Visual Recognition Mini Project 2 - Report

Vansh Sinha IMT2022122
Ishan Singh IMT2022124

May 2025

Abstract

This report presents our approach to the AIM825 course project **Multimodal Visual Question Answering with Amazon Berkeley Objects Dataset**.

1 Introduction

This project explores the construction and evaluation of a multimodal VQA system using the Amazon Berkeley Objects (ABO) dataset, which contains 147,702 product listings and 398,212 catalog images with multilingual metadata.

The primary objectives include:

- Generating diverse and meaningful VQA dataset using image and metadata information.
- Evaluating baseline VQA models without fine-tuning.
- Fine-tuning selected models using Low-Rank Adaptation (LoRA) for parameter-efficient training.
- Assessing the performance of all models using standard and additional evaluation metrics.

This report details our methodology for dataset curation, model selection, fine-tuning strategy, and experimental evaluation. All dataset curation, training and inference is done on Kaggle on 2xT4 GPU.

2 Dataset Curation

2.1 VQA Dataset from ABO

We have used the Amazon Berkeley Objects (ABO) Dataset to create a VQA dataset which will be used in fine-tuning.

2.1.1 Preprocessing

We aggregate all 16 JSON shards of ABO product listings (located under `abo-listings/listings/metadata`) into memory and extract:

- **product_type**: the primary category label,
- **main_image_id**: the identifier of the main image,
- **other_image_id**: a comma-separated list of secondary image IDs.

These fields are collated into a Pandas DataFrame `df` for downstream processing. We also observed that the images related to other_image_id are of the same object as in the main_image_id image but with a different view. This can be seen in the Fig. 1. Here leftmost image is from main_image_id and the rest are from its other_image_id section. Thus to keep variety among selected images and to avoid any duplicates any row whose `main_image_id` appears in another product's `other_image_id` list is removed prior to merging.

Figure 1: Relation between main_image_id and other_image_id

We read the image metadata CSV (`images.csv`) from `/abo-small/images/metadata`, which provides `image_id`, relative `path`, and dimensions. After renaming `image_id` to `main_image_id`, we left-merge with `df`, drop unnecessary columns (`height`, `width`, `other_image_id`), and prepend the base directory `abo-small/images/small/` to form a complete `path` for each image.

2.1.2 Sampling Strategies

We experimented with 2 different techniques for sampling images out of our processed image dataset:

1. **Random Oversampling**: For each category in `product_type`, we sample (with replacement) a fixed number of images (25000), equalizing class frequencies. Since it is sampled with replacement so in those classes where the no of images is less than (25000/no of classes) we have duplicated the data, relying on the fact that in the vqa generation through the API call, the duplicated images will have different sets of questions.

2. **Custom Stratified Sampling**: Here we implement an iterative, "fair-share" sampling strategy: Initially n=25000 and this strategy repeatedly divides the remaining number of images you need (n) equally across all product types (C categories), then for each category it randomly picks up to that share (I = $\lfloor n/C \rfloor$ items) or all remaining items if the category has fewer than I. Those sampled rows are removed from the pool, and n is reduced by how many were taken. This process loops—recomputing category counts, recalculating each share, and sampling again—until we have drawn our target total (or we can't sample any more because all categories are exhausted).

The resulting image paths from each method are exported as `oversampled_paths.csv` and `sampled_paths.csv` for prompt-based VQA dataset generation.

2.1.3 Prompt-Based VQA Generation

From each sampled image pool, we generate question–answer pairs by calling Google's Gemini 2.0 Flash API with concise, attribute-focused prompts. Prompts were designed to produce 5 question-answer pairs per product image with the answer being one word about a specific visual attribute (e.g. color, material, shape, brand, etc.). It also ensures that no digits are returned as answer rather they are alphabetic. The API returns JSON entries {`"question"`, `"answer"`}, which we pair with the corresponding image path. These form our fine-tuning dataset. The prompts were also refined using Gemini 2.0 Flash API.

2.2 VQA Dataset from SQID

We did not want to use the same ABO dataset for evaluations as there are many classes which are fully included in our fine-tuning datasets (Section

2.1.1), so taking a complement of the images used in fine-tuning from the ABO dataset will lead to the case where many classes are not included in the test dataset and then the test dataset will be biased with images from only the high cardinality classes. Thus we have used another dataset i.e. Shopping Queries Image Dataset (SQID) to create another VQA dataset which will be used in evaluating the baseline and fine-tuned models.

