Multimodal Visual Question Answering with Amazon Berkeley Objects Dataset
REPORT

1) Introduction: Visual Question Answering (VQA)

- **Definition**: VQA is a multimodal AI task where the model answers natural language questions based on visual input (images).
- **Objective**: Requires understanding of both visual elements (objects, attributes, spatial relations) and textual semantics to generate accurate answers which is why it is called as multimodal.
- **Significance**:
    - Combines computer vision, natural language processing, and reasoning.
    - Real-world applications:
        - Assistive technology (for the visually impaired)
        - Smart retail and e-commerce (product Q&A)
        - Educational tools and robotics
    - **Example Use Case**: A user uploads an image and asks, *"What material is this shoe made of?"*—the system must analyze the image and provide a factually correct response.
- **Model Evolution**:
    - Early models: CNN + LSTM (basic image + text fusion)
    - Mid-stage: Attention-based models (e.g., Bottom-Up Attention)
    - Recent: Vision-language transformers (e.g., BLIP, CLIP, VilBERT)
- **Project Focus**:
    - Applying VQA to the Amazon Berkeley Objects (ABO) dataset.
    - Task complexity: Fine-grained questions about real-world products (e.g., material, type, size).
    - Requires precise, domain-specific visual reasoning beyond general datasets

2) Dataset: Amazon Berkeley Objects (ABO)

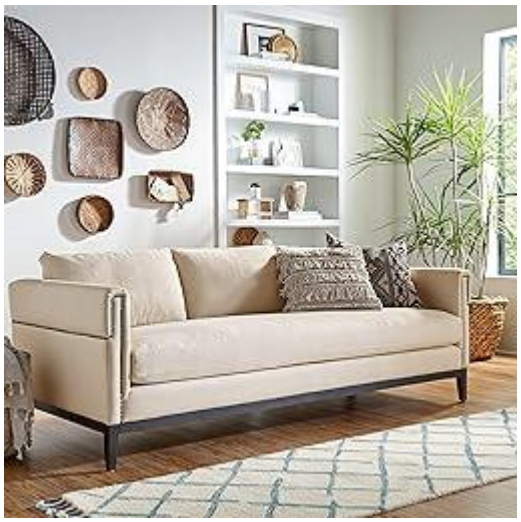
- The ABO dataset is a large-scale, object-centric dataset designed for vision tasks in real-world e-commerce scenarios.
- Contains over 150,000 unique consumer products, making it one of the richest sources of structured visual product data.
- It contains

  Visual Data:
    - Professional-grade product images (white background, isolated objects).
    - Clean, consistent visual quality—ideal for training vision models.
    - Each product includes multiple viewpoints (e.g., top, side, angled), enabling robust visual grounding and spatial reasoning.

  Textual Metadata:

    - Titles, product descriptions, and structured attributes (e.g., material, color, category, brand).
    - These serve as a rich source of natural language grounding for building VQA datasets.

- Example



image

Metadata for Product `B075X4QMX3` of `amazon.com`


```
    "language_tag": "en_US",
    "value": "Stone & Beam Westport Modern Nailhead Upho
  }
],
"3dmodel_id": "B075X4QMX3",
"brand": [
  {
    "language_tag": "en_US",
    "value": "Stone & Beam"
  }
],
"bullet_point": [
  {
    "language_tag": "en_US",
    "value": "Elevate your style with this head-turning
  },
  {
    "language_tag": "en_US",
    "value": "87\"W x 37.75''D x33''H"
  },
  {
    "language_tag": "en_US",
    "value": "60% cotton/40% linen fabric; metal nailhea
  },
  {
```


Metadata includes multilingual title, brand, model, year, product type, color, description, dimensions, weight, material, pattern, and style.

