Multimodal Visual Question Answering with Amazon Berkeley Objects Dataset

T2-24-25 - AIM 825 - Visual Recognition - Sec A

May 2025

Authors

Ayush Kashyap (IMT2022129)
Rishit Mane (IMT2022564)
Rohan Rajesh (IMT2022575)

Contents

1 Introduction

This project centers on developing a Visual Question Answering (VQA) system utilizing the Amazon Berkeley Objects (ABO) dataset. The primary objectives include curating a dataset focused on single-word answers, evaluating state-of-the-art pre-trained models such as BLIP-2 and CLIP, and fine-tuning them using Low-Rank Adaptation (LoRA). To assess model performance, we employed metrics like Accuracy and F1 Score. All experiments were conducted on Kaggle's free cloud GPU infrastructure, adhering to the constraint of a 7-billion-parameter limit. This report outlines the complete workflow—from dataset preparation and model selection to performance evaluation and fine-tuning—while also discussing the challenges faced and contributions made throughout the process.

2 Dataset Overview

The Amazon-Berkeley Objects (ABO) dataset serves as a comprehensive resource tailored for multimodal learning tasks such as Visual Question Answering (VQA). Designed with e-commerce applications in mind, the dataset offers an extensive collection of product data. For our work, we opted for a streamlined variant that strikes a balance between detail and storage constraints. This version includes 147,702 product entries and 398,212 associated images, providing a robust foundation for training and evaluating VQA models effectively.

1. abo-images-small

We utilized the `abo-images-small` version, which is a compact and downsampled alternative to the original 100GB image dataset, bringing the size down to approximately 3GB. The images in this set are resized to a maximum of 256x256 pixels, making them ideal for quick experimentation and resource-efficient training. Accompanying the images is a metadata CSV file that maps each primary image ID to its corresponding file path and lists all related images. This structured format simplifies image retrieval and facilitates consistent pairing of product visuals with their associated data.

2. abo-listings

The `abo-listings` dataset contains enriched product metadata formatted in JSON, totaling around 83MB in size. Each record includes a variety of categorical and descriptive attributes such as `brand`, `color`, `item_id`, `product_type`, `main_image_id`, `other_image_id`, and `item_keywords`, among others. These fields provide essential semantic context that complements the visual data, enabling the generation of informative and diverse VQA pairs. The metadata enhances the model's ability to understand product-related queries by linking textual and visual elements seamlessly.

3 Data Curation and Processing Overview

The objective of the data curation process was to construct a high-quality Visual Question Answering (VQA) dataset with single-word answers. This was achieved using the Amazon Berkeley Objects (ABO) dataset's `abo-images-small` variant, comprising 3GB

of data across 398,212 images, each with a resolution of 256x256 pixels. Additionally, the `abo-listings` metadata (83MB) was utilized to provide textual context for each image. The goal was to generate diverse and visually answerable question-answer pairs that could effectively fine-tune and evaluate VQA models within the computational limits of Kaggle's free GPUs. Gemini 2.0 API, known for its advanced multimodal capabilities, was employed to process images and metadata, resulting in a dataset that maintains a balance between diversity, difficulty, and domain relevance for e-commerce products.

3.1 Metadata Retrieval and Cleaning

The metadata extraction and filtering process involved the following key steps:

- The `abo-listings` metadata was provided as compressed JSON files. These files were decompressed using Python's `gzip` and `shutil` libraries, resulting in uncompressed JSON files that were more manageable for processing.
- Each JSON file was inspected to validate its structure, ensuring it consisted of line-delimited JSON objects that could be processed individually.
- Records were filtered to retain only those entries containing essential attributes such as `main_image_id`, `brand`, `color`, and `product_type`. This filtering process was further refined to focus on listings from specific countries (India and the US), ensuring consistency in English-language prompts.
- Descriptions were compiled by aggregating data from relevant metadata fields, including `bullet_point`, `color`, `product_type`, and `item_keywords`. Priority was given to entries in English (`en_IN`, `en_US`) to align with the intended language for prompts.
- The cleaned metadata was transformed into structured CSV files containing key attributes such as `main_image_id`, `overall_description`, `colour_description`, and `other_description`.
- Metadata entries were then associated with corresponding images from the `abo-images-small` dataset using the `main_image_id` as the linking key.

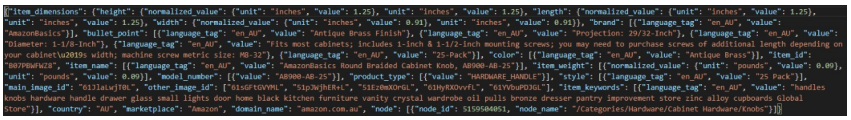
{"item_dimensions": {"height": {"normalized_value": {"unit": "inches", "value": 1.25}, "unit": "inches", "value": 1.25}, "length": {"normalized_value": {"unit": "inches", "value": 1.25}, "unit": "inches", "value": 1.25}, "width": {"normalized_value": {"unit": "inches", "value": 0.91}, "unit": "inches", "value": 0.91}}, "brand": [{"language_tag": "en_AU", "value": "AmazonBasics"}], "bullet_point": [{"language_tag": "en_AU", "value": "Antique Brass Finish"}, {"language_tag": "en_AU", "value": "Projection: 29/32-Inch"}, {"language_tag": "en_AU", "value": "Diameter: 1-1/8-Inch"}, {"language_tag": "en_AU", "value": "Fits most cabinets; includes 1-Inch & 1-1/2-Inch mounting screws; you may need to purchase screws of additional length depending on your cabinet\u2019s width; machine screw metric size: M3-.32"}, {"language_tag": "en_AU", "value": "25-Pack"}], "color": [{"language_tag": "en_AU", "value": "Antique Brass"}], "item_id": "B07P8WFWZ8", "item_name": [{"language_tag": "en_AU", "value": "AmazonBasics Round Braided Cabinet Knob, AB900-AB-25"}], "item_weight": [{"normalized_value": {"unit": "pounds", "value": 0.09}, "unit": "pounds", "value": 0.09}], "model_number": [{"value": "AB900-AB-25"}], "product_type": [{"value": "HARDWARE_HANDLE"}], "style": [{"language_tag": "en_AU", "value": "25 Pack"}], "main_image_id": "61JlaLwjT0L", "other_image_id": ["61sGFtGVYML", "51pJWjhERkL", "51Ez0mXOrGL", "61HyRXOvvFL", "61YVbuPOJGL"], "item_keywords": [{"language_tag": "en_AU", "value": "handles knobs hardware handle drawer glass small lights door home black kitchen furniture vanity crystal wardrobe dil pulls bronze dresser pantry improvement store zinc alloy cupboards global store"}], "country": "AU", "marketplace": "Amazon", "domain_name": "amazon.com.au", "node": [{"node_id": 5195584051, "node_name": "/Categories/Hardware/Cabinet Hardware/Knobs"}]}

