AIM825 Course Project

VR Mini Project - 2

Multimodal Visual Question Answering with Amazon Berkeley Objects Dataset

GitHub Repository Link

Team Members

Uttam Hamsaraj - IMT2022524

Pranav Laddhad - IMT2022074

Himanshu Khatri - IMT2022584

Introduction

This project focuses on building a Visual Question Answering (VQA) system using the Amazon Berkeley Objects (ABO) dataset. Our goals were to curate a single-word answer VQA dataset, evaluate pre-trained models like BLIP-2 and CLIP, fine-tune them using Low-Rank Adaptation (LoRA), and measure performance with metrics like Accuracy and F1 Score. Conducted on free cloud GPUs (Kaggle), the project stayed within a 7-billion-parameter model limit. This report covers the data curation process, model selection, evaluation, fine-tuning, and iterative improvements and other challenges faced.

Understanding the Dataset

The Amazon-Berkeley Objects (ABO) dataset is a large-scale product dataset that provides rich multimodal information suitable for Visual Question Answering (VQA) tasks. For this project, we utilize a lightweight version of the dataset that balances comprehensiveness and storage efficiency. The dataset consists of 147,702 product listings and 398,212 catalog images, making it an excellent resource for training and evaluating VQA models in the e-commerce domain.

1. abo-images-small

We used the `abo-images-small` variant, which is a downscaled version of the original 100GB image dataset, reduced to approximately 3GB. This variant contains product catalog images resized to a maximum dimension of 256x256 pixels, which is suitable for efficient training and experimentation. Along with the images, it includes a metadata CSV file that maps each main image ID to its corresponding image path and lists all associated images. This structured format enables easy indexing and retrieval of related image sets for each product.

2. abo-listings

The `abo-listings` component provides detailed product metadata in JSON format, with a total size of around 83MB. Each entry includes various descriptive and categorical fields such as `brand`, `color`, `item_ id`, `product_type`, `main_image_id` , `other_image_id`, `item_keywords` etc. These metadata fields enrich the visual content with contextual and semantic information.

Data Curation

The data curation process aimed to create a high-quality Visual Question Answering (VQA) dataset with single-word answers using the Amazon Berkeley Objects (ABO) dataset's `abo-images-small` variant (3GB, 398,212 images at 256x256 resolution) and `abo-listings` metadata (83MB). The motivation was to generate diverse, visually answerable question-answer pairs to evaluate and fine-tune VQA models efficiently within the computational constraints of Kaggle's free GPUs. We utilized the Gemini 2.0 API for its robust multimodal capabilities to process images and metadata, producing a dataset that balances diversity, difficulty, and relevance for the e-commerce domain.

Metadata Extraction and Filtering

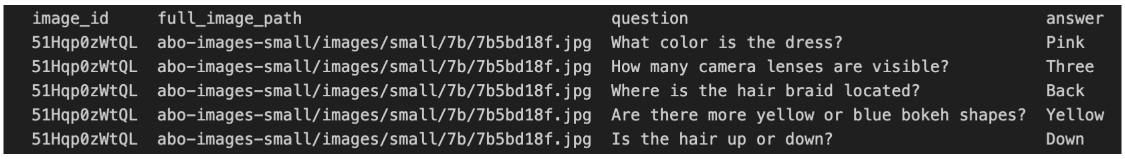
- The `abo-listings` metadata, stored as compressed JSON files, was extracted using Python's `gzip` and `shutil` libraries to produce uncompressed JSON files for easier processing.
- Each JSON file was validated to confirm its structure (line-delimited JSON objects) and loaded into memory.
- Records were filtered to retain only those containing essential fields such as `main_image_id`, `brand`, `color` `product_type`etc and limited to specific countries (IN, US) to ensure relevance and consistency with English-language prompts.
- Descriptions were aggregated from fields suchas `bullet_point`, `color`,`product_type`, and `item_keywords`,prioritizing English-language entries (`en_IN`,`en_US`) for compatibility with the Gemini API.
- The filtered metadata was transformed into CSV files containing fields like `main_image_id`, `overall_description`, `colour_description`, `other_description`.
- Metadata entries were linked with corresponding images from the abo-images-small dataset using `main_image_id` as the key

