Agentic AI Lab

B. TECH 3rd YEAR

SEMESTER: 6TH

SESSION:2026-27

SUBMITTED BY: **AYUSHI TIWARI (2023334003)**

SECTION H AND GROUP: G2

SUBMITTED TO **Mr. AYUSH KUMAR SINGH**

FACULTY DESIGNATION

ASSISITANT PROFESSOR @ SHARDA UNIVERSITY

NOIDA, UTTAR PRADESH, INDIA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SHARDA

SCHOOL OF COMPUTER SCIENCE & ENGINEERING SHARDA

UNIVERSITY, GREATER NOIDA

Fine-tuning BLIP on an Image Captioning Dataset

This document explains **what is happening step by step** in the notebook **`Fine_tune-BLIP_on_an_image_captioning_dataset.ipynb`**. The goal of the notebook is to **fine-tune a BLIP (Bootstrapped Language Image Pretraining) model** for **image captioning** using a football image dataset.

1. Environment Setup

Installing required libraries

```
%pip install git+https://github.com/huggingface/transformers.git@main
%pip install -q datasets
```

What this does:

- Installs the **latest development version** of Hugging Face `transformers` (needed for BLIP support).
- Installs the `datasets` library to load datasets from Hugging Face Hub.

2. Loading the Image Captioning Dataset

```
from datasets import load_dataset

dataset = load_dataset("ybelkada/football-dataset", split="train")
```

What this does:

- Downloads a **football image dataset** from Hugging Face.
- Each data sample contains:
  - An **image**
  - A **text caption** describing the image
- The dataset is loaded into memory as a Hugging Face `Dataset` object.

This dataset will be used to teach the model how to generate captions for football-related images.

3. Inspecting the Dataset

Typical operations (implicit or explicit):

- Checking dataset length
- Viewing a single sample

Purpose:

- Understand the structure of the data
- Ensure images and captions are correctly loaded

4. Loading the BLIP Model and Processor

```
from transformers import BlipForConditionalGeneration, AutoