Context-Aware Image Captioning with BLIP-2: Generating Rich Captions from News Photos

TECHNISCHE UNIVERSITÄT DARMSTADT

Ali Shariati Najafabadi
September 1, 2025

Abstract

This report presents the development and implementation of a context-aware image captioning system using BLIP-2, fine-tuned on the GoodNews dataset. Unlike traditional captioning systems that generate literal descriptions like "A person standing in front of a building", our approach integrates contextual signals including temporal, geographical, and social information to produce richer, more meaningful captions such as "Reporter covering the climate summit in Paris, December 2024". Through distributed multi-GPU training on 4× RTX 2080 Ti GPUs with significant memory constraints, we successfully fine-tuned only 183,814,144 parameters (4.91% of total) including the Q-Former, language model head, and last decoder layer. The project demonstrates effective context integration within hardware limitations and provides a foundation for more sophisticated multimodal understanding systems.

Contents

1 Introduction

Image captioning has evolved significantly from basic object detection to sophisticated scene understanding. Traditional captioning systems focus on literal visual content description, producing captions like "A man in a suit speaking at a podium". However, such descriptions lack the contextual richness that humans naturally incorporate when describing images. Context-aware captioning addresses this limitation by integrating additional information sources to generate more meaningful and informative descriptions.

The motivation for context-aware captioning stems from real-world applications where understanding the broader context is crucial. News organizations, social media platforms, and digital archives require captions that capture not just what is visible in an image, but also the circumstances, people, locations, and events that give the image meaning. For instance, a photo of a political figure at a podium becomes significantly more informative when captioned as "President Biden addressing climate change at COP28 summit in Dubai, December 2023".

This project implements a context-aware image captioning system using BLIP-2, a state-of-the-art vision-language model, fine-tuned on the GoodNews dataset. The GoodNews dataset provides news images with rich metadata including article context, named entities, locations, and temporal information, making it ideal for training models to incorporate contextual signals.

2 Related Work and Background

2.1 Vision-Language Models

The field of vision-language understanding has witnessed remarkable progress with the development of large-scale multimodal models. CLIP (Contrastive Language-Image Pre-training) pioneered the use of contrastive learning for aligning visual and textual representations using 400 million image-text pairs. This approach demonstrated strong zero-shot capabilities and robust cross-modal understanding.

Building on CLIP's success, BLIP (Bootstrapping Language-Image Pre-training) introduced a unified vision-language understanding and generation framework. BLIP employs a bootstrap approach with three main objectives: image-text contrastive learning, image-text matching, and image-conditioned language modeling. This multi-objective training enables BLIP to excel in both understanding and generation tasks.

2.2 BLIP-2 Architecture

BLIP-2 represents a significant advancement in vision-language models through its two-stage bootstrapped pre-training approach. The architecture consists of three main components:

- **Frozen Vision Encoder**: A pre-trained image encoder (ViT-L/14) that processes visual input without further training
- **Querying Transformer (Q-Former)**: A trainable module that bridges vision and language modalities
- **Frozen Language Model**: A large language model (OPT-2.7B) for text generation

The Q-Former is the key innovation in BLIP-2, serving as a trainable interface between the frozen vision encoder and language model. It uses learnable query embeddings to extract visual features most relevant for language generation, enabling efficient training without requiring updates to the large pre-trained components.

2.3 Context Integration in Image Captioning

Traditional image captioning approaches focus primarily on visual content analysis. Recent research has explored incorporating external knowledge and contextual information to generate more informative captions. Context can include:

- **Temporal Context**: Time and date information
- **Geographical Context**: Location and place names
- **Social Context**: People, organizations, and relationships
- **Event Context**: Specific events or occasions

The GoodNews dataset enables training models to integrate such contextual signals by providing news images with rich metadata extracted from associated articles and captions.