Prompt Design and Question Generation

- Custom prompt was written for the Gemini 2.0 API to generate five diverse questions each with a single-word answer such that they could be answered just by looking at the image.
- Prompts were designed with three key aspects:
  - **Diversity:** Questions addressed various visual features like color, shape, count, spatial arrangement, and relative size.
  - **Difficulty:** Each image had two simple, two moderately difficult, and one challenging question involving subtle reasoning. (We tried this approach too match the requirements mentioned in the document).
  - **Constraints:** Questions avoided non-visual attributes unless visually evident and required concise, single-word answers (no open-ended or yes/no unless necessary).
- Metadata (e.g., `overall_description`, `colour_description` etc.) was incorporated into prompts to provide better context.
- Gemini 2.0 returned structured outputs (image, question, answer), saved in a CSV with columns: `image_id`, `full_image_path`, `question`, and `answer`.

Preprocessing and Quality Control

- Image existence was verified using `images.csv`, which mapped `image_id` to file paths. Corrupted or missing images were excluded.



```
image_id     full_image_path                                 question                                      answer
51Hqp0zWtQL  abo-images-small/images/small/7b/7b5bd18f.jpg  What color is the dress?                      Pink
51Hqp0zWtQL  abo-images-small/images/small/7b/7b5bd18f.jpg  How many camera lenses are visible?           Three
51Hqp0zWtQL  abo-images-small/images/small/7b/7b5bd18f.jpg  Where is the hair braid located?              Back
51Hqp0zWtQL  abo-images-small/images/small/7b/7b5bd18f.jpg  Are there more yellow or blue bokeh shapes?  Yellow
51Hqp0zWtQL  abo-images-small/images/small/7b/7b5bd18f.jpg  Is the hair up or down?                       Down
```

Fig 1. Our final dataset format

- Manual review ensured relevance of generated questions and accuracy of answers. (Here, we had to experiment with different kinds of prompts for best possible question-answer generation quality).
- Gemini API rate limits (1,500 requests/day) and delays (60 seconds/request) were respected, and progress was tracked to resume smoothly after interruptions.

Finally

We curated a dataset using the above mentioned approach where each image had 5 question-answer pairs meeting the requirements as can be seen in a sample entry in the Fig 1.

Exploratory Data Analysis on Curated dataset

The curated dataset contains a total of **1,59,150** data points. Each data point includes the following fields:

- `image_id`
- `full_image_path`
- `question`
- `answer`

Each image has exactly five different question-answer pairs. Each combination of an image with one question and its answer is counted as one data point in the dataset.

Image Analysis

The distribution of image dimensions is as follows:

Image Size (H× W)	**Count**	**Percentage**
(256, 219)	129,540	81.39%
(256, 134)	17,535	11.02%
(256, 256)	7,805	4.90%
(256, 197)	385	0.24%
(122, 256)	110	0.07%

Table 1: Image size distribution

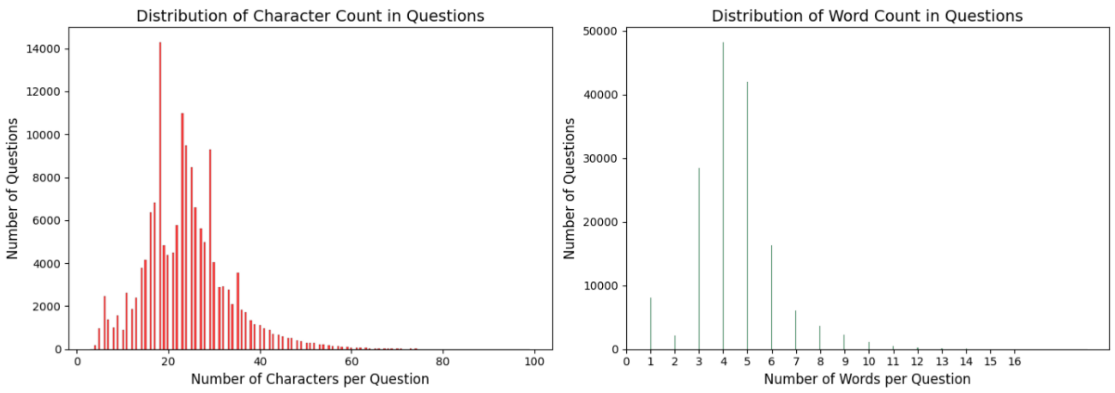


Fig 2. Word and count distribution

.