Figure 1: Sample mapping of metadata to corresponding product image.

3.2 Subset Extraction from Dataset

Given the computational limitations, a subset of approximately 10,000 product listings was selected from the filtered metadata. This subset was carefully curated to ensure diversity across major product categories, including clothing, electronics, and furniture. The subset selection ensured that each product type was well-represented, enabling the generation of a varied set of question-answer pairs. Each selected listing was linked to its respective image using the `main_image_id` key.

3.3 Designing Input Prompts and Generating Queries

To facilitate effective question-answer generation, a custom prompt was crafted for the Gemini 2.0 API. The aim was to generate five diverse questions per image, each yielding a concise, single-word answer. The prompt design focused on three main aspects:

- **Diversity:** Questions targeted various visual features, such as color, shape, quantity, spatial positioning, and relative size.
- **Difficulty:** Each image was associated with two simple questions, two moderately challenging questions, and one complex question involving nuanced visual reasoning.
- **Constraints:** Questions were limited to visually discernible attributes, avoiding abstract or contextual content unless it was visually apparent. Responses were strictly limited to single words, discouraging open-ended or yes/no answers unless necessary.

Metadata attributes like `overall_description` and `colour_description` were incorporated into the prompts to provide contextual information. The structured outputs generated by Gemini 2.0, consisting of image paths, questions, and answers, were organized into a CSV format with the following columns: `image_id`, `full_image_path`, `question`, and `answer`.

3.4 Data Refinement and Integrity Verification

Quality control measures were implemented to ensure dataset integrity and relevance:

- The existence of each image was verified using the `images.csv` mapping file. Missing or corrupted images were excluded from the dataset.
- A manual review was conducted to validate the relevance and clarity of generated questions and their corresponding answers. Multiple prompt versions were tested to optimize question quality and answer conciseness.
- Metadata inconsistencies, such as missing or ambiguous color descriptors, were addressed through visual inspection or were excluded if irreparable.
- The Gemini API's rate limits (1,500 requests per day) and request delays (60 seconds per request) were adhered to, ensuring smooth progress tracking and resumption capabilities in case of interruptions.

4 Data Exploration of the Compiled Dataset

The compiled dataset encompasses a comprehensive collection of **159,150 individual records**, each carefully curated to support in-depth data exploration. Every record is structured with four essential fields: `image_id`, a unique identifier assigned to each image to enable accurate tracking and referencing; `full_image_path`, which specifies the complete file path for accessing the image within the dataset's storage framework; `question`, a targeted query crafted to extract specific information about the image's visual elements; and `answer`, a succinct response that addresses the query, encapsulating key observations or details. A distinctive feature of the dataset is that each image is

associated with exactly five unique question-answer pairs, with each pair recorded as a separate entry. This design amplifies the dataset's analytical potential by providing multiple inquiry angles for a single image, thereby enriching its applicability for tasks such as visual question answering, attribute identification, or machine learning model training. The structure fosters a versatile framework, allowing models to leverage diverse query-response combinations tied to a single visual source.

4.1 Image Dimension Insights

The dataset reveals a distinct pattern in the dimensions of its images, with a strong preference for specific height and width combinations. Out of the total image collection, the most prevalent size is **256 pixels in height by 219 pixels in width**, accounting for **129,540 images**, or approximately **81.39%** of the dataset. The next most common dimension is **256 pixels by 134 pixels**, comprising **17,535 images** (**11.02%**). Images measuring **256 pixels by 256 pixels** follow, with **7,805 instances** (**4.90%**). A smaller group of **385 images** (**0.24%**) have dimensions of **256 pixels by 197 pixels**. Finally, the least frequent size is **122 pixels by 256 pixels**, occurring in just **110 images** (**0.07%**).

Notably, the vast majority of images share a consistent height of **256 pixels**, with only a minor fraction deviating to **122 pixels**. This uniformity in height, despite variations in width, streamlines preprocessing and analysis tasks, as it reduces the need for extensive resizing or normalization across the dataset.

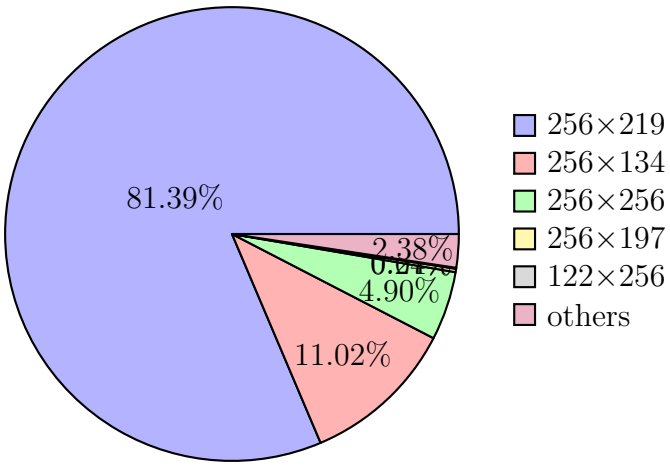

Figure 2: Image Dimension Distribution (Pie Chart)

4.2 Analysis of Queries

The queries in the dataset vary in terms of word count and format. On average, each query contains 4 words, with around 8,048 being single-word queries. The lengthiest query spans 19 words, though this is an exception.