3 Dataset and Methodology

3.1 GoodNews Dataset

The GoodNews dataset consists of news images from The New York Times with corresponding articles, captions, and rich metadata. The dataset provides:

- News images with original high-resolution URLs
- Human-written captions with contextual information
- Article text and metadata
- Named entity annotations (people, places, organizations)
- Temporal information extracted from URLs and content
- Geographic location data

The dataset was processed through a comprehensive five-stage pipeline to prepare it for training.

3.2 Data Processing Pipeline

3.2.1 Stage 1: URL Enhancement (`0_url_to_superjumbo.py`)

The original image URLs were converted to high-resolution "superJumbo" variants to ensure sufficient image quality for training. This involved systematic URL manipulation to access the highest available resolution images from The New York Times image servers.

3.2.2 Stage 2: Image Download (`1_get_images.py`)

A multi-threaded download system was implemented to efficiently retrieve images from the enhanced URLs. The system included:

- Parallel processing with configurable thread count
- Error handling and retry mechanisms
- Progress tracking for robust data collection
- Network error handling for failed downloads

3.2.3 Stage 3: Context Extraction (`2_create_context_dataset.py`)

Using spaCy's natural language processing capabilities, contextual information was extracted from captions and metadata:

- **Geographic Entities**: Locations, cities, countries (GPE, LOC)
- **Temporal Information**: Dates in YYYY-MM-DD format extracted from URLs and text
- **Person Names**: Individual people and public figures (PERSON entities)
- **Organizations**: Companies, institutions, sports teams (ORG entities)
- **Events**: Major events like Olympics, World Cup, summits
- **News Sections**: Extracted from URL structure (world, sports, business, etc.)

Context strings were formatted as: `[category | location | date | PERSON=name | EVENT=event]`

3.2.4 Stage 4: Image Preprocessing (`3_resize_images.py`)

All images were resized to 224×224 pixels to match BLIP-2's input requirements. The preprocessing included:

- Center cropping to square aspect ratio
- RGB conversion for RGBA/palette images
- JPEG quality optimization
- File size optimization for training efficiency
- Path updates in JSON files

3.2.5 Stage 5: Filesystem Organization (`4_organize_folders.py`)

Images were organized into hash-based subdirectories (00-ff) to improve filesystem performance and enable efficient data loading during training. This organization is crucial for handling large datasets with thousands of images.

3.3 Training Methodology

3.3.1 Model Architecture and Fine-tuning Strategy

Given the substantial size of BLIP-2 (3,744,761,856 total parameters) and hardware constraints, we adopted a highly selective fine-tuning approach targeting specific model components:

- **Q-Former**: Trainable transformer module bridging vision and language
- **Language Model Head**: Output projection layers
- **OPT Decoder**: Last 1 decoder layer only (layer 31 of 32)

This conservative strategy enabled training 183,814,144 parameters (4.91% of the total 3,744,761,856 parameters) while maintaining the powerful pre-trained representations in the vision encoder and most language model layers.

3.3.2 Distributed Training Implementation

The most significant technical challenge was implementing efficient multi-GPU training within severe hardware constraints. Our `safe_multi_gpu_training.py` script addressed several critical aspects:

Memory Optimization Techniques:

- **Mixed Precision Training**: FP16 with automatic mixed precision to reduce memory usage by approximately 50%
- **Gradient Checkpointing**: Trading computation for memory by recomputing activations during backward pass
- **8-bit Optimization**: Using bitsandbytes AdamW8bit optimizer for additional memory savings

- **Gradient Accumulation**: 64 steps to simulate large batch sizes without memory overflow

Distributed Training Configuration:

Listing 1: Multi-GPU Training Setup (run.sh)

```
export CUDA_VISIBLE_DEVICES=0,1,2,3
export MASTER_ADDR=localhost
export MASTER_PORT=29500

torchrun \
    --nproc_per_node=4 \
    --master_addr=localhost \
    --master_port=29500 \
    safe_multi_gpu_training.py
```

Hardware Constraints and Critical Solutions:

The primary challenge was working with 4× NVIDIA GeForce RTX 2080 Ti GPUs, each with 11GB VRAM. This configuration provided 44GB total GPU memory, which proved insufficient for training multiple decoder layers simultaneously. Initially, attempts to train more than one decoder layer resulted in out-of-memory errors.

