**Introduction to the Problem**

Image captioning is a challenging task in the field of computer vision and natural language processing, requiring a model to **generate meaningful descriptions** of images. This problem combines **feature extraction** from images (using deep convolutional neural networks) and **sequence generation** (using recurrent neural networks or transformers).

The primary goal of this project is to **develop an automated image captioning model** that can accurately describe images in natural language. This has a wide range of real-world applications, including:

* **Assisting visually impaired individuals** by describing images.
* **Enhancing content-based image retrieval** in search engines.
* **Improving human-computer interaction** through intelligent image understanding.

Despite advancements in deep learning, image captioning remains a **difficult task** due to:

1. The complexity of **mapping visual features to meaningful text**.
2. The **ambiguity** of multiple possible correct captions.
3. The challenge of **long-term dependencies** in sentence generation.

This project explores the **encoder-decoder architecture** with a CNN-based encoder (ResNet) for feature extraction and an RNN-based decoder (LSTM) for generating captions. The model is evaluated using the **BLEU score**, a widely used metric for text generation quality.

### ****Model Architecture and Design Rationale****

Our image captioning model follows an **Encoder-Decoder architecture**, where a **Convolutional Neural Network (CNN)** extracts visual features from an image, and a **Recurrent Neural Network (RNN)** generates a textual description. This approach effectively captures both **spatial and sequential dependencies**, making it suitable for translating images into coherent sentences.

#### **1. Encoder: ResNet for Feature Extraction**

The encoder is a **pretrained ResNet model** (e.g., ResNet-34) that processes an input image and extracts meaningful feature representations.

* **Why ResNet?**
  + ResNet’s **skip connections** help prevent vanishing gradients, making it effective for deep learning tasks.

#### **2. Decoder: LSTM for Sentence Generation**

The decoder is a **Long Short-Term Memory (LSTM) network**, which takes the encoded image features and generates a caption word by word.

* **Why LSTM?**
  + LSTM handles **long-term dependencies** better than a standard RNN, preventing issues like vanishing gradients.
  + It maintains a **context vector**, allowing it to generate coherent sentences.
  + The hidden state of the LSTM is initialized using the **bottleneck layer** from the encoder output.

The decoder consists of:

1. **Embedding Layer**: Converts input word indices into dense word vectors.
2. **LSTM Cell**: Processes the embedded words and maintains context.
3. **Fully Connected Layer (Classifier)**: Predicts the next word from the LSTM output.
4. **Softmax Activation**: Converts logits into probabilities for word selection.

#### **3. Training Strategy**

* The model is trained using **Cross-Entropy Loss**, which is well-suited for multi-class classification tasks like word prediction.
* **Adam Optimizer** is used for efficient parameter updates.
* The model generates captions **autoregressively**, meaning each predicted word is fed back as input for the next word prediction.

### ****Data Preprocessing Steps and Training Strategy****

#### **1. Data Preprocessing Steps**

Before training the image captioning model, the dataset undergoes several preprocessing steps to ensure consistency and improve model performance.

##### **a. Image Processing**

* **Resizing**: All images are resized to a fixed dimension (e.g., **224×224**).
* **Normalization**: Pixel values are normalized to the range **[0,1]**.
* **Conversion to Tensors**: Images are transformed into PyTorch tensors for processing.

##### **b. Text Processing**

* **Cleaning Captions**: Captions are cleaned from whitespaces, punctuations, and special symbols.
* **Tokenization**: Captions are split into individual words.
* **Vocabulary Creation**:
  + A vocabulary is built from the dataset, mapping each unique word to an index.
  + Special tokens are added:
    - #START (indicating the start of a sentence)
    - #END (marking the end of a sentence)
    - #PAD (for padding shorter sentences)
    - #UNK (unknown words)
* **Padding and Encoding**:
  + Captions are converted to sequences of integers using the vocabulary.
  + Shorter captions are padded to a fixed length to ensure uniform input size.

#### **2. Training Strategy**

##### **a. Model Initialization**

* **Encoder (CNN - ResNet)**:
  + The last fully connected layer is removed, and the output is a feature vector.
* **Decoder (LSTM-based RNN)**:
  + The LSTM’s hidden state is initialized with the encoded image features.
  + The decoder generates words sequentially based on previous words and the encoded image representation.

##### **b. Loss Function**

* **Categorical Cross-Entropy Loss** is used to compare predicted word probabilities with the actual target words.
* The loss is **only computed on non-padding tokens** to prevent the model from being penalized for padded words.