An examination of the queries reveals several recurring patterns in their structure. Queries that begin with "what" often seek details about visual elements, such as the color or form of objects within the image. In contrast, queries starting with "is" are usually binary, expecting a yes or no answer, as exemplified by questions like "Is there a ...?"

Additionally, queries that commence with "how" commonly address counts, seeking to determine the number of specific elements present, as seen in "How many ...?"

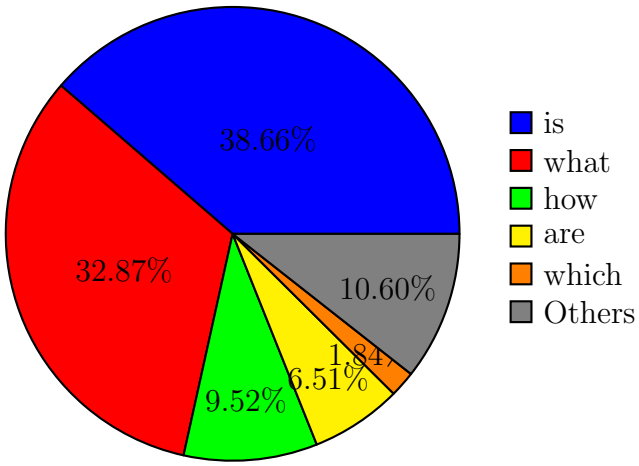

Figure 3: Distribution of Starting Words in Queries

4.3 Analysis of Response Categories

The dataset analysis indicates that each response is a single word, categorized into three distinct groups based on their characteristics. Approximately 45% of these responses are affirmative or negative terms, such as "yes" or "no," which signify agreement or disagreement with a given prompt. Another 10% of the responses consist of numerical values, providing quantitative insights within the dataset. The remaining 45% are descriptive terms that capture attributes like colors, shapes, or other observable features, offering qualitative details about the subject matter. A comprehensive overview of the most frequent responses is included to underscore their distribution and importance in the dataset.

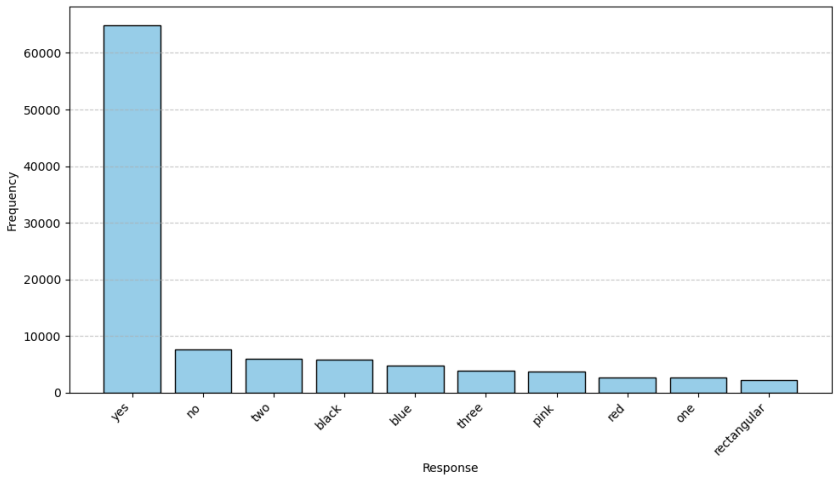

Figure 4: Commonly Appearing Responses

5 Curated Dataset Preprocessing Steps

The preprocessing of the curated dataset involved several steps to enhance data quality and balance. Initially, 54 data points with missing responses were eliminated to ensure completeness. Exploratory data analysis revealed that approximately 45% of responses were binary "yes" or "no" answers, which were subsequently downsampled to constitute only 2% of the dataset, resulting in 89,785 remaining data points. To improve question quality, 8,048 single-word questions were removed. To address imbalances observed in the answer distribution (as shown in Table 3), frequent response types were randomly downsampled to prevent any single answer from dominating, as visualized in the accompanying figure. This filtering reduced the dataset to 75,484 data points. From this, 70,000 data points were randomly selected for further analysis. Finally, these were divided into training and validation sets using an 80:20 split, yielding 56,000 data points for training and 14,000 for validation.