The solution involved implementing a conservative parameter selection strategy:

Listing 2: Critical Parameter Selection Logic

```
def setup_trainable_params_efficient(model, rank, output_dir):
    last_n_layers = 1  # CRITICAL: Conservative setting for 11GB GPUs

    for name, param in model.named_parameters():
        should_train = False

        if "qformer" in name.lower():
            should_train = True
        elif "lm_head" in name:
            should_train = True  # Language model head
        elif "language_model.model.decoder.layers." in name:
            # Extract layer number and check if in last N layers
            layer_num = int(layer_str)
            if layer_num >= (num_decoder_layers - last_n_layers):
                should_train = True  # ONLY LAST 1 LAYER

        param.requires_grad = should_train
```

This approach successfully enabled training while staying within memory constraints. For future work with larger GPUs (24GB or more), the `last_n_layers` parameter can be increased to train additional decoder layers and potentially improve performance.

4 Experimental Results

4.1 Training Progress and Convergence

Training was successfully completed over 12 epochs using the distributed setup on 4× RTX 2080 Ti GPUs. The loss progression demonstrated effective learning despite limited trainable parameters:

The training exhibited strong initial convergence with loss dropping from over 10.0 to 1.34 in the second epoch, followed by steady optimization. The best validation loss of 0.4836 was achieved at epoch 8, indicating effective learning without significant overfitting.

4.2 Model Configuration and Scaling

4.3 Learning Rate Schedule

The training employed a sophisticated cosine annealing schedule with warmup:

- **Warmup Phase**: Linear increase from 0 to 3×10^{-5} over 120 steps
- **Peak Learning Rate**: 3×10^{-5} at step 120
- **Cosine Decay**: Gradual decrease to 8×10^{-6} over remaining 480 steps (steps 120-600)

This schedule provided stable training while allowing sufficient exploration during the early phases and fine-grained optimization in later epochs.

Table 1: Complete Training Loss Progression (August 29-30, 2025)

Epoch	Average Loss	Validation Loss	Notes
1	10.7450	-	Major convergence from initial loss
2	1.3448	-	Dramatic improvement
3	0.5968	-	Continued steep decrease
4	0.5513	-	Stabilizing
5	0.5362	-	Steady progress
6	0.5212	-	Consistent optimization
7	0.5136	-	Gradual improvement
8	0.5101	**0.4836**	**Best validation achieved**
9	0.5034	-	Continued learning
10	0.5054	-	Near convergence
11	0.4978	-	Final optimization
12	0.5019	-	Training complete

Table 2: Model Configuration and Hardware Scaling Guidelines

GPU Configuration	Model	Total Trainable	Layers Trained	Expected Performance
4× 11GB RTX 2080 Ti	**blip2-opt-2.7b**	**183.8M (4.91%)**	**last_n_layers = 1**	**Current setup**
4× 16GB	blip2-opt-2.7b	400M (10.7%)	last_n_layers = 4	Better accuracy
4× 24GB RTX 4090	blip2-flan-t5-xl	800M (15%)	last_n_layers = 8	Best performance
8× 24GB	blip2-flan-t5-xl	Full model	All layers	Professional grade

4.4 Qualitative Results

The trained model demonstrated strong context-aware caption generation capabilities across various domains:

Figure 1: Sports Context Example - Bill Belichick

Sports Context Example:

- **Original**: “Patriots Coach Bill Belichick on Tuesday. ”I think Belichick is better at keeping pressure on the passer than passing a physics test,” a cosmologist said.”
- **Generated**: “Context: [sports 2015-11-02 PERSON Bill Belichick] Caption: Bill Belichick, the coach of the New England Patriots, at a news conference in Foxborough, on Monday.”
- **Context**: `[sports | Belichick | 2015-01-28 | PERSON=Bill Belichick]`

Political Context Example:

- **Original**: “President Recep Tayyip Erdogan voted on Sunday at a polling station in Istanbul. His party gained a decisive victory.”
- **Generated**: “Context: [world Turkey 2015-11-13 PERSON Erdogan] Caption: President Recep Tayyip Erdogan of Turkey voting in Istanbul on Sunday. Mr. Erdogans said at a news conference after casting his own vote.”