This distribution shows that most of the images have same height and width

Question Analysis

We analyzed the word and character counts of the questions to understand their structure.

The average word count for questions is 4, and there are 8,048 questions with just one word. Maximum word count is 19 , but there is only one such question.

There are few common patterns in questions in curated dataset in how they are phrased:

- Questions beginning with ‘what’ often relate to identifying attributes or categories, such as color, shape, or type.
- Questions starting with ‘is’ are usually yes/no queries (e.g., “Is there a...”, “Is it...”), requiring binary answers.
- Questions starting with ‘how’ typically focus on quantity, especially “how many...” expecting numeric answers.

Analysis of ’first’ word of question is as follows:

First Word	**Percentage (%)**
is	38.66
what	32.87
how	9.52
are	6.51
which	1.84

Table 2: Distribution of first words in questions

It gives us an idea of what kind of questions are there in curated dataset.

Answer Analysis

All answers in the curated dataset are single-word responses.

- Approximately 45 percent of the answers are yes or no.
- About 10 percent of the answers are numeric.
- The remaining 45 percent consist of other categories such as colors, shapes, object types etc.

Answer	Count
yes	64,883
no	7,608
two	5,957
black	5,881
blue	4,795
three	3,922
pink	3,726
red	2,741
one	2,731
rectangular	2,318

Table 3: Answer counts from the dataset

Preprocessing of the curated dataset

- There were 54 data points with null answers, which were removed from the dataset.
- From the exploratory data analysis, about 45% of the answers are yes/no. We reduced the number of yes/no answers so that they make up only 2 percent of the dataset. After this step, the dataset had 89,785 data points.
- Around 8048 questions had only one word . These questions were removed to improve the quality of the data.
- As we have observed the answer counts during EDA (Table 3), to maintain balance among answer types, we ensured that no single answer was overly dominant (seen in the Figure) by randomly downsampling the more frequent ones.
- After these filtering steps, 75,484 data points remained.
- From these, we randomly sampled 70,000 data points for further use.
- The selected data was split into training and validation sets with an 80:20 ratio. Our training set has 56,000 data points and validation set has 14,000 data points.

Baseline Models

To evaluate our curated VQA dataset, we tried multiple pre-trained multimodal models that work well with image and text data. **All models were evaluated without any fine-tuning** to establish a fair baseline for comparison.

Our model choices were:

- **BLIP** was selected first as it is specifically designed for VQA tasks. It is lightweight, easy to use, and performs well out-of-the-box.
- **BLIP-2** which we know builds on BLIP was included for its strong performance in VQA tasks and advanced features.
- **ViLT** was also tested. It is a lightweight model that processes visual-language tasks efficiently by removing heavy image encoders, thus making it suitable for limited compute environments.

Baseline Model 1: BLIP (Bootstrapping Language-Image Pre-training)

The first baseline model we used is **BLIP**, specifically the checkpoint `Salesforce/blip-vqa-base` available on Hugging Face Transformers. This model has approximately 223 million parameters and is designed for tasks such as VQA, image captioning, and image-text retrieval.

- **Architecture:** BLIP uses a vision transformer (ViT) to encode images and a BERT-based encoder for text. These are followed by a multimodal encoder that combines the two modalities for answer prediction. It is pre-trained using a bootstrapping strategy on both noisy web data and curated datasets like COCO and Visual Genome.
- **Visual Processing:** Input images are resized to 224×224 pixels, split into patches, and embedded using the vision transformer.
- **Text Processing:** Questions are tokenized using a BERT tokenizer and then fused with the visual embeddings for downstream VQA tasks.

