Technical Report: BLIP-2 with LoRA Fine-Tuning for Visual Question Answering

1. Data Curation

Dataset Overview

For this project, we utilized the Amazon Berkeley Objects (ABO) dataset, which contains high-quality product images with associated metadata. The dataset presents an ideal foundation for visual question answering in the e-commerce domain due to its diverse product categories and detailed metadata.

VQA Pair Generation

Gemini API Integration We developed a custom VQA generation pipeline using Google's Gemini 1.5 Pro API. The pipeline (`abo_vqa_generator.py`) performs the following steps:

1. **Image Selection**: Samples product images from the ABO dataset
2. **Metadata Extraction**: Extracts relevant product attributes from ABO metadata:
    - Product name
    - Color information
    - Material details
    - Product descriptions
3. **Prompt Engineering**: We engineered specific prompts to guide Gemini in generating diverse, relevant questions:

```
prompt = f"""
Generate {num_questions} diverse visual question-answer pairs for this product image.

Product information:
- Name: {product_name}
- Color: {color_info}
- Material: {material_info}
- Description: {product_description}

Guidelines:
1. Each question should be answerable by looking at the image.
2. IMPORTANT: Answers MUST be exactly ONE WORD only. No phrases or multiple words allowed.
3. Include a variety of question types: {', '.join(QUESTION_TYPES)}
4. Include different difficulty levels: {', '.join(DIFFICULTY_LEVELS)}
5. Questions should focus on visual attributes like color, shape, material, pattern, count,
6. For numbers, use digits (e.g., "3" instead of "three").
"""
```

The prompt design ensures: - **Single-word answers**: Critical for standardized evaluation - **Question diversity**: Across predefined categories (color, shape, material, etc.) - **Difficulty variation**: Easy, medium, and hard questions - **Visual focus**: Questions answerable from the image alone

Data Quality Control The generated VQA pairs underwent multiple quality control steps: 1. **Format validation**: Ensuring strict adherence to single-word answers 2. **Redundancy filtering**: Removing duplicate question patterns 3. **Manual review**: Sample review of generated questions for quality assurance

The final dataset comprised 4,987 unique VQA pairs, stored in a CSV format with three columns: `image_name`, `question`, and `answer`.

2. Model Choices

Selection of BLIP-2

We selected BLIP-2 (Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models) as our base model for several reasons:

1. **Architecture advantages**: BLIP-2 employs a Querying Transformer (Q-Former) that efficiently connects a frozen vision encoder with a frozen language model, making it particularly well-suited for VQA tasks.
2. **Parameter efficiency**: BLIP-2 achieves strong performance while requiring fewer parameters than competing models. The architecture includes:
   - Frozen ViT as the vision encoder
   - Q-Former for cross-modal connections
   - Large Language Model (OPT or FLAN-T5) for text generation
3. **Strong zero-shot capabilities**: BLIP-2 demonstrates robust zero-shot performance on various vision-language tasks, providing a solid foundation for fine-tuning.

Alternative Models Considered

Before settling on BLIP-2, we evaluated several alternatives:

1. **BLIP (first generation)**:
   - Pros: Established performance on VQA benchmarks
   - Cons: Less parameter-efficient than BLIP-2, lacks the Q-Former architecture
2. **LLaVA**:
   - Pros: Strong performance with instruction tuning
   - Cons: Higher computational requirements, less targeted for concise answer generation
3. **VisualBERT**:
   - Pros: Simple architecture, well-established in literature
   - Cons: Lower performance ceiling compared to newer models

BLIP-2 Variant Selection

We primarily worked with the `Salesforce/blip2-opt-2.7b` variant, which offers a good balance between performance and computational requirements:

BLIP-2 Variant	Parameters	VRAM Required	Relative Performance
blip2-opt-2.7b	2.7B	~5GB	Good
blip2-opt-6.7b	6.7B	~12GB	Better
blip2-flan-t5-xl	-	~5GB	Comparable to 2.7B
blip2-flan-t5-xxl	-	~20GB	Best, but most resource-intensive

The 2.7B parameter model was selected for its accessibility on consumer-grade hardware while still delivering strong performance.

3. Fine-Tuning Approaches

LoRA (Low-Rank Adaptation)

Technique Overview LoRA is a parameter-efficient fine-tuning method that significantly reduces the number of trainable parameters. It works by:

1. Freezing the original model weights
2. Injecting trainable rank decomposition matrices into specific layers
3. Learning adaptations through these low-rank matrices

The key equation that describes LoRA is:

`h = Wx + \Delta Wx = Wx + BAx`

Where: - W is the original frozen weight matrix - B is a trainable matrix of shape (d $\times$ r) - A is a trainable matrix of shape (r $\times$ k) - r is the rank, a hyperparameter much smaller than min(d, k)

Implementation for BLIP-2 Our implementation applied LoRA to specific parts of the BLIP-2 model:

```
lora_config = LoraConfig(
    r=16,                       # Rank dimension
    lora_alpha=32,              # Scaling factor
    lora_dropout=0.05,          # Dropout probability
    target_modules=[            # Target specific layers
        "q_proj", "v_proj", "k_proj", "o_proj",  # Attention layers
        "gate_proj", "down_proj", "up_proj"      # MLP layers
    ],
    bias="none",
    task_type=TaskType.CAUSAL_LM
)
```

We targeted: - **Attention mechanisms**: Query, key, value, and output projections - **MLP blocks**: Gate projections, up projections, and down projections

These layers were selected based on prior research showing they are most impactful for adapting language models to new tasks.

