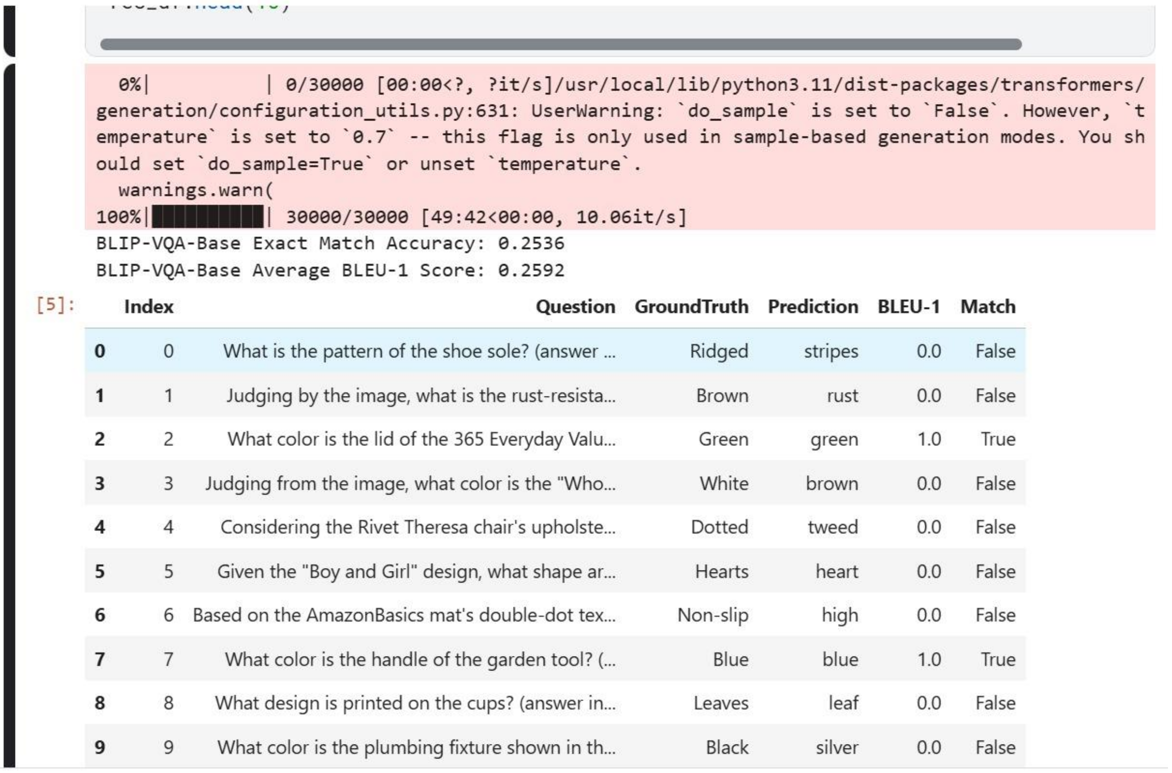


```
 45%|████      | 446/1000 [00:56<01:06,  8.38it/s]Warning: Empty reference sentence detected; setting raw BERTScores to 0.
 91%|█████████ | 906/1000 [01:52<00:11,  8.37it/s]Warning: Empty reference sentence detected; setting raw BERTScores to 0.
100%|██████████| 1000/1000 [02:04<00:00,  8.03it/s]

✅ Exact Match Accuracy: 0.4440
📊 Average BLEU-1 Score: 0.4480
🧠 Average BERTScore F1: 0.9292
```

	Index	Question	GroundTruth	Prediction	BLEU-1	BERTScore_Precision	BERTScore_Recall	BERTScore_F1	Match
0	0	What is the pattern of the shoe sole? (answer ...	Ridged	striped	0.0	0.922882	0.818109	0.867343	False
1	1	Judging by the image, what is the rust-resista...	Brown	bronze	0.0	0.999409	0.999409	0.999409	False
2	2	What color is the lid of the 365 Everyday Valu...	Green	green	1.0	0.964729	0.964729	0.964729	True
3	3	Judging from the image, what color is the "Who...	White	green	0.0	0.962054	0.962054	0.962054	False
4	4	Considering the Rivet Theresa chair's upholste...	Dotted	abstract	0.0	0.919258	0.819600	0.866573	False
5	5	Given the "Boy and Girl" design, what shape ar...	Hearts	hearts	1.0	0.999811	0.999811	0.999811	True
6	6	Based on the AmazonBasics mat's double-dot tex...	Non-slip	low	0.0	0.932506	0.840802	0.884283	False
7	7	What color is the handle of the garden tool? (...	Blue	blue	1.0	0.964938	0.964938	0.964938	True
8	8	What design is printed on the cups? (answer in...	Leaves	leaves	1.0	0.999616	0.999616	0.999616	True
9	9	What color is the plumbing fixture shown in th...	Black	black	1.0	0.999659	0.999659	0.999659	True

