

# **LAB FILE**

for

AGENTIC AI LAB

CSCR3215

SEMESTER: VI<sup>th</sup>

Section: CS-F

Group: G2

## **SUBMITTED BY**

**Harshit Arora**

**2023277664**



SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY

SHARDA UNIVERSITY, GREATER NOIDA – 201310

# Fine-Tuning BLIP for Image Captioning

## Code Explanation (Fine\_Tuning.ipynb)

This document explains the complete workflow of the `Fine_Tuning.ipynb` notebook, which demonstrates how to fine-tune a pre-trained **Salesforce BLIP (Bootstrapping Language-Image Pre-training)** model for an **image captioning task** using a football-specific image-text dataset.

---

## 1. Setup and Installation

The notebook begins by installing the required Python libraries:

- **transformers**  
Installed directly from the Hugging Face GitHub repository to access the latest model architectures and updates.
- **datasets**  
Used to efficiently load, manage, and preprocess datasets from the Hugging Face Hub.

These libraries provide the necessary tools for model loading, data handling, training, and inference.

---

## 2. Loading the Dataset

The dataset used is **ybelkada/football-dataset**, loaded using the `load_dataset()` function from the `datasets` library.

Key points:

- Only the **train split** is loaded.
- Each data sample contains:
  - An **image** (football-related scene)
  - A corresponding **text caption**

The notebook displays a sample entry to understand the dataset structure and verify the image-caption pairing.

---

## 3. Custom PyTorch Dataset

A custom dataset class named **ImageCaptioningDataset** is created by extending `torch.utils.data.Dataset`.

**Class Methods:**

- `__init__()`

- Accepts the raw dataset and a processor object.
- Stores them for later use.
- `__len__()`
  - Returns the total number of samples in the dataset.
- `__getitem__()`
  1. Retrieves an image–caption pair.
  2. Uses the processor to preprocess inputs:
    - Image resizing and normalization
    - Caption tokenization
  3. Uses `padding="max_length"` to maintain uniform sequence lengths.
  4. Sets `return_tensors="pt"` to generate PyTorch tensors.
  5. Removes the extra batch dimension using `.squeeze()`.

This custom dataset prepares data in a format suitable for model training.

---

#### 4. Model and Processor Initialization

The notebook initializes the pre-trained model:

- **Model:** `BlipForConditionalGeneration`
- **Checkpoint:** `Salesforce/blip-image-captioning-base`
- **Processor:** `AutoProcessor`

The processor handles both:

- Image preprocessing
- Text tokenization

A **DataLoader** is created with:

- `Batch size = 2`
- `shuffle=True`

This enables efficient batch-wise training.

---

#### 5. Training Loop

The training logic is implemented using a standard PyTorch training loop.

**Key Components:**

### 1. Optimizer

- AdamW optimizer
- Learning rate: 5e-5

### 2. Device Configuration

- Uses GPU (cuda) if available
- Otherwise falls back to CPU

### 3. Training Process

- Runs for **50 epochs**
- For each batch:
  - input\_ids (text tokens) and pixel\_values (image tensors) are moved to the device
  - The model performs a forward pass
  - Labels are set as input\_ids to compute caption generation loss
  - Backpropagation is performed using loss.backward()
  - Model weights are updated using optimizer.step()
  - Gradients are reset using optimizer.zero\_grad()

This process fine-tunes the BLIP model on the football dataset.

---

## 6. Inference (Caption Generation)

After training, the notebook demonstrates how to generate captions:

1. An image is selected from the dataset.
2. The image is processed using the processor to obtain pixel\_values.
3. The model.generate() function produces output token IDs.
4. The processor decodes these tokens into a readable caption using batch\_decode().

This step validates the effectiveness of the fine-tuned model.

---

## 7. Loading a Fine-Tuned Model

The notebook also shows how to load an already fine-tuned model from the Hugging Face Hub:

- **Model:** ybelkada/blip-image-captioning-base-football-finetuned

This confirms successful training and allows visualization of generated captions on multiple images using matplotlib.

---

## Conclusion

This notebook provides a complete implementation of **fine-tuning a multimodal Transformer model for image captioning**, covering dataset preparation, model training, and inference using the Hugging Face ecosystem.