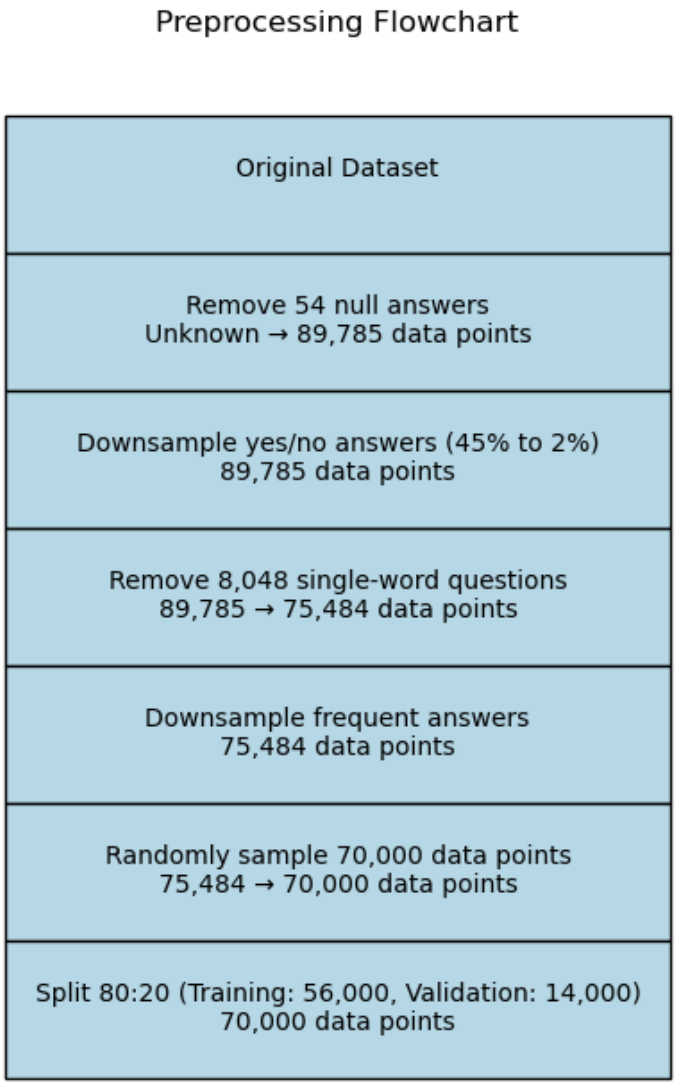

Figure 5: Flowchart of the dataset preprocessing steps

6 Key Metrics for VQA Model Evaluation

The evaluation of a Visual Question Answering (VQA) model requires multiple metrics to assess various dimensions of performance. Below is a detailed summary of the primary metrics employed to evaluate different models.

6.1 F1 Score

The F1 Score represents the harmonic mean of precision and recall, computed at the token level. It effectively balances false positives and false negatives, providing a robust measure of answer quality, particularly in cases where predictions are partially correct. The F1 Score is especially insightful for multi-token answers, where token-level granularity is critical.

6.2 BERTScore

BERTScore leverages contextual embeddings from a pre-trained BERT model to calculate precision, recall, and F1 scores. This approach enables the evaluation of semantic similarity beyond mere surface-level token matching, capturing nuances such as synonyms and paraphrases. BERTScore is particularly valuable for assessing answers that are semantically accurate but lexically distinct from the reference.

6.3 METEOR Score

The METEOR metric incorporates synonymy, stemming, and word order when comparing predicted answers to reference answers. Its flexibility surpasses that of exact match metrics, making it a standard in machine translation for its ability to recognize partial matches and semantic similarities. METEOR's alignment-based scoring is especially effective for evaluating short, diverse responses.

6.4 ROUGE-L F1 Score

ROUGE-L evaluates the longest common subsequence (LCS) between predicted and reference answers, prioritizing fluency and sequential overlap without demanding exact token matches. While it balances precision and recall, its performance is less effective for short answers, particularly single-word responses. Nevertheless, ROUGE-L remains valuable for assessing longer, sentence-level outputs.

7 Baseline Models for VQA Evaluation

To assess our curated Visual Question Answering (VQA) dataset, we evaluated several pre-trained multimodal models capable of processing both images and text. All models were tested in their pre-trained state without fine-tuning to ensure a fair and consistent baseline for comparison. The models selected for evaluation are as follows:

1. **BLIP**: Chosen for its specialized design for VQA tasks, BLIP offers a compact architecture and delivers robust performance without requiring modifications.

2. **BLIP-2**: An advanced version of BLIP, this model was included due to its enhanced capabilities and superior effectiveness in VQA applications.

3. **ViLT**: Selected for its efficient handling of visual-language tasks, ViLT eliminates heavy image encoders, making it ideal for environments with constrained computational resources.

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

- **Overview**: The initial baseline model evaluated is **BLIP** (Bootstrapping Language-Image Pre-training), utilizing the `Salesforce/blip-vqa-base` checkpoint from Hugging Face Transformers. With around 223 million parameters, BLIP is engineered for Visual Question Answering (VQA), image captioning, and image-text retrieval tasks.

- **Model Structure**: BLIP employs a vision transformer (ViT) to process images and a BERT-based text encoder. A multimodal encoder integrates these modalities to generate answers, leveraging pre-training on noisy web datasets and curated collections like COCO and Visual Genome through a bootstrapping approach.

- **Image Handling**: Images are resized to 224×224 pixels, divided into patches, and transformed into embeddings via the vision transformer for further processing.

- **Text Handling**: Input questions are tokenized using a BERT tokenizer, then combined with visual embeddings to facilitate VQA task predictions.

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

- **Overview**: The second baseline model evaluated is **BLIP-2** (Bootstrapping Language-Image Pre-training - 2), utilizing the `Salesforce/blip2-opt-2.7b` checkpoint from Hugging Face Transformers. With approximately 2.7 billion parameters, BLIP-2 incorporates a robust language model and employs a two-stage pre-training approach to surpass the performance of its predecessor, BLIP.

- **Model Structure**: BLIP-2 integrates a vision transformer (ViT) with a large-scale language model (OPT-2.7B) through a Querying Transformer (Q-Former). The Q-Former extracts key visual features from images and relays them to the language model. Its two-stage pre-training—focusing on vision-language alignment and generative learning—enhances its effectiveness for single-word VQA tasks.

- **Image Processing**: Images are resized to 224×224 pixels and processed by the ViT. The Q-Former then filters and selects the most relevant visual tokens to forward to the language model.

- **Text Processing**: Input questions are tokenized and fed into the OPT-2.7B model, which produces answers by combining textual input with the selected visual tokens.

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

- **Overview**: The third baseline model evaluated is **ViLT-B/32** (Vision-and-Language Transformer), utilizing the `dandelin/vilt-b32-finetuned-vqa` checkpoint from Hugging Face Transformers. With approximately 87 million parameters, ViLT is among the most lightweight multimodal transformer models tested, efficiently integrating visual and textual inputs through a unified processing approach.
- **Model Structure**: ViLT employs a single transformer encoder that jointly processes image patches and text tokens. Images are divided into patches and projected into a shared embedding space with text, enabling seamless processing by the transformer. The model leverages the `ViltProcessor` for preprocessing both modalities.
- **Image Processing**: Input images are segmented into 32×32 pixel patches, which are linearly mapped to the same dimensional space as text embeddings for integrated processing.
- **Text Processing**: Text inputs are tokenized using a WordPiece tokenizer with a vocabulary of 30,000 tokens, preparing them for joint processing with visual data.