We load the SQID product–image manifest directly from Hugging Face in Parquet format, drop any entries with missing URLs, and uniformly sample 1,000 records (random seed 42) for our evaluation pool. Each sampled URL is fetched via HTTP into a local `images/` directory, with progress tracked by `tqdm`. Images are then opened with PIL to confirm integrity; any failed or corrupted files are discarded.

Using the Gemini 2.0–Flash API, we invoke the same prompt that we used in VQA generation through ABO dataset, to get question answer pairs for images from SQID. Then these question-answer pairs are linked with their respective image paths and thus, a new VQA dataset is generated for evaluations from SQID.

3 Evaluation Metrics

Evaluating Visual Question Answering (VQA) models with single-word answers can be challenging due to the variability in expression, even when semantically correct. Hence, we adopted a combination of standard and semantic-aware metrics to holistically assess model performance.

3.1 Standard Metrics

- **Accuracy**: Measures the proportion of exact matches between predicted and ground-truth answers. It is calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of questions}}$$

  This metric is strict and only considers a prediction correct if it matches the ground-truth token exactly.

- **Macro F1 Score**: Since some answers appear less frequently than others, macro-averaged F1 is useful to balance the influence of each class. It computes precision and recall independently for each label and then averages them, making it more robust to class imbalance.

3.2 Additional Metrics

1. BERT F1 Score

BERT F1 measures semantic similarity using contextual embeddings from a pre-trained BERT model. It compares the similarity of predicted and ground-truth answers at the embedding level, providing a more flexible and meaningful evaluation.

Justification:

- Useful when predictions are semantically correct but not token-exact (e.g., *"sofa"* vs. *"couch"*).
- Captures contextual meaning even for single-word answers by leveraging subword embeddings.

Advantage: Compared to accuracy or F1, BERT F1 tolerates synonyms and paraphrases, giving credit to answers that are correct in meaning but fail exact string comparison.

2. Wu-Palmer Similarity (WUPS)

Wu-Palmer Similarity measures the depth of two concepts in a taxonomy (like WordNet) and how similar they are semantically. It is defined as:

$$\text{WUPSim}(a, b) = \frac{2 \times \text{depth}(\text{LCS}(a, b))}{\text{depth}(a) + \text{depth}(b)}$$

Justification:

- Designed for comparing single-word nouns, making it especially suitable for VQA with single word answers.
- Recognizes hyponyms/hypernyms (e.g., *"animal"* vs. *"dog"*).

Advantage: Provides a graded score (between 0 and 1) indicating how semantically close the prediction is to the ground truth, which is especially helpful for ontology-aware evaluation.

3.3 Why Semantic Metrics Matter

Since our dataset focuses on single-word answers derived from visual content (object type, material, color, etc.), exact match metrics may penalize

near-correct answers unfairly. BERT F1 and WuPalmer Similarity help mitigate this by accounting for semantic correctness, offering a more nuanced assessment of model performance.

We report all four metrics to ensure both syntactic precision and semantic understanding are evaluated fairly.

4 Models Used

The following models were used for the project:

- Salesforce/blip-vqa-capfilt-large (385M baseline parameters)
- google/paligemma-3b-pt-224 (2.9B baseline parameters)
- dandelin/vilt-b32-finetuned-vqa (118M baseline parameters)

As per the project constraints, we were required to work within a total parameter budget of 7B parameters. The initial plan was to include one small model (under 1B parameters), one medium model (around 3B parameters), and one large model (6–7B parameters) to cover a wide range of model scales. However, during initial testing, we observed that large models incurred extremely high inference times, indicating that fine-tuning them would be computationally expensive and time-consuming.

As a result, we revised our strategy to balance performance with practicality. The final selection consists of two small models and one medium-sized model. This configuration allows for faster experimentation and feasible fine-tuning within our hardware and time constraints, while still maintaining architectural diversity and performance for vision-language tasks.

5 Fine-Tuning with LoRA

Based on the evaluation scores from Wu-Palmer Similarity and BERTScore (see next section) , **BLIP** and **PaliGemma** showed promising performance in absolute terms. Additionally, since **ViLT** utilizes a classifier head rather than free-form generation, it was deemed less suitable for our open-ended question answering objective. Hence, we selected only BLIP and PaliGemma for fine-tuning.