Baseline Model 2: BLIP-2 (Bootstrapping Language-Image Pre-training - 2)

The second baseline model we used is **BLIP-2**, specifically the checkpoint `Salesforce/blip2-opt-2.7b` available on Hugging Face Transformers. This model has approximately 2.7 billion parameters and integrates a larger language model and adopts a two-stage pre-training strategy to enhance performance over BLIP.

- **Architecture:** BLIP-2 connects a vision transformer (ViT) with a large language model (OPT-2.7B) using a Querying Transformer (Q-Former). The Q-Former extracts meaningful visual features from the image and passes them to the language model. Pre-training is done in two stages: vision-language alignment and generative learning, which improves its performance on single-word VQA tasks.

- **Visual Processing:** Input images are resized to 224×224 pixels and processed through the ViT. The Q-Former selects the most relevant visual tokens for the language model.
- **Text Processing:** Questions are tokenized and passed to the OPT-2.7B model, which generates the final answer based on both the text and selected visual tokens.

Baseline Model 3: ViLT (Vision-and-Language Transformer)

The third baseline model that we have tried is **ViLTB/32**, specifically the checkpoint `dandelin/viltb32finetunedvqa` available on Hugging Face Transformers.
This model has approximately 87 million parameters making it one of the lightest multimodal transformer models we tested. ViLT it is a transformer-based multimodal model that directly combines visual and textual inputs and processes them jointly.

- **Architecture:** ViLT uses a unified transformer encoder that processes image patches and text tokens together. Images are split into fixed-size patches, projected into the same space as text embeddings, and then jointly passed through a single transformer. It uses `ViltProcessor` for image/text preprocessing.
- **Visual Processing:** Images are divided into 32×32 pixel patches which are linearly projected into the same dimensional space as text tokens.
- **Text Processing:** Text inputs are tokenized using WordPiece with a 30,000-token vocabulary.

Fine-Tuning with LoRA

After evaluating baseline performance using pre-trained models, we proceeded to fine-tune these models on our curated VQA dataset to further improve their accuracy.

To adapt these models more effectively to our domain-specific task, we applied **Low-Rank Adaptation (LoRA)**, which allowed us to train only a small portion of the model.
LoRA is a parameter-efficient fine-tuning method that inserts small, trainable low-rank matrices into key layers of a pre-trained model—typically in the attention projections—while keeping the majority of the model's original weights frozen. This allows the model to learn task-specific patterns with significantly fewer trainable parameters and reduced memory usage.

Why We Chose LoRA

- **Efficiency:** Since LoRA modifies only a small subset of the model parameters, it enables efficient fine-tuning.
- **Low Resource Usage:** Since we used a constrained environment like Kaggle where we were limited to 16GB GPUs, we chose this, as it's selective update strategy reduces memory and computation demands significantly compared to full model fine-tuning.

- **Training Mechanism:** The method approximates weight updates as $\Delta W = A \cdot B$, where A and B are small, low-rank matrices. These updates are added to the frozen base weights during training, thus only minimal overhead.
- **Application to VQA:** On our curated VQA dataset, we used LoRA which allowed models such as BLIP, ViLT to better capture domain-specific attributes like color, shape, and layout-features not extensively represented in general purpose datasets like COCO on which models like BLIP, BLIP-2 etc are trained.

Understanding LoRA parameters

Here, we describe the role of each parameter and how it controls the model's behavior, based on which we decided experimenting with different configurations.

