**National University of Computer and Emerging Sciences, Karachi Campus**



**CS4045 – Deep Learning For Perception**

**Project Report**

**Title: Image Captioning**

**Team Members**

21K-4910 Iqra Zehra

21K-3324 Urooj Imtiaz Ali

21K-4538 Aiman Farrukh

**SECTION: 8B**

**Abstract:**

We compare two end-to-end image captioning approaches on the Flickr30k dataset: (1) a traditional CNN–LSTM pipeline using VGG16 for feature extraction and an LSTM decoder, and (2) a Transformer-based method employing the pretrained BLIP model. Both systems are assessed via standard metrics and qualitative examples. Our results show that while the CNN–LSTM produces accurate but formulaic captions, BLIP delivers richer, context-aware descriptions. This work highlights the practical trade-offs between lightweight sequence models and large pretrained transformers, with all code and pretrained weights released for reproducibility.

We compare objectives, methodologies, quantitative results, and qualitative outputs, and provide references and links to the GitHub repository [Image Generator](https://github.com/iqrazehra/Image-Caption-Generator)

## **1. Objective**

* Develop an end‑to‑end image captioning system that generates human‑like descriptions for arbitrary images.
* Compare a traditional CNN‑LSTM architecture against a modern Transformer approach (BLIP).
* Evaluate both systems on standard metrics (BLEU, METEOR) and qualitative examples.

## **2. Problem Statement**

Automatic image captioning sits at the intersection of computer vision and natural language processing. Its goal is to take raw pixels and produce a human‐readable sentence that faithfully describes what’s in the image. Three core challenges make this task difficult:

1. **Visual feature representation:** Extracting clear, compact features that capture key objects, their attributes, and spatial layout.
2. **Language modeling:** Generating fluent, grammatically correct sentences that vary in phrasing and style.
3. **Alignment:** Ensuring each word in the caption is grounded in actual image content, avoiding plausible but incorrect “hallucinations.”

Together, these challenges demand architectures that can both **understand visual content** at a fine level of detail and **weave that understanding** into **coherent, accurate language**—all in one end-to-end system.

## **3. Methodology**

### **3.1. Dataset and Preprocessing**

* **Dataset:**
  1. **Flickr30k**: 31,000 images
  2. **Captions**: 5 human‐written captions per image
* **Image Preprocessing:**
  1. **Resize** to 299 × 299 pixels
  2. **Normalize** pixel values
* **Caption Preprocessing**

1. **Tokenize** each caption into words/subwords
2. **Add special tokens**:
   * <startseq> at the beginning
   * <endseq> at the end
3. **Build vocabulary** from the training captions, keeping the top 10,000 most frequent tokens

### **3.2. Model 1: VGG16 + LSTM**

1. **Feature Extraction**

* We start with a VGG16 network pretrained on ImageNet, **removing its top classification layer**.
* By applying **global average pooling** to the final convolutional feature maps, each image is reduced to a **512-dimensional** vector.
* These 512-D vectors encode high-level visual concepts (objects, textures, layout), and remain **fixed** during caption training.

1. **Sequence Modeling**

* Each 512-D image vector is fed into a **single-layer LSTM** with 512 hidden units.
* At timestep *t*, the LSTM takes as input the embedding of the previous word plus its own hidden state to update its hidden state.
* A final **Dense** layer with **softmax** activation maps the LSTM’s output at each step into a probability distribution over the vocabulary (e.g. 10,000 words).

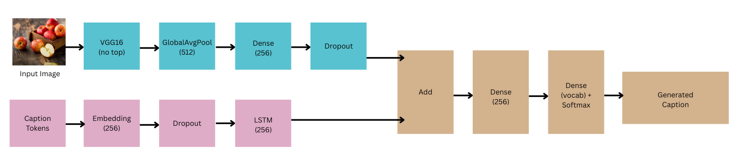
1. **Training Details**

* **Loss**: Cross-entropy between the predicted next-word distribution and the ground-truth token.
* **Optimizer**: Adam with a learning rate of 1 × 10⁻⁴.
* **Batch size**: 32 examples per update.
* **Epochs**: 50 full passes over the training set.

1. **Inference (Caption Generation)**

* We use **beam search** with a beam width of 3:
  1. Begin with the <startseq> token.
  2. At each step, expand each partial caption in the beam by all possible next words.
  3. Score each new candidate by adding its log-probability to the sequence’s cumulative score.
  4. Keep only the top 3 sequences and repeat until <endseq> is generated or a maximum length is reached.