Figure 2: Political Context Example - President Erdogan

Figure 3: Legal/Political Context Example - Supreme Court

- **Context**: `[world | Istanbul | 2015-11-03 | PERSON=Recep Tayyip Erdogan]`

Legal/Political Context Example:

- **Original**: “Chief Justice John G. Roberts Jr., right, with Associate Justice Anthony M. Kennedy. Justice Roberts is known as a premier appeals brief craftsman.”
- **Generated**: “Context: [us 2017-01-18 PERSON the Supreme Court] Caption: Chief Justice John G. Roberts Jr., left, and Justice Anthony M. Kennedy at the Supreme Court in Washington on Tuesday.”
- **Context**: `[business | 2016-10-01 | PERSON=John G. Roberts Jr.]`

Figure 4: Legal Context Example - Bill Cosby

Legal Context Example:

- **Original**: “Bill Cosby arriving Wednesday at the Montgomery County Courthouse.”
- **Generated**: “Context: [nyregion New York 2015-03-17 PERSON Cosby] Caption: Bill Cosby at a court hearing in Philadelphia last month. He has pleaded not guilty to charges that he sexually assaulted Andrea Constand, a former Temple University employee, in 2004.”
- **Context**: `[arts | 2018-04-25 | PERSON=the Montgomery County Courthouse]`

International News Context Example:

Figure 5: International News Context Example - Barcelona Attack

- **Original**: “Mourners on Friday on Las Ramblas, the site of a deadly terrorist attack in Barcelona, Spain.”
- **Generated**: “Context: [world Barcelona 2017-09-18 PERSON Marjory Stoneman Douglas High School] Caption: A memorial for the victims of the attack in Barcelona, Spain, on Wednesday.”
- **Context**: `[world | Las Ramblas | 2017-08-19]`

These examples demonstrate the model’s ability to:

- Integrate temporal information (dates and time references)
- Incorporate geographic context (countries and cities)
- Recognize and include named entities (people and organizations)
- Understand event context (elections, sports competitions, legal proceedings)
- Generate coherent, contextually appropriate captions

4.5 Evaluation Metrics

The model was evaluated using standard image captioning metrics implemented in `evaluate_results.py`:

- **BLEU-4**: Measures n-gram overlap with reference captions
- **ROUGE-L**: Evaluates longest common subsequence similarity
- **Length Statistics**: Compares generated and reference caption lengths

The evaluation framework provides both quantitative metrics and qualitative examples to assess model performance comprehensively.

5 Technical Challenges and Solutions

5.1 Critical Multi-GPU Training Challenges

5.1.1 Memory Constraints - The Primary Bottleneck

The most critical challenge was GPU memory limitation. With 4× RTX 2080 Ti GPUs providing 11GB each, the total 44GB was insufficient for training multiple decoder layers simultaneously. Initial attempts to train with `last_n_layers = 4` consistently resulted in CUDA out-of-memory errors.

Solution: We implemented a highly conservative approach with `last_n_layers = 1`, training only the final decoder layer along with the Q-Former and language model head. This reduced the trainable parameter count to a manageable 183.8M parameters while maintaining the model’s core capabilities.