- Dataset used for our project
  - **abo-images-small.tar** — Downscaled (max 256 pixels) catalog images and metadata (3 Gb)
    - All the images were in the small folder and there was a csv with image details
      - Image id: - referring image
      - Height: - height of the image
      - Width: - width of the image
      - Path: - path to the image
  - **abo-listings.tar** — Product listings and metadata (83 Mb)
    - It's a Json file consisting about the fine details of every image in multiple languages

3) Data Preprocessing

- To prepare the ABO dataset for VQA tasks, multiple preprocessing steps were applied to ensure data quality, consistency, and alignment between images and metadata.
- Dropping Irrelevant Fields: -
  - Removed metadata fields that are not required for VQA like
    - spin_id
    - 3d_model_id
    - node_id
- Language Filtering: -
  - Dataset included metadata in multiple languages like English, French, Spain etc.
  - Since we were working only on English we dropped all the metadata in other languages.
- Checked for unique values of domain names and other attributes: -
  - We checked for value counts of each entry in metadata so that if it is unique for every item then we can drop or else can validate the whether the information is needed or not.
- Dropping the redundant data: -

    - There were some entries like height and normalized height so we decided to keep one and discarded the others and applied the same logic to other data.
- Image–Metadata Mapping Check: -
    - Finally checked that every meta data is mapped to the image and maps correctly.

4) Data Curation

- To build the VQA pairs we used the Gemini 2.0 Flash model via Google Generative AI Python SDK.
- Each generation batch consisted of:
    - 10 product images (PIL Image format)
    - 10 corresponding metadata blocks (text-based, describing the product)
- We called an API with 10 images in a batch and wanted to have context rich question answers so used the prompt

> *You are a question-answer pair generator specializing in vision-language alignment for e-commerce products.*
> *Given the image and detailed metadata of each product, generate *one concise and context-rich question* per product.*
> ***Important: ***
> *Do not focus solely on color. Ensure the questions cover different aspects such as product brand, shape, material, design, size, or functionality.*
> *The question must require visual input and be directly related to attributes visible in both the image and metadata.*
> *Avoid yes/no questions; answers must be a single, fact-based word.*
> *### Output format:*
> *product 1:*
> *Question: <Your question here>*
> *Answer: <Short descriptive answer>*
> *product 2:*
> *Question: <Your question here>*
> *Answer: <Short descriptive answer>*
> *...*
> *product 10:*
> *Question: <Your question here>*
> *Answer: <Short descriptive answer>*
> *Do not include any extra commentary*.

- A brief on method followed-
    - We defined a function named generate_vqa_batch to generate VQA for a batch of 10 images. We sent images,metadata and prompt to Gemini API and parsed the response we got to extract Q&A pairs.
    - To manage API usage efficiently, we defined:
    - A list named API_KEYS containing multiple API keys, used in a round-robin fashion to overcome rate limits and enable processing of more images.
    - progress.txt to store starting index of batch
    - curated_vqa_dataset6.csv to store output i.e, Question & Answer pairs along with image_id and batch_id.

- We iterated through available image ids in steps of 10(batch size)
- Called the generate_vqa_batch function with the current batch.
- For each Q&A result, We matched it to the correct image ID in the batch and stored it in Output csv.

- It was very challenging to curate the data because the token limit for one api call Input token limit 1,048,576 Output token limit 8,192. So we had to make sure that there are no unnecessary data lag specifying language tags.
- It took more than 10 days to curate the data for 398212 images
- Also, after curating the data, we found inconsistency like had multiple same questions for a single image and mostly all were colour oriented questions so we had to identify these entries and delete them.
- Finally concatenated with our images.csv which had the image id and path dimensions.
  File named: - complete_updated.csv
- Example of curated data: -

Q) What utensil is pictured?
A) spoon

Q) What animal is featured on the phone case?
A) pug

5) Baseline evaluation

- The goal of baseline evaluation is to assess the performance of off-the-shelf Visual Question Answering (VQA) models on the curated dataset without any fine-tuning.

5.1) Model

- We used the BLIP (Bootstrapped Language Image Pretraining) model — specifically the Salesforce/blip-vqa-base variant — for baseline inference.