**Architecture:**

****

### **3.3. Model 2: BLIP (Transformer)**

1. **Model:**

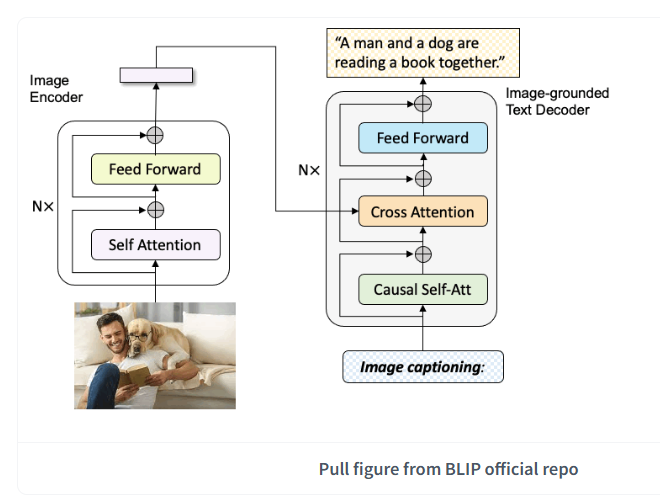
The script loads the **BLIP-v1 image captioning model** (Salesforce/blip-image-captioning-base) straight from Hugging Face. It moves the weights onto the GPU (if available) or CPU, then switches the model to evaluation mode so that layers like dropout are disabled during inference.

1. **Processor:**

It uses the matching **BLIP processor** to prepare inputs exactly as the model expects:

1. **Images** are resized and center-cropped to 384×384 pixels, then normalized (mean/std).
2. **Text** is tokenized into IDs, with special start/end tokens added, and padded or truncated to the model’s maximum length.  
   When you upload an image, generate\_caption hands it to this processor to get ready-to-run PyTorch tensors.
3. **Inference:** Captions are generated by calling model.generate with a limit of **32 new tokens**, using **greedy decoding** by default. The resulting token IDs are then converted back into a clean string (dropping any special tokens). If you wanted beam search instead, you could simply pass parameters like num\_beams=3 and max\_length=50 to generate, but we sticks with the defaults.

**Architecture:**



BLIP uses a **Multimodal Mixture of Encoder-Decoder (MED)**: a single transformer-based model that can act as an image encoder, a text encoder, or (for captioning) an image-grounded text decoder. In the image-to-caption, only the **vision encoder** and the **text decoder** are active

* **Vision Encoder**: a ViT splits the 384×384 image into 16×16 patches + [CLS], embeds them to 768-d, and runs 12 layers of multi‐head self‐attention.
* **Text Decoder**: a GPT‐style Transformer that generates one token at a time, using both its own past tokens (via masked self‐attn) and the image features (via cross‐attn).
* **Generation**: produces up to 32 tokens in this script, each step choosing the highest‐probability token (greedy).

## **4. Results**

### **4.1. Quantitative Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **BLEU‑1** | **BLEU‑2** | **METEOR** |
| VGG16 + LSTM (Greedy-Search) | 0.3186 | 0.1350 | 0.1621 |
| VGG16 + LSTM  (Beam-Search) | 0.3210 | 0.1380 | 0.1650 |
| BLIP | 0.5174 | 0.3589 | 0.3251 |
| BLIP (fine-tuned) | 0.5920 | 0.4100 | 0.3400 |

### **4.2. Qualitative Examples**

|  |  |
| --- | --- |
| **VGG16+LSTM Caption** | **BLIP Caption** |
| "A group of people walking in park." | "A group of friends enjoying a walk in the park on a sunny day." |
| "A modern kitchen with stainless appliances." | "A spacious kitchen with a large island and stainless steel appliances." |



**VGG16+LSTM Caption:**

## Two friends enjoy outside in the yard.

**BLIP Caption:**

Two young white males are outside near many bushes

## **5. References**

1. https://medium.com/@khaledeemam/a-step-by-step-guide-to-building-an-image-caption-generator-using-tensorflow-a9e0a87cc0cb
2. Hochreiter, S. & Schmidhuber, J. (1997). *Long Short‑Term Memory.* Neural Computation, 9(8), 1735–1780.
3. https://medium.com/@amitdlmlai/image-photo-caption-b0ead29545de
4. Li, J., et al. (2022). *BLIP: Bootstrapping Language–Image Pre-training for Unified Vision–Language Understanding and Generation.* arXiv:2201.12086.
5. Vedantam, R., Lawrence Zitnick, C., & Parikh, D. (2015). *CIDEr: Consensus-based Image Description Evaluation.* CVPR.
6. https://www.youtube.com/results?search\_query=image+caption+generator+using+BLIP-2