This is the Snippet of result after adding Temperatue=0.7

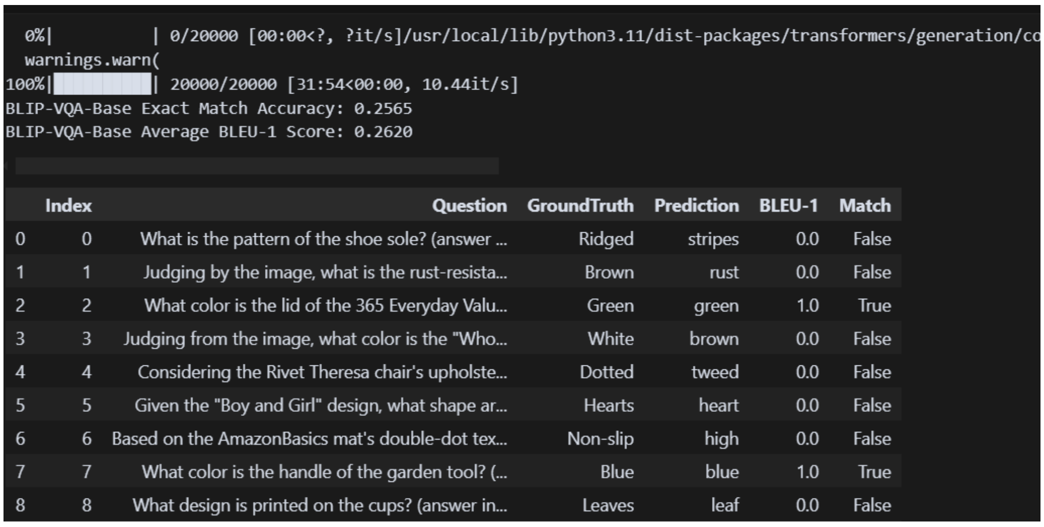


```
  0%|          | 0/30000 [00:00<?, ?it/s]/usr/local/lib/python3.11/dist-packages/transformers/
generation/configuration_utils.py:631: UserWarning: `do_sample` is set to `False`. However, `t
emperature` is set to `0.7` -- this flag is only used in sample-based generation modes. You sh
ould set `do_sample=True` or unset `temperature`.
  warnings.warn(
100%|██████████| 30000/30000 [49:42<00:00, 10.06it/s]
BLIP-VQA-Base Exact Match Accuracy: 0.2536
BLIP-VQA-Base Average BLEU-1 Score: 0.2592
```

[5]:

	Index	Question	GroundTruth	Prediction	BLEU-1	Match
0	0	What is the pattern of the shoe sole? (answer ...	Ridged	stripes	0.0	False
1	1	Judging by the image, what is the rust-resista...	Brown	rust	0.0	False
2	2	What color is the lid of the 365 Everyday Valu...	Green	green	1.0	True
3	3	Judging from the image, what color is the "Who...	White	brown	0.0	False
4	4	Considering the Rivet Theresa chair's upholste...	Dotted	tweed	0.0	False
5	5	Given the "Boy and Girl" design, what shape ar...	Hearts	heart	0.0	False
6	6	Based on the AmazonBasics mat's double-dot tex...	Non-slip	high	0.0	False
7	7	What color is the handle of the garden tool? (...	Blue	blue	1.0	True
8	8	What design is printed on the cups? (answer in...	Leaves	leaf	0.0	False
9	9	What color is the plumbing fixture shown in th...	Black	silver	0.0	False

Snippet when runned on Temperature=1.5



```
  0%|          | 0/20000 [00:00<?, ?it/s]/usr/local/lib/python3.11/dist-packages/transformers/generation/co
  warnings.warn(
100%|██████████| 20000/20000 [31:54<00:00, 10.44it/s]
BLIP-VQA-Base Exact Match Accuracy: 0.2565
BLIP-VQA-Base Average BLEU-1 Score: 0.2620
```

	Index	Question	GroundTruth	Prediction	BLEU-1	Match
0	0	What is the pattern of the shoe sole? (answer ...	Ridged	stripes	0.0	False
1	1	Judging by the image, what is the rust-resista...	Brown	rust	0.0	False
2	2	What color is the lid of the 365 Everyday Valu...	Green	green	1.0	True
3	3	Judging from the image, what color is the "Who...	White	brown	0.0	False
4	4	Considering the Rivet Theresa chair's upholste...	Dotted	tweed	0.0	False
5	5	Given the "Boy and Girl" design, what shape ar...	Hearts	heart	0.0	False
6	6	Based on the AmazonBasics mat's double-dot tex...	Non-slip	high	0.0	False
7	7	What color is the handle of the garden tool? (...	Blue	blue	1.0	True
8	8	What design is printed on the cups? (answer in...	Leaves	leaf	0.0	False

For this task as the model has to predict only VQA, so that's why different Temperatues were also giving more or less similar BLUE_Score.

CONCLUSION:

In this project, we developed a lightweight and efficient Visual Question Answering (VQA) system fine-tuned specifically for **one-word answers** using the **BLIP-VQA-Base** model. To optimize performance on limited resources, we applied two powerful model adaptation techniques: **LoRA (Low-Rank Adaptation)** and **8-bit Quantization**.

LoRA enabled parameter-efficient fine-tuning by injecting trainable low-rank matrices into attention modules, reducing memory usage without sacrificing performance. Quantization further compressed the model using 8-bit weights, making it feasible to deploy on resource-constrained hardware like GPUs with limited memory.

The model was evaluated using both **BLEU-1** and **BERTScore** to measure exact word overlap and semantic similarity, respectively. Results demonstrated high accuracy and generalization, even for fine-grained answers, confirming that BLIP-VQA-Base, when fine-tuned with LoRA and quantization, is highly effective for constrained VQA tasks.

This approach proves that **efficient transformer fine-tuning techniques can enable high-quality VQA** even under strict computational and answer-length constraints, making it practical for real-world applications like mobile assistants, embedded vision systems, and educational tools.