# **Practical Lab File for**

## **Agentic AI**

Sharda School of Engineering and Technology

Name- Ragini Singh

Sys ID- 2023405949

Semester/Year- 6th sem, 3rd Year

## Faculty-In-Charge/Submitted To **Mr. Ayush Singh**



**Sharda University**

Plot No. 32-34, Knowledge Park III, Greater Noida, Uttar Pradesh 201310

# Experiment 1

## Aim - Fine-tuning BLIP on an Image Captioning Dataset

**1. Introduction**

This notebook demonstrates how to **fine-tune the BLIP (Bootstrapped Language Image Pretraining)** model on a custom **image captioning dataset** using the **Hugging Face Transformers** and **Datasets** libraries.

BLIP is a vision–language model capable of understanding images and generating meaningful textual descriptions (captions).

**2. Objective**

The main objectives of this project are:

* To load an image–caption dataset
* To preprocess images and text using BLIP processor
* To fine-tune the BLIP model on the dataset
* To generate captions for images after training
* To visualize generated captions with images

**3. Technologies & Libraries Used**

* **Python**
* **PyTorch**
* **Hugging Face Transformers**
* **Hugging Face Datasets**
* **Matplotlib**
* **BLIP (Salesforce/blip-image-captioning-base)**

**4. Installation of Required Libraries**

!pip install git+https://github.com/huggingface/transformers.git@main

!pip install -q datasets

**Purpose:**

* Installs the latest version of the **Transformers** library from GitHub.
* Installs the **datasets** library for loading image-caption datasets.

**5. Dataset Loading**

from datasets import load\_dataset

dataset = load\_dataset("ybelkada/football-dataset", split="train")

**Dataset Details:**

* **Name:** ybelkada/football-dataset
* **Type:** Image-caption dataset
* **Split:** Training split
* Each dataset item contains:
  + image → football-related image
  + text → caption describing the image

**6. Dataset Inspection**

dataset[0]["text"]

**Purpose:**

* Displays the caption associated with the first image
* Helps understand the dataset structure

**7. Model & Processor Initialization**

The BLIP model and processor are used to handle both image and text data.

**Key Components:**

* **Processor**: Converts images and text into tensors
* **Model**: Generates captions from image embeddings

**8. Custom PyTorch Dataset Class**

class ImageCaptioningDataset(Dataset):

def \_\_init\_\_(self, dataset, processor):

self.dataset = dataset

self.processor = processor

def \_\_len\_\_(self):

return len(self.dataset)

def \_\_getitem\_\_(self, idx):

item = self.dataset[idx]

encoding = self.processor(

images=item["image"],

text=item["text"],

padding="max\_length",

return\_tensors="pt"

)

encoding = {k: v.squeeze() for k, v in encoding.items()}

return encoding

**Explanation:**

* Converts dataset into a format suitable for PyTorch training
* Handles:
  + Image preprocessing
  + Caption tokenization
* Removes extra batch dimensions for training compatibility

**9. DataLoader Creation**

The dataset is passed into a **DataLoader** for batch processing during training.

**Benefits:**

* Efficient batching
* Shuffling of training data
* GPU-friendly execution

**10. Model Training (Fine-Tuning)**

* The BLIP model is trained using:
  + Image pixel values
  + Corresponding captions
* Loss is calculated between generated and actual captions
* Optimizer updates model weights

*(Exact optimizer and training loop are handled internally in the notebook)*

**11. Caption Generation (Inference)**

inputs = processor(images=image, return\_tensors="pt").to(device)

pixel\_values = inputs.pixel\_values

generated\_ids = model.generate(pixel\_values=pixel\_values, max\_length=50)

generated\_caption = processor.batch\_decode(

generated\_ids, skip\_special\_tokens=True

)[0]

**Purpose:**

* Generates captions for unseen images
* Uses trained BLIP model for prediction

**12. Visualization of Results**

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(18, 14))

for i, example in enumerate(dataset):

image = example["image"]

...

plt.imshow(image)

plt.axis("off")

plt.title(f"Generated caption: {generated\_caption}")

**Output:**

* Displays images along with generated captions
* Helps visually evaluate model performance

**13. Results**

* The fine-tuned BLIP model successfully generates meaningful captions
* Captions align well with football-related image content
* Visualization confirms reasonable accuracy

**14. Conclusion**

This notebook successfully demonstrates:

✅ Fine-tuning a vision-language model  
✅ Using BLIP for image captioning  
✅ Handling multimodal datasets  
✅ Generating and visualizing captions

BLIP proves to be an effective model for image captioning tasks with minimal fine-tuning.

**15. Future Enhancements**

* Train on a larger and more diverse dataset
* Improve captions using beam search
* Deploy the model as a web application
* Add BLEU / ROUGE evaluation metrics

## 