8 Fine-Tuning with LoRA

Following the evaluation of pre-trained models, we fine-tuned these models on our curated Visual Question Answering (VQA) dataset to enhance their performance. To optimize adaptation to our domain-specific task, we employed Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning technique. LoRA introduces small, trainable low-rank matrices into critical layers of the pre-trained model, typically the attention mechanisms, while keeping most original weights fixed. This approach enables the model to learn task-specific patterns with minimal trainable parameters and reduced computational resources.

8.1 Rationale for Adopting LoRA

The decision to utilize LoRA was driven by its numerous advantages, tailored to our fine-tuning needs:

- **Computational Efficiency**: LoRA modifies only a small fraction of the model's parameters, facilitating rapid and resource-efficient fine-tuning.
- **Minimal Resource Requirements**: Operating in a constrained environment, such as Kaggle with 16GB GPUs, LoRA's selective parameter updates significantly lower memory and computational demands compared to full model fine-tuning.
- **Training Approach**: LoRA approximates weight updates using the formula $\Delta W = A \cdot B$, where A and B are compact low-rank matrices. These updates are integrated with the frozen base weights during training, incurring minimal overhead.
- **Suitability for VQA**: Applied to our VQA dataset, LoRA enabled models like BLIP and ViLT to better capture domain-specific features, such as colors, shapes, and layouts, which are less prevalent in general-purpose datasets like COCO used for pre-training models like BLIP and BLIP-2.

8.2 Fine-Tuning Approach

To enhance the performance of vision-language transformer models for Visual Question Answering (VQA), we fine-tuned them using Low-Rank Adaptation (LoRA) on a dataset derived from the Amazon Berkeley Objects (ABO) collection. The training was tailored for Kaggle's dual 16 GB GPU environment, utilizing the `peft` (Parameter-Efficient Fine-Tuning) library to integrate lightweight adapters into the models. This approach delivered robust results with minimal trainable parameters and low resource demands. Additionally, we crafted a bespoke data pipeline to ensure seamless compatibility with the ViLT model's architecture and LoRA's requirements.

8.2.1 Data Processing and Structuring

The VQA dataset, provided in CSV format, was meticulously prepared to support effective model training and evaluation. Each entry included an image identifier, a natural language question, and a single-word answer. To ensure balanced representation of question types (e.g., color, quantity, material, shape) and product categories, the dataset was divided into 80% training and 20% validation sets. A custom PyTorch dataset class, named `VQADataHandler`, was developed to streamline data management and preprocessing, with the following key functionalities:

- **Data Partitioning**: The 80:20 train-validation split maintained proportional representation of diverse question and product types, fostering model generalization and reliable evaluation.
- **Image Acquisition**: Images, sourced from the `abo-images-small` dataset with 256×256 RGB resolution, were accessed using reconstructed paths from image IDs, with verification to ensure file integrity.
- **Image Transformation**: Images were resized to 224×224 pixels with aspect-ratio-preserving padding. Training incorporated augmentations like random cropping and normalization, while a fallback mechanism replaced missing or corrupted images with blank RGB placeholders to ensure continuous training.
- **Text Tokenization**: Questions were tokenized using a HuggingFace tokenizer aligned with the pre-trained model, producing input IDs and attention masks. Padding and truncation to a maximum length (e.g., 128 tokens) ensured uniform batch processing.
- **Answer Mapping**: Single-word answers were converted to unique indices or token IDs using a vocabulary derived from the training data, padded or truncated to a fixed length (e.g., 32 tokens) for consistent batch formats.
- **Output Organization**: The dataset class delivered a dictionary with image tensors, tokenized question tensors, attention masks, and encoded answers, formatted to eliminate unnecessary batch dimensions for compatibility with the training pipeline.

8.2.2 LoRA-Enhanced Model Setup

The ViLT model was adapted using LoRA to optimize fine-tuning within the constraints of the Kaggle GPU environment. The `peft` library facilitated the integration of LoRA

adapters, enabling efficient training with minimal resource overhead. The configuration prioritized memory efficiency and flexibility, with the following details:

- **Adapter Implementation**: LoRA adapters were inserted into the ViLT model's self-attention layers, keeping the base parameters frozen while training only the adapter weights, significantly reducing the number of trainable parameters.
- **Adapter Settings**: Configured with a rank of 32, the LoRA adapters balanced expressiveness and efficiency. The `peft` framework's modularity supported experimentation with adapter placement and dropout rates.
- **Resource Optimization**: PyTorch's Automatic Mixed Precision (AMP) was employed for mixed-precision training, reducing memory usage and accelerating computation. Combined with LoRA's lightweight adapters, this ensured training fit within the Kaggle environment's memory limits.

8.3 Training Setup

- **Training Strategy**: The model was fine-tuned using mini-batches with gradient accumulation to manage memory constraints. To counter class imbalance—particularly among frequently occurring answers such as common object colors—a weighted cross-entropy loss function was employed. Optimization was handled using AdamW, ensuring stable convergence throughout training.
- **Epochs and Validation**: The fine-tuning process spanned several epochs. Validation metrics such as loss and accuracy were monitored after each epoch to track generalization performance. An early stopping criterion was used to halt training when validation improvement stagnated, helping to prevent overfitting.
- **Computational Environment**: Experiments were conducted on a Kaggle environment with dual 16 GB NVIDIA GPUs. Mixed-precision training was utilized via PyTorch AMP to improve computational efficiency and reduce memory usage. Lightweight LoRA adapters, integrated through the `peft` library, allowed efficient parameter updates while keeping the majority of the base model frozen.