##### **c. Optimization**

* **Adam Optimizer** is used for training due to its efficiency in handling sparse gradients and adaptive learning rates.

##### **d. Training Process**

1. **Forward Pass**:
   * The encoder extracts image features.
   * The decoder takes the encoded features and a caption sequence to generate predictions.
2. **Loss Computation**:
   * The predicted words are compared with the actual caption words.
3. **Backpropagation**:
   * Gradients are computed, and the model weights are updated using **Adam**.
4. **Checkpointing**:
   * The model is periodically saved to prevent data loss and allow for resuming training.
5. **Evaluation During Training**:
   * **BLEU Score** is used to measure how similar the predicted captions are to the true captions.

### ****Results****

#### **1. Quantitative Evaluation Metrics**

To assess the performance of the image captioning model, the **BLEU (Bilingual Evaluation Understudy) score** is used. BLEU measures the similarity between the generated and reference captions based on n-gram overlaps.

**Model Performance:**

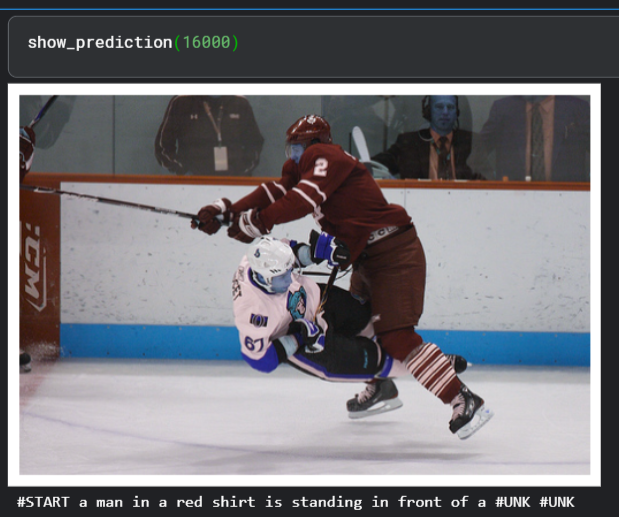
* BLEU-score: **0.2009%**

#### **2. Qualitative Results (Sample Captions)**

Here are some example captions generated by the model compared with the actual ground-truth captions:

##### **Example 1**

**Image:**



* **Ground Truth:** "a hockey player in black and white collides with another player wearing blue"
* **Generated Caption:** "#START a man in a red shirt is standing in front of a #UNK #UNK"

##### **Example 2**

**Image:**



* **Ground Truth:** "a thin brown horse standing and a small black horse sitting on a sand"
* **Generated Caption:** "#START a man in a red shirt is standing in front of a #UNK #UNK "

##### **Example 3**

**Image:**



* **Ground Truth:** "a young boy jumping off a chair"
* **Generated Caption:** "#START a man in a red shirt is standing in front of a #UNK #UNK"

**Limitations of the Current Approach**

1. **Limited Generalization to Unseen Data**
   * The model may struggle to generate accurate captions for images that contain objects, scenes, or interactions not well-represented in the training dataset.
2. **BLEU Score as the Only Evaluation Metric**
   * BLEU is useful for assessing n-gram similarity, but it does not measure **semantic correctness** or **fluency** effectively.
   * It does not account for synonyms or different phrasings that might still be valid captions.
3. **Bias in Caption Generation**
   * The model might overfit to frequent captions in the dataset, leading to **repetitive or generic outputs** (e.g., "a man is standing" for many images).
   * It may fail to properly describe **rare objects or actions**.

**Potential Improvements**

1. **Use of More Robust Evaluation Metrics**
   * In addition to BLEU, consider using:
     + **METEOR** (accounts for synonyms and word alignment)
     + **CIDEr** (measures consensus among multiple references)
     + **ROUGE-L** (considers sequence matching for fluency)
2. **Improving the Model Architecture**
   * **Replacing RNN-based decoder with Transformer-based models** (e.g., ViT + GPT-like decoder) for better long-range dependencies.
3. **Beam Search or Sampling for Caption Generation**
   * Instead of using argmax() during decoding, **applying beam search, top-k sampling, or nucleus sampling** to generate more natural and diverse captions.

## References

1. <https://arxiv.org/pdf/1502.03044v2>
2. <https://www.digitalocean.com/community/tutorials/writing-resnet-from-scratch-in-pytorch>
3. <https://medium.com/@wangdk93/lstm-from-scratch-c8b4baf06a8b>
4. <https://youtu.be/y2BaTt1fxJU?si=10vHpteH-vsX5bFf>