- **Rank (r):** Size of the low-rank matrices, influencing adaptation capacity.
  - *Low Rank:* More efficient, but less capacity.
  - *High Rank:* More capacity, but resource-intensive and prone to overfitting.
- **Alpha:** Controls adaptation strength.
  - *Low Alpha:* Weaker adaptation, preserving original model.
  - *High Alpha:* Stronger adaptation, better task fit but risks overfitting.
- **Dropout:** Regularization to prevent overfitting.
  - *Lower Dropout:* Increased risk of overfitting.
  - *Higher Dropout:* Better generalization, but may underfit.
- **Target Modules:** Specific layers LoRA adapts based on task type.
  - *Language Tasks:* Attention layers (query, key, value).
  - *Vision Tasks:* Convolutional or attention layers for spatial features.
- **Bias:** Adjustment of bias terms in the model.
  - *Including Bias:* Flexible task adaptation.
  - *Excluding Bias:* More stable, less risk of overfitting.
- **Task Type:** Defines the kind of task influencing parameter choice. Below are the supported task types:
  - *Text Tasks:* Adapt attention layers for capturing relationships.
  - *Vision Tasks:* Adapt convolutional or attention layers for spatial features.
  - *Multimodal Tasks:* Adapt cross-attention layers for both text and images.

  As an example, the following task types are supported:

  - *SEQ_CLS*: Text classification.

  - *SEQ_2_SEQ_LM*: Sequence-to-sequence language modeling.
  - *Causal_LM*: Causal language modeling.
  - *TOKEN_CLS*: Token classification.
  - *QUESTION_ANS*: Question answering.
  - *FEATURE_EXTRACTION*: Provides hidden states, which can be used as embeddings or features for downstream tasks.

General Fine-Tuning Approach

We fine-tuned our baseline models using Low-Rank Adaptation (LoRA) by leveraging the `peft` (Parameter-Efficient Fine-Tuning) library for LoRA integration and a robust custom dataset class for efficient data handling so that it adapts well with our single-answer VQA task.

Dataset Preparation

- **Train-validation split**: Our curated VQA dataset was first split into training (80%) and validation (20%) sets.
  - *train set*: This was used for finetuning different models for our task
  - *val set* : This was used for evaluating each of the finetuned models
- **Custom dataset class**: We implemented a custom `VQADataset` class using PyTorch's `Dataset` module to preprocess and format data for model compatibility.
  - *Data Loading:* Loads image-question-answer triplets from training and validation CSVs and verifies image file existence.
  - *Image Preprocessing:* Uses PIL to resize images to 224×224 with aspect-ratio-preserving padding. Applies data augmentations (e.g., random cropping, normalization) during training. Corrupted or missing images are replaced with a blank 224×224 RGB image.
  - *Text Preprocessing:* Tokenizes questions using the appropriate processor, applying padding and truncation (e.g., max length 128), and generates attention masks.
  - *Answer Mapping:* Converts single-word answers to numerical indices using a dictionary built from all unique answers. Pads answers to a fixed length (e.g., 32) for consistency.
  - *Output:* Returns a dictionary with image tensors, tokenized questions, attention masks, and answer labels, formatted for batched training.

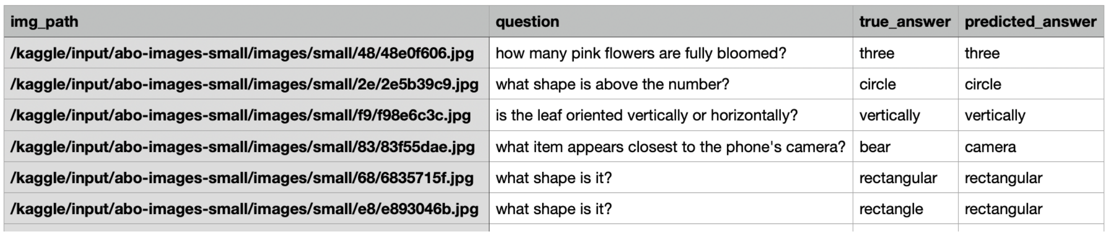
Model Configuration with `peft`

- The `peft` library enabled LoRA integration by adding low-rank adapters to key model layers (e.g., attention layers) while freezing original weights, significantly reducing trainable parameters.

Training Setup

- **Training config**: The models are fine-tuned using HuggingFace's Trainer API with different setups and configurations.
- **Training Loop**: The training set was processed in small batches with gradient accumulation, using a weighted cross-entropy loss.
- **Epochs and Monitoring**: Training ran for a few epochs, with validation metrics (e.g., loss, accuracy) monitored to prevent overfitting.