Training Workflow The fine-tuning process consisted of several key components:

1. **Dataset Preprocessing**:
    - Efficient batching with consistent tensor shapes
    - Fixed max_length tokenization for all examples
    - Vision encoder feature caching to accelerate training
2. **Training Configuration**:
    - Batch size: 4
    - Learning rate: 1e-4 with linear decay
    - Training epochs: 3
    - Validation split: 20% of the dataset
    - Mixed precision (fp16) training
3. **Checkpointing**:
    - Models saved based on validation BERT score improvements
    - Final model and best checkpoints preserved

The entire fine-tuning process required approximately 2-3 hours on a single NVIDIA RTX 3090 GPU, demonstrating the efficiency of the LoRA approach.

4. Evaluation Metrics

BERT Score

BERT Score was selected as our primary evaluation metric due to its advantages for semantic understanding in VQA:

1. **Semantic similarity**: BERT Score leverages contextual embeddings to capture semantic similarity beyond exact word matching
2. **Robust to synonyms**: Can recognize correct answers expressed with different vocabulary
3. **Correlation with human judgments**: Better aligned with human assessment of answer quality than exact match metrics

Implementation We used the `bert-score` library to compute F1 scores between generated and ground truth answers:

```
P, R, F1 = bert_score(
    generated_answers,
    ground_truth_answers,
    lang="en",
```

```
    rescale_with_baseline=True
)
```

Results and Analysis

Model	BERT Score	Inference Time (images/sec)
BLIP-2 Baseline	0.66	9.8
BLIP-2 with LoRA	0.70	9.5

The LoRA fine-tuned model achieved a **6% relative improvement** in BERT Score with minimal impact on inference time.

Additional Performance Analysis

We conducted further analysis of model performance across different question types:

Question Type	Baseline Score	LoRA Fine-tuned Score	Improvement
Color	0.72	0.78	+0.06
Shape	0.69	0.74	+0.05
Material	0.64	0.70	+0.06
Count	0.58	0.65	+0.07
Pattern	0.62	0.69	+0.07

The most substantial improvements were observed in counting questions and pattern recognition, suggesting the fine-tuning particularly enhanced the model's ability to handle these more challenging visual reasoning tasks.

5. Additional Contributions and Novelty

Efficient Preprocessing Pipeline

We developed a specialized preprocessing pipeline for BLIP-2 that significantly improves training efficiency:

```python
class VQADatasetPreprocessor:
    def __init__(self, csv_path, image_dir, processor, model, device, batch_size=8):
        # Initialize components

    def process_data(self):
        """Generate text features for all images and questions in the dataset"""
        # Batch processing of vision encoder features
        # Store processed features
```

This preprocessing approach: 1. **Reduces redundant computation**: Processes vision features only once 2. **Minimizes memory footprint**: Uses fixed tensor shapes and efficient padding 3. **Accelerates training**: Achieves up to 3x speedup compared to on-the-fly processing

Adaptive LoRA for Multimodal Models

Our implementation extends LoRA to effectively work with multimodal architectures:

1. **Selective layer targeting**: We identified and targeted only the most relevant layers in BLIP-2's complex architecture
2. **Preservation of cross-modal alignment**: Fine-tuning approach maintains the cross-modal connections established during pre-training
3. **Quantitative analysis of layer impact**: Conducted ablation studies to determine optimal LoRA configuration

Single-Word Answer Optimization

We specifically optimized the model for generating concise, single-word answers:

1. **Prompt engineering**: Crafted prompts that direct the model toward concise answers
2. **Generation constraints**: Limited token generation (max_new_tokens=4)
3. **Post-processing**: Implemented extraction of single-word answers from model outputs

```
# Decode the response and extract the first word of the answer
full = processor.decode(generated_ids[0], skip_special_tokens=True)
answer = full.split("Answer in one word:")[-1].strip().split()[0]
```

This approach results in answers that are both more accurate and more aligned with the desired output format, enhancing both human readability and automated evaluation.

Conclusion

Our work demonstrates that LoRA fine-tuning is an effective approach for adapting BLIP-2 to specialized VQA tasks, achieving meaningful performance improvements with minimal computational overhead. The combination of efficient data generation, targeted model adaptation, and optimized inference creates a practical solution for e-commerce visual question answering that can run on consumer hardware.