LoRA Implementation Details

- Applied Low-Rank Adaptation (LoRA) with rank 8 to fine-tune attention layers of transformer-based models.

- Only the low-rank matrices were trained; base model weights remained frozen which reduced trainable parameters, enabling efficient training on limited hardware.

Training Configuration and Hardware

- Training was conducted using Hugging Face's `Trainer` and `TrainingArguments` classes with PyTorch backend.
- Hardware: NVIDIA 2xT4 Kaggle GPU.
- Key configurations:
  - `epochs = 1` for Paligemma; `epochs = 5` for BLIP.
  - `per_device_train_batch_size = 1` for Paligemma; `batch_size = 32` for BLIP.
  - `save_steps = eval_steps = 14500` for Paligemma; `save_steps = eval_steps = 100` for BLIP.
    This means that after every 14500 (or 100) training steps, a full evaluation was performed and the evaluation loss was computed once. Checkpoints were saved at the same frequency.
- Early stopping was used:
  - `EarlyStoppingCallback` with `patience = 1` and `threshold = 0.0` for PaliGemma.
  - For BLIP, patience was increased to 5 to allow more training iterations.
- Key-Value (KV) caching was enabled by default when calling the model.generate() function.

Challenges Encountered

- Quantization could not be applied effectively, as NVIDIA T4 GPUs do not support `bfloat16`, which is essential for many quantization workflows.
- `fast_attention_v2` is unsupported on T4; `fast_attention_v1` is enabled by default in the Transformers library.

6 Baseline Evaluation

The following table presents the performance scores of the **baseline models** on our curated SQID dataset.

Table 1: Baseline Model Evaluation Results

Model	Accuracy	Macro F1	Wu-Palmer Sim	BERT F1
BLIP	0.250	0.086	0.717	0.859
PaliGemma	0.322	0.180	0.770	0.868
VILT	0.217	0.057	0.711	0.869

7 FineTuned Evaluations

The following table presents the performance scores of the **fine-tuned models** on our curated SQID dataset. Since we had two different sampling techniques for creating the fine-tuning dataset, each model has 2 sets of scores based on which of the 2 sampled dataset it was fine-tuned on.

Table 2: Fine-Tuned Model Performance by Sampling Technique

Sampling	Model	Accuracy	Macro F1	Wu-Palmer Sim	BERT F1
Random Oversampled	PaliGemma	0.741	0.469	0.905	0.943
	BLIP	0.456	0.126	0.786	0.879
Custom Sampled	PaliGemma	0.739	0.452	0.904	0.940
	BLIP	0.456	0.131	0.797	0.880

8 Results and Analysis

The evaluation results clearly demonstrate the substantial improvements achieved through fine-tuning both the PaliGemma and BLIP models. When compared to their respective baselines, both models show gains across all metrics—Accuracy, Macro F1, Wu-Palmer Similarity (WUPS), and BERT F1. These improvements validate the effectiveness of the fine-tuning process, especially when trained on a carefully sampled dataset.

For PaliGemma, the improvement was most significant. Fine-tuning on the random oversampled dataset more than doubled the model's accuracy and Macro F1 score. Semantic metrics also showed strong gains, with BERT F1 increasing by over 8.6% and WUPS by approximately 17.5%. This suggests that fine-tuning enabled the model not only to improve exact answer

matching but also to better understand semantic equivalence, which is particularly important for single-word answers in a visual question answering setting.

BLIP also benefited from fine-tuning, though to a lesser extent. Accuracy and Macro F1 increased notably, and both BERT F1 and WUPS saw modest improvements. While BLIP's performance improved overall, its semantic gains were smaller than those of PaliGemma, indicating that PaliGemma was more responsive to the fine-tuning process, especially in capturing nuanced visual semantics.

When comparing the two sampling techniques, PaliGemma performed slightly better when trained on the random oversampled dataset. Although the gains over the custom sampled variant were marginal, the oversampling approach yielded consistently higher metrics, making it the preferred method for this model. In contrast, BLIP showed a slight advantage when trained on the custom sampled dataset, particularly in WUPS and Macro F1.

Considering all models and sampling methods, the best-performing configuration was the PaliGemma model fine-tuned on the random oversampled dataset. This model exhibited the highest levels of both exact and semantic accuracy, confirming its robustness and generalization capabilities. Consequently, it was chosen as the final model for submission and further evaluation.