Finally, after training completion, predictions were generated on the validation set. These predictions were saved into a CSV file alongside the ground truth answers for evaluation. (Fig. 3)



img_path	question	true_answer	predicted_answer
/kaggle/input/abo-images-small/images/small/48/48e0f606.jpg	how many pink flowers are fully bloomed?	three	three
/kaggle/input/abo-images-small/images/small/2e/2e5b39c9.jpg	what shape is above the number?	circle	circle
/kaggle/input/abo-images-small/images/small/f9/f98e6c3c.jpg	is the leaf oriented vertically or horizontally?	vertically	vertically
/kaggle/input/abo-images-small/images/small/83/83f55dae.jpg	what item appears closest to the phone's camera?	bear	camera
/kaggle/input/abo-images-small/images/small/68/6835715f.jpg	what shape is it?	rectangular	rectangular
/kaggle/input/abo-images-small/images/small/e8/e893046b.jpg	what shape is it?	rectangle	rectangular

Fig 3. Prediction csv file sample records

Fine-tuning ViLTB/32

First, we fine-tuned the **ViLTB/32** base model on our custom dataset.

Custom Dataset Class To adapt the base ViLTB/32 model for our curated dataset, we implemented a custom `Dataset` class named `VQADataset`. This class reads image-question-answer triplets from the training CSV and performs the following operations:

- Preprocesses each image using `PIL`, ensuring consistent dimensions (384x384) by applying aspect-ratio-preserving padding.
- Processes each question-image pair using the `ViltProcessor`, converting them into tokenized tensors compatible with the ViLT model.
- Maps the textual answer to a corresponding class index using an `answer_to_idx` dictionary constructed from the full answer set.
- Returns a dictionary containing all model input tensors and the label for supervised training.

LoRA Configuration.

- r = 32
- lora_alpha = 64
- target_modules = [“query”, “value”]
- task_type = “SEQ_CLS”

This allowed us to fine-tune only a small fraction of parameters while retaining the pre-trained knowledge of the base ViLT model.

Training Configuration. The model was fine-tuned using HuggingFace’s Trainer API with the following setup:

- num_train_epochs = 10
- fp16 = True

After training completion, predictions were generated on the validation set. These predictions were saved into a CSV file alongside the ground truth answers for evaluation.

Similarly, as mentioned in the *General Fine-Tuning Approach* section and illustrated in the ViLTB finetuning section above, we did the fine tuning for other models too.

Fine-tuning blip-vqa-base (version-1)

Secondly, we fine-tuned the **blip-vqa-base** model on our custom dataset for different parameters.

LoRA Configuration

- `r = 8`
- `lora alpha = 32`
- `target modules = ["query", "value"]`
- `lora dropout = 0.1`
- `bias = "none"`

This setup enabled parameter-efficient fine-tuning by injecting trainable adapters into the `query` and `value` projection layers of the attention mechanism, which are key to capturing contextual relationships. The rest of the model remained frozen, preserving the pre-trained knowledge of the base BLIP model while allowing focused task-specific adaptation.

Training Configuration This was the setup used for `Trainer` API:

- `num train epochs = 3`
- `learning rate = 5e-5`
- `fp16 = True`

Fine-tuning blip-vqa-base (version-2)

LoRA Configuration

- `r = 16`
- `lora alpha = 32`
- `target modules = ["query", "value"]`
- `lora dropout = 0.1`
- `bias = "none"`

This setup is almost same as version-1 but the value r (size of rank) of matrix was changed from 8 to 16 with an intent to make the model learn with more capacity. This also enabled parameter-efficient fine-tuning by injecting trainable adapters into the `query` and `value` projection layers of the attention mechanism.

Training Configuration This was the setup used for `Trainer` API:

- `num train epochs = 3`
- `learning rate = 5e-5`
- `fp16 = True`

Fine-tuning blip-vqa-base (version-3)

LoRA Configuration

- `r = 32`
- `lora alpha = 32`
- `target modules = ["query", "value"]`
- `lora dropout = 0.1`
- `bias = "none"`

This setup is almost same as version-2 but the value r (size of rank) of matrix was changed from 16 to 32 so that the capability of the model increases even furthur.