8.4 Training Loop

The training loop runs for a predefined number of epochs (3 in this case) and iterates over the dataset in batches. Eighty percent of our handcrafted dataset was utilized to assist in fine-tuning the model. During each epoch:

- **Forward Pass**: For each batch, the model performs a forward pass where the image-question pairs are processed.
- **Backward Pass**: The loss is calculated, and gradients are backpropagated to update the model's parameters.
- **Loss Calculation and Optimization**: The loss is calculated, and the optimizer updates the model parameters accordingly. We use a learning rate scheduler to ensure that the learning rate decays appropriately during training.

The training time per epoch is tracked, and the average loss is computed to monitor progress. At the end of each epoch, the model's performance is evaluated, and the model is saved periodically.

8.5 Fine-Tuning ViLTB/32

The fine-tuning of the ViLTB/32 model was tailored to a custom Visual Question Answering (VQA) dataset comprising image-question-answer triplets. This process involved creating a specialized data pipeline and applying Low-Rank Adaptation (LoRA) to efficiently adapt the model while leveraging its pre-trained capabilities. The following subsections detail the dataset preprocessing and the LoRA-based fine-tuning configuration.

```python
# LoRA config and apply
lora_config = LoraConfig(
    r=32,
    lora_alpha=64,
    target_modules=["query", "value"],
    lora_dropout=0.1,
    bias="none",
    task_type="SEQ_CLS"
)
model = get_peft_model(model, lora_config)
print("LoRA applied now")
model.print_trainable_parameters()
```

```
LoRA applied now
trainable params: 2,707,651 || all params: 115,830,662 || trainable%: 2.3376
```

```python
os.environ["WANDB_DISABLED"] = "true"

# Training arguments
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=10,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    learning_rate=4e-5,
    logging_dir="./logs",
    logging_steps=10,
    fp16=True,
    remove_unused_columns=False,
    disable_tqdm=False,
    save_strategy="epoch",
    eval_strategy="epoch",  # Evaluate on validation set each epoch
    load_best_model_at_end=True,
    metric_for_best_model="accuracy",
)
```

Figure 6: Fine-tuning the VILT model using LoRA with rank $r = 32$ and LoRA alpha of 64. The configuration targets the `query` and `value` modules for adaptation in a sequence classification task. Training is set up with Hugging Face's `Trainer`, leveraging half-precision (fp16) training and evaluation based on validation accuracy.

8.5.1 Dataset and Preprocessing

To prepare the custom VQA dataset for training, a dedicated PyTorch dataset class, `VQADataHandler`, was developed to manage preprocessing tasks. The dataset was pro-

cessed to ensure compatibility with the ViLTB/32 model, with the following key steps:

- **Image Standardization**: Images were resized to 384×384 pixels using the Python Imaging Library (PIL), applying aspect-ratio-preserving padding to maintain consistent dimensions.
- **Data Transformation**: The `ViltProcessor` was utilized to tokenize questions and convert image-question pairs into tensors suitable for the ViLT model.
- **Answer Encoding**: Single-word answers were mapped to numerical class indices using a pre-built answer-to-index dictionary derived from the entire answer set, enabling supervised learning.
- **Output Structuring**: The `VQADataHandler` class returned a dictionary containing image tensors, tokenized question tensors, and encoded answer labels, formatted for seamless integration with the training pipeline.

8.5.2 LoRA-Based Fine-Tuning

The ViLTB/32 model was fine-tuned using Low-Rank Adaptation (LoRA), a technique designed to enhance parameter efficiency while maintaining the integrity of pre-trained weights. The training procedure was implemented using the HuggingFace `Trainer` API, with the following configuration:

- **LoRA Configuration**: LoRA adapters were integrated into the `query` and `value` projection layers with a rank of $r = 32$ and a LoRA alpha value of 64. This significantly reduced the number of trainable parameters while preserving the model's learning capacity.
- **Task Specification**: The fine-tuning task was defined as sequence classification (`SEQ_CLS`), enabling the model to address Visual Question Answering (VQA) as a multi-class classification problem.
- **Training Setup**: The model was trained over 10 epochs using mixed-precision (fp16) training to improve computational efficiency and reduce memory usage. All training was conducted within the Kaggle GPU environment.
- **Evaluation Output**: Upon completion of training, predictions were generated on the validation set. These predictions were saved along with the corresponding ground-truth answers in a CSV file for further evaluation and analysis.

8.6 Fine-tuning the `blip-vqa-base` Model

This subsection details the experiments conducted to fine-tune the `blip-vqa-base` model. Three versions of fine-tuning were performed, each targeting different configurations to optimize model performance. These experiments explored the effect of LoRA ranks (r) and epochs on model training.

8.6.1 Fine-tuning `blip-vqa-base` (Version-1)

- **Objective**: Baseline fine-tuning experiment using the `blip-vqa-base` model.
- **LoRA Configuration**:
  - **Rank (r)**: 8.
  - **Target Modules**: Applied LoRA adaptation to the `query` and `value` projection layers.
  - **Dropout**: LoRA dropout rate was set to 0.1.
  - **Bias**: Disabled (set to `none`).
- **Training Setup**:
  - **Number of Epochs**: 3 and 4 epochs were tested to observe training dynamics.
  - **Batch Size**: Set to 4 samples per device.
  - **Gradient Accumulation**: Accumulated gradients over 4 steps.
  - **Learning Rate**: 5×10^{-5}.
  - **Precision**: Half-precision (fp16) was enabled to optimize training speed and memory usage.
- **Outcome**: This version served as a baseline for subsequent experiments, providing initial insights into the performance of `blip-vqa-base` with LoRA rank $r = 8$.