5.1.2 Parameter Selection Strategy

The implementation required careful parameter selection to maximize training effectiveness within memory constraints:

Listing 3: Conservative Parameter Selection Strategy

```
# Critical configuration for 11GB GPUs
last_n_layers = 1  # Maximum possible with current hardware

# Trainable components breakdown:
# 1. Q-Former
# 2. Language Model Head: Output projection layers
# 3. OPT Decoder: ONLY layer 31 of 32 (last layer)
# Total: 183,814,144 parameters (4.91% of model)
```

5.2 Memory Optimization Strategies

Several memory optimization techniques were essential for successful training:

- **Gradient Checkpointing**: Reduces memory usage at the cost of computation by recomputing activations
- **Mixed Precision (FP16)**: Automatic mixed precision training with loss scaling
- **8-bit Optimization**: bitsandbytes AdamW8bit optimizer for additional memory savings
- **Large Gradient Accumulation**: 64 steps to simulate large effective batch sizes without memory overflow

5.3 Scalability Considerations

The implemented solution provides clear scaling paths for improved hardware:

Table 3: Detailed Hardware Scaling Guidelines

GPU Configuration	Memory	Recommended Config	Expected Improvements
4× 11GB RTX 2080 Ti	**44GB**	**last_n_layers = 1**	**Current limitations**
4× 16GB	64GB	last_n_layers = 4	Better context integration
4× 24GB RTX 4090	96GB	last_n_layers = 8	Significant performance boost
8× 24GB	192GB	Full fine-tuning	Research-grade results

6 Discussion and Future Work

6.1 Achievements Within Constraints

This project successfully demonstrates the feasibility of context-aware image captioning using BLIP-2 with severely limited computational resources. Key achievements include:

- Successful implementation of distributed multi-GPU training with extreme memory constraints
- Development of a robust 5-stage data processing pipeline for the GoodNews dataset
- Creation of a context-aware captioning system that integrates temporal, geographical, and social information
- Implementation of memory-efficient training strategies enabling work with limited GPU resources
- Achievement of meaningful context integration with only 4.91% of model parameters trainable

6.2 Critical Limitations

Several significant limitations constrained the current implementation:

6.2.1 Hardware Constraints

The 11GB GPU memory limitation severely restricted the number of trainable parameters. With only 4.91% of the model parameters being trainable, the fine-tuning was more limited than optimal. This constraint particularly affected the language model's ability to fully adapt to the context-aware generation task.

6.2.2 Training Strategy Limitations

The conservative training approach, while necessary for the hardware constraints, limited the model's ability to:

- Learn deeper contextual representations
- Adapt more language model layers to the specific task
- Achieve potentially better performance with more trainable parameters

6.3 Future Improvements

6.3.1 Enhanced Hardware Utilization

With access to larger GPUs (24GB or more), the training could be significantly improved by:

- Increasing `last_n_layers` to train more decoder layers (4-8 layers)
- Using the larger BLIP-2 FlanT5-XL model for better performance
- Implementing fuller fine-tuning of the language model
- Training with larger effective batch sizes

6.3.2 Advanced Context Integration

Future work could explore more sophisticated context integration approaches:

- Cross-modal attention mechanisms that explicitly model visual-contextual relationships
- Hierarchical context representation with different levels of detail
- Dynamic context weighting based on image content
- Integration of external knowledge bases for richer context

7 Implementation Details

7.1 Project Structure and Scripts

The project follows a systematic structure with specialized scripts for each processing stage:

- **Dataset Preparation**: 5-stage pipeline (`0_url_to_superjumbo.py` through `4_organize_folders.py`)
- **Training**: `safe_multi_gpu_training.py` with conservative memory management
- **Inference**: `best_caption.py` for caption generation
- **Evaluation**: `evaluate_results.py` for comprehensive assessment

7.2 Memory Management Strategy

The training implementation includes sophisticated memory management:

Listing 4: Memory Optimization Configuration

```
# Key optimization settings
mixed_precision = True  # FP16 training
gradient_checkpointing = True
gradient_accumulation_steps = 64
optimizer = "adamw_8bit"  # bitsandbytes optimization

# Conservative parameter selection
trainable_params = {
    "qformer": "all_layers",
    "lm_head": "output_projection",
    "decoder_layers": "last_1_only"  # Critical constraint
}
```

8 Conclusion

This project successfully demonstrates the implementation of a context-aware image captioning system using BLIP-2, achieving meaningful context integration despite severe hardware constraints. The work provides valuable insights into distributed training of large multimodal models with limited resources and establishes a foundation for future research in context-aware vision-language understanding.