Training Configuration This was the setup used for `Trainer` API:

- `num train epochs = 3`
- `learning rate = 5e-5`
- `fp16 = True`

Fine-tuning blip-vqa-base (version-4)

LoRA Configuration

- `r = 16`
- `lora alpha = 32`
- `target modules = ["query", "value", "key"]`
- `lora dropout = 0.1`
- `bias = "none"`

Here, even **key** was included in the target modules along with **query** and **value**. This was an attempt to increase the number of trainable parameters. Even the learning rate was reduced from 5e-5 to 2e-5.

Training Configuration This was the setup used for `Trainer` API:

- `num train epochs = 3`
- `learning rate = 2e-5`
- `fp16 = True`

Fine-tuning BLIP - 2

We even tried to fine-tune the `blip2-opt-2.7b` base model on our custom dataset.

- This version has 2.7B parameters.
- Kaggle's 16 GB GPU was insufficient the model used approximately 14 GB, causing CUDA out-of-memory errors.
- We tried to apply optimizations like:
  - Single-sample processing; no parallel processing
  - CPU offloading for some operations
  - Image resizing
  - Memory clearing after every few steps

- We then switched to the lighter **Salesforce/blip2-flan-t5-xl**, but still faced memory errors.
- Additionally, we attempted the `accelerate` offloading to CPU for more GPU memory but encountered dtype mismatch issues (`float16` inputs vs. weights) because of this.
- As a result, fine-tuning was infeasible due to memory constraints.

Evaluation

We have used the following evaluation metrics to assess the performance of our models.

Exact Match Accuracy - The percentage of model predictions that exactly match the ground truth answers.

Synonym Accuracy - The percentage of predictions that are synonymous with the true answers, even if they are not exact matches. This metric uses WordNet to identify synonyms.

Exact Match Precision - The proportion of correct exact matches out of all predictions that the model labeled as correct.

Exact Match Recall - The proportion of correct exact matches out of all true answers.

Exact Match F1 - The harmonic mean of exact match precision and recall.

BERT Score - A metric that calculates precision, recall, and F1 based on contextual embeddings to measure the semantic similarity between predicted and true answers.

Cosine Similarity - The cosine of the angle between the embedding vectors of the predicted and true answers, representing their semantic closeness.

Full evaluation table is given in the next page. .

Table 4: Evaluation Metrics for all Models on Custom VQA Dataset

Model	**EMA (%)**	**SA (%)**	**EMP (%)**	**EMR (%)**	**EMF1 (%)**	**RLF1 (%)**	**BSP (%)**	**BSR (%)**	**BSF1 (%)**	**CS (%)**
ViLT Baseline	27.53	29.61	27.53	100.0	43.18	28.02	90.73	88.68	89.54	63.54
BLIP Baseline	46.57	50.71	46.57	100.0	63.54	47.55	91.43	88.46	89.78	74.36
ViLT LoRA r32	54.46	55.78	–	–	–	–	98.98	98.70	98.82	–
BLIP LoRA V1	62.32	64.57	62.32	100.0	76.78	63.56	92.33	90.44	91.24	81.62
BLIP LoRA V2	62.38	64.68	62.38	100.0	76.83	63.67	92.38	90.45	91.28	81.65
BLIP LoRA V3	62.48	64.76	62.48	100.0	76.91	63.74	92.46	90.50	91.34	81.68
BLIP LoRA V4	59.6	62.11	59.6	100.0	74.69	61.19	93.87	92.48	93.07	80.69

- *Abbreviations*:
  - EMA: Exact Match Accuracy
  - SA: Synonym Accuracy
  - EMP: Exact Match Precision
  - EMR: Exact Match Recall
  - EMF1: Exact Match F1
  - RLF1: ROUGE-L F1
  - BSP: BERTScore Precision
  - BSR: BERTScore Recall
  - BSF1: BERTScore F1
  - CS: Cosine Similarity