8.6.2 Fine-tuning `blip-vqa-base` (Version-2)

- **Objective**: Explore the impact of increasing LoRA rank and batch size on model performance.
- **LoRA Configuration**:
  - **Rank (r)**: 16.
  - **Target Modules**: Similar to Version-1, applied to `query` and `value` layers.
  - **Dropout**: Maintained at 0.1.
  - **Bias**: Disabled (set to `none`).
- **Training Setup**:
  - **Number of Epochs**: 3 and 4 epochs were tested.
  - **Batch Size**: Increased to 8 samples per device for faster processing.
  - **Gradient Accumulation**: Not utilized due to the larger batch size.
  - **Learning Rate**: Adjusted to 4×10^{-5} for better stability with the larger batch size.
  - **Precision**: Half-precision (fp16) was enabled.
- **Outcome**: Increasing the LoRA rank to $r = 16$ showed promising results, with better training stability and minor improvements in accuracy compared to Version-1.

8.6.3 Fine-tuning `blip-vqa-base` (Version-3)

- **Objective**: Investigate the effect of higher LoRA rank and additional epochs on model performance.
- **LoRA Configuration**:
  - **Rank (r)**: 32.
  - **Target Modules**: LoRA was applied to `query` and `value` layers.
  - **Dropout**: Set to 0.1.
  - **Bias**: Disabled (set to `none`).
- **Training Setup**:
  - **Number of Epochs**: 3 and 4 epochs were tested.
  - **Batch Size**: Retained at 8 samples per device.
  - **Gradient Clipping**: Introduced gradient clipping with a maximum norm of 1.0 to improve training stability.
  - **Learning Rate**: 4×10^{-5}.
  - **Precision**: Half-precision (fp16) was used.
- **Outcome**: LoRA rank $r = 32$ provided the most stable training and the highest performance, showcasing the benefit of higher rank for model adaptation.

8.6.4 Comparative Analysis

- **LoRA Ranks**:
  - Rank $r = 8$ (Version-1): Served as the baseline, with consistent results across epochs.
  - Rank $r = 16$ (Version-2): Showed a slight improvement in stability and minor gains in accuracy.
  - Rank $r = 32$ (Version-3): Achieved the best performance, highlighting the advantage of higher rank in fine-tuning.
- **Epochs**:
  - Increasing epochs from 3 to 4 provided marginal benefits in some cases but also introduced risks of overfitting, especially at lower ranks.
  - With $r = 32$, the model benefited from additional training time, resulting in more stable and improved performance.
- **Batch Size and Gradient Clipping**:
  - Increasing the batch size from 4 to 8 reduced training time while maintaining performance consistency.
  - Gradient clipping (introduced in Version-3) helped stabilize training with the higher LoRA rank ($r = 32$).

```python
lora_config = LoraConfig(
    r=32,
    lora_alpha=32,
    target_modules=["query", "value"],
    lora_dropout=0.1,
    bias="none"
)
model = get_peft_model(model, lora_config)
print("LoRA applied to the model")
model = accelerator.prepare(model)
```

```
LoRA applied to the model
```

```python
training_args = TrainingArguments(
    output_dir="/kaggle/working/results",
    run_name="blip_vqa_lora_finetune",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=4,
    learning_rate=5e-5,
    weight_decay=0.01,
    logging_dir='/kaggle/working/logs',
    logging_steps=10,
    save_strategy="epoch",
    fp16=True,
    remove_unused_columns=False,
    report_to="none"
)

# Create Trainer instance with default data collator
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    data_collator=default_data_collator,
)
```

Figure 7: Demonstration of fine-tuning the BLIP model using LoRA with rank $r = 32$. The configuration applies LoRA to the `query` and `value` projection layers and sets up training using the Hugging Face `Trainer` API.

8.7 Challenges Faced During Fine-Tuning

Despite the overall success in fine-tuning the **BLIP-VQA-Base** model using LoRA, we encountered several practical and technical challenges that required troubleshooting and adaptation:

- **Limited GPU Availability and Compute Budget**: The fine-tuning was performed on a constrained GPU environment (free Kaggle notebooks), which imposed restrictions on total GPU hours and memory. We had to carefully balance batch sizes, sequence lengths, and model configuration to avoid out-of-memory errors and training interruptions.
- **Kaggle Runtime Instability**: Kaggle notebooks frequently timed out or became unresponsive during longer training sessions. This made it difficult to run multi-epoch training without interruption. We adapted by modularizing code into smaller cells when applicable and using shorter training loops for experimentation.
- **LoRA with BLIP Architecture**: While integrating LoRA into the BLIP model, we initially encountered errors related to undefined inputs like `inputs_embeds`.

The BLIP-VQA model did not expose all expected fields by default, requiring us to override or wrap the forward method to ensure compatibility with LoRA's expected input types.

- **Incompatibility with Quantization Modules**: We attempted to use the `bitsandbytes` library for parameter quantization to further reduce memory usage. However, on Kaggle, the latest module failed to install or be recognized for reasons we weren't entirely sure of, despite multiple attempts. This limited our ability to explore 4-bit or 8-bit quantization strategies.

- **Limited Experimentation Bandwidth**: Due to time and hardware constraints, we were limited in the number of hyperparameter tuning runs and ablation studies. As a result, certain design decisions (e.g., dropout rate, learning rate) were based on references from internet resources rather than exhaustive experimentation.

These challenges highlight the complexities involved in adapting large-scale vision-language models under constrained environments. Nonetheless, the experience reinforced the value of modularity, incremental testing, and efficient debugging in low-resource research settings.

9 Model Evaluation and Fine-Tuning Outcomes

This section presents a comprehensive evaluation of baseline model performance and the results of fine-tuning BLIP and ViLT models using Low-Rank Adaptation (LoRA) on a Visual Question Answering (VQA) task. The analysis includes baseline performance of pre-trained models (CLIP, BLIP, BLIP-2, ViLT), fine-tuning results for BLIP and ViLT with various LoRA configurations, and a comparative analysis of the outcomes, highlighting the efficacy of LoRA and optimal training strategies.