Component	Details
Model Type	Vision-Language Pretrained Transformer (BLIP)
Architecture	Vision Encoder (ViT) + Text Decoder (GPT-style)
Total Parameters	~247 million (approximate, for blip-vqa-base)
Vision Encoder	ViT-B/16 (Vision Transformer, Base size, patch size 16x16)
Text Decoder	GPT2-style Transformer decoder (cross-attends to image features)
Input Image Size	224x224
Max Text Length	30 tokens (during question/answer encoding)

Max Generated Tokens	Up to 5 (for answer generation in our setup)
Decoder Strategy	Greedy decoding (default)
Tokenizer	BPE (Byte Pair Encoding) tokenizer
Pretraining	Bootstrapped language-image pretraining (captioning, retrieval, VQA tasks)
Framework	PyTorch + HuggingFace Transformers

5.2) Why BLIP for VQA?

- BLIP excels in image-text understanding and generative capabilities but is resource-intensive to fine-tune

5.3) Input for the model

- Each entry contains an image, a question generated via Gemini API, and a one-word ground-truth answer.

5.3) Samples

- Samples for Baseline: 5,000 curated samples, 20000 curated samples
- Small Subsample: A smaller subset of 500 samples was also used to explore how model performance varies with less data.

5.4) Inference and prediction

- For each sample, the following steps were performed:
    1. Image loaded from the ABO-small dataset.
    2. Paired with a corresponding generated question.
    3. Passed through the BLIP model to get a predicted answer (limited to 5 tokens, trimmed to 1-word for direct comparison).
    4. Prediction was compared to the ground-truth answer using:
        - Exact token match (for accuracy and token-level F1)
        - BERTScore (semantic similarity)

5.5) Metrics Used

- Accuracy: Proportion of exact one-word matches.
- Token-level F1 Score: Binary classification (correct/incorrect) per sample.
- BERTScore (F1): Semantic similarity of generated vs. reference answer using contextual embeddings.

5.6) Results

image	question	ground_truth	prediction	correct	token_f1	bertscore_f1
97/97a71b	What shape is the base?	octagonal	square	FALSE	0	0.837835
7e/7e3891	What building is featured in the image?	empire	empire	TRUE	1	1
7f/7f4b9c1	What word is written in all caps?	ingredients	all	FALSE	0	0.998885
cf/cf663b2	What organic material is printed on this phone case?	leaves	leaves	TRUE	1	1
dc/dce153	What material is this hardware plated with?	brass	metal	FALSE	0	0.999461
5e/5ec335	What is the primary material of the Rivet chair?	leather	leather	TRUE	1	1
e4/e40a8c	What is the filling material?	foam	cheese	FALSE	0	0.999566
46/466b4e	What is the color of the bow on the hat?	black	black	TRUE	1	1
57/579f2c	What brand is printed on the remote?	amazonbasics	panasonic	FALSE	0	0.822434
5c/5c118c	What is the shape of the drawer handle?	rectangular	circle	FALSE	0	0.999464
6c/6c10c2	What shape is the pasta shown on the package?	fusilli	square	FALSE	0	0.800159
ba/ba9d82	What shape is the pendant?	heart	heart	TRUE	1	1
fd/fdb3eb	What is the heel's general shape?	block	pointed	FALSE	0	0.999296
c0/c0d00b	What is the pattern density?	speckled	smooth	FALSE	0	0.799749
3e/3e99c8	What word is written inside the USDA logo?	organic	organic	TRUE	1	1
2d/2dd125	What is the primary color of these sporting gloves?	black	black	TRUE	1	1

Sample size	Accuracy(%)	Token F1 score	BERTScore
500	32.60	0.3260	0.9596
5000	34.96	0.3496	0.9592
20000	35.25	0.3525	0.9611

6) Fine-Tuning BLIP for Visual Question Answering with LoRA

- The goal of this project was to fine-tune a pre-trained Visual Question Answering (VQA) model, specifically BLIP (Bootstrapped Language-Image Pretraining), using a parameter-efficient fine-tuning strategy called LoRA (Low-Rank Adaptation). The idea was to make the model effective at answering visual questions with minimal computational cost by combining LoRA with model quantization.