Baseline Model Performance

The baseline performance of four pre-trained models—CLIP, BLIP, BLIP-2, and ViLT—was assessed to establish a reference for the VQA task:

- **BLIP's Superiority**: BLIP achieved the highest test accuracy of 46.60%, significantly outperforming other models. Its BERTScore Precision (0.9143), ROUGE Score (0.4756), and BERT Cosine Similarity (0.7437) indicate strong semantic alignment with ground-truth answers. A low Levenshtein Distance of 2.73 reflects minimal edits needed to match predictions to correct answers.

- **CLIP's Limitations**: CLIP performed poorly, with a test accuracy of 2.6%, suggesting it is ill-suited for this VQA task.

- **Intermediate Performers**: BLIP-2 and ViLT recorded test accuracies of 24.90% and 26.11%, respectively, with BERT and ROUGE scores lower than BLIP's, indicating moderate alignment with ground truth.

- **Key Insight**: BLIP emerged as the most robust baseline, demonstrating superior predictive and semantic accuracy for VQA.

9.1 LoRA Fine-Tuning of BLIP

The BLIP model was fine-tuned using LoRA with varying ranks to optimize performance while maintaining parameter efficiency:

- **Optimal Configuration**: The highest performance was achieved with a LoRA rank of $r = 32$ at epochs 3 and 4, yielding a test accuracy of 62.5%. The ROUGE Score was marginally higher at epoch 3 (0.6375) than epoch 4 (0.6374), with BERT Cosine Similarity stable at 0.8163 (epoch 3) and 0.8162 (epoch 4), suggesting peak performance by epoch 3.
- **Lower Ranks**: Configurations with $r = 8$ and $r = 16$ yielded slightly lower results. For instance, $r = 16$ achieved a test accuracy of 62.4% and a ROUGE Score of 0.6367 at epoch 3.
- **Observation**: Higher LoRA ranks enhance performance by increasing trainable parameters, with optimal results achieved early in training, indicating efficient adaptation to the VQA dataset.

9.2 LoRA Fine-Tuning of ViLT

The ViLT model was fine-tuned with LoRA ($r = 32$) to improve its VQA capabilities:

- **Performance Gains**: The fine-tuned ViLT achieved a test accuracy of 54.46%, a substantial improvement over its baseline accuracy of 26.11%, demonstrating LoRA's effectiveness.
- **Semantic Metrics**: The model recorded a BERT Precision of 0.9898, BERT Recall of 0.9871, and BERT F1 Score of 0.9883, reflecting high precision and relevance in predictions. A macro F1 Score of 0.0841 suggests balanced but improvable performance across classes.
- **Conclusion**: LoRA significantly enhanced ViLT's VQA performance, particularly in semantic alignment, though its macro F1 indicates potential for further optimization.

9.3 Comparative Analysis and Insights

The fine-tuning results underscore LoRA's impact on improving VQA performance:

- **BLIP Improvements**: LoRA fine-tuning boosted BLIP's test accuracy from 46.60% to 62.5%, with enhanced ROUGE Scores and BERT Cosine Similarity, confirming improvements in syntactic and semantic accuracy.
- **ViLT Advancements**: ViLT's accuracy increased from 26.11% to 54.46% with LoRA ($r = 32$), highlighting the technique's ability to elevate even lower-performing baseline models.
- **Rank Influence**: Higher LoRA ranks ($r = 32$) consistently improved performance by allowing greater model adaptation, though diminishing returns between epochs 3 and 4 suggest early stopping to prevent overtraining.

- **Metric Strengths**: Strong BERT Precision, BERT F1, and low Levenshtein Distance across both models indicate accurate and semantically aligned predictions, suitable for practical VQA applications.

Configuration	Test Accuracy (%)	ROUGE Score	BERT Cosine Similarity
$r = 8$, Epoch 3	62.30	0.6358	0.8163
$r = 8$, Epoch 4	62.30	0.6362	0.8161
$r = 16$, Epoch 3	62.40	0.6367	0.8165
$r = 16$, Epoch 4	62.30	0.6360	0.8162
$r = 32$, Epoch 3	62.50	0.6375	0.8163
$r = 32$, Epoch 4	62.50	0.6374	0.8162

Table 1: Performance metrics for LoRA-based BLIP fine-tuning configurations on the VQA task.

Metric	Value
Test Accuracy	54.46%
Macro F1 Score	0.0841
Synonym Accuracy	55.79%
BERT Precision	0.9898
BERT Recall	0.9871
BERT F1 Score	0.9883

Table 2: Performance metrics for ViLT fine-tuned with LoRA ($r = 32$) on the VQA task.

Model	Test Accuracy (%)	BERTScore(Precision)	ROUGE Score	BERT Cosine Similarity	Levenshtein Distance
CLIP	2.6	0.1364	0.0299	0.2732	6.16
BLIP	46.60	0.9143	0.4756	0.7437	2.73
BLIP-2	24.90	0.7608	0.2587	0.5522	4.14
ViLT	26.11	0.7812	0.2734	0.5721	3.98

Table 3: Performance metrics for baseline models on the Visual Question Answering (VQA) task.

9.4 Conclusion

The evaluation and fine-tuning results demonstrate the power of LoRA as a parameter-efficient method for enhancing VQA model performance. BLIP stood out as the strongest baseline and fine-tuned model, achieving a test accuracy of 62.5% with LoRA ($r = 32$). ViLT, despite a weaker baseline, showed remarkable improvement, reaching 54.46% accuracy. These findings highlight the benefits of higher LoRA ranks and early stopping to optimize training efficiency, providing actionable insights for deploying VQA models in resource-constrained environments.

10 GitHub Repository

The full implementation of this project, including the dataset, code, and results, is available on GitHub at:

`https://github.com/rish710/VR_Project`