6.1) **Parameter-Efficient Fine-Tuning Strategy**

6.1.1) **LoRA (Low-Rank Adaptation)**

- LoRA reduces the number of trainable parameters by injecting low-rank matrices into certain parts of the model (e.g., attention layers) while keeping the majority of the model frozen. This makes the training faster and more memory-efficient
- **Why LoRA?**
  - It updates fewer parameters. Reduced trainable parameters significantly (r=8,alpha=32)
  - It's memory-efficient and Enabled quick training with fewer resources.
  - Works well even on smaller datasets or limited compute setups.

6.1.2) **Quantization Strategy**

- **8-bit and 4-bit Precision: -**To further save memory and support training on resource-constrained environments, we applied 8-bit and 4-bit quantization using BitsAndBytesConfig. This allowed the model to be loaded in lower precision, significantly reducing memory usage and enabling training on consumer GPUs.

6.2) **Model Architecture**

6.2.1) Base Model

- We used the Salesforce/blip-vqa-base model. This architecture includes:
- A vision transformer that encodes image features
- A text transformer that understands the question

- Cross-attention layers that align image and text features

6.2.2) Adaptations Made

- Quantized the model to 8-bit using BitsAndBytesConfig
- Applied LoRA on:
  - Query and value projection matrices
  - Vision encoder attention layers

6.2.3) LoRA Configuration:

```
LoraConfig(r=8,
lora_alpha=32,
target_modules=["query", "value",
"vision_encoder.encoder.layer.*.attention.attention"],
lora_dropout=0.05,
bias="none"
)
```

6.3) **Dataset and Input Processing**

6.3.1) Input Format

- Each sample includes:
- An image
- A question (generated using Gemini API)
- A one-word answer (ground truth)

6.3.2) Preprocessing Steps

- Loaded images using PIL and converted them to RGB
- Added prompt formatting: e.g., "Answer in one word: "
- Tokenized using BLIP's processor with a max length of 32

6.3.3) Dataset Splits

- Baseline samples: 10,000 curated samples
- Split into: training, validation, and test sets
- Subsampling experiments: Performed training on 500, 1000, and 1500 samples to monitor metric behavior.

6.4) **Fine-Tuning Setup**

- Epochs: 1 (but ran also for 3 epoch in some experiments)
- Batch size: 8
- Learning rate: 1e-4
- Weight decay: 0.01
- Warmup steps: 500
- Precision: Mixed FP16 when using CUDA
- Hardware: CUDA GPU (preferred), fallback to CPU
- Training Framework: Hugging Face Trainer API
- After training, the model and processor were saved under the name: blip-vqa-lora-final

7) Evaluation Metrics

7.1) Chosen Metrics and Rationale

- Accuracy: Checks for exact one-word match (case-insensitive)
- BLEU-1: Measures unigram overlap (basic lexical similarity)
- BERTScore (F1): Captures semantic similarity using contextual embeddings

7.2) Why these Metrics

METRIC	STRENGTHS	LIMITATIONS
Accuracy	Strict, easy to interpret	Sensitive to synonyms/paraphrases
BLEU-1	Simple, good for short answers	Doesn't understand meaning
BERTScore	Captures meaning, context-aware	Computationally heavy, may over-penalize synonyms

7.3) Metric Implementation

- BLEU used smoothing (method1) to avoid zero scores
- Exact match was case-insensitive and trimmed
- BERTScore used the 'en' pre-trained model for embeddings

7.4) Loss trends over epochs

7.5) Time and memory usage stats

7.6) Accuracy, BLEU, and BERTScore over Validation set

8) Challenges

- Memory Constraints:
  - Solved using 8-bit quantization and small batch sizes
- Answer Variations:
  - Ground truth vs. predicted might differ semantically (e.g., "couch" vs. "sofa"
- Image Loading Issues:
  - Some images were corrupted or missing, handled with fallback logic
- Metric Limitations:
  - BLEU not ideal for very short or one-word answers
  - BERTScore sometimes overly strict on near-synonyms

9) Observations

- We observed increase in the accuracy after fine tuning the data and using quantization.
- Incorrect object recognition in complex images.
- Misunderstood ambiguous or poorly structured questions.
- Literal vs. Semantic Answers**:** E.g., "orange" vs. "fruit"
- Format Mismatch**:** Prediction was valid but slightly off (e.g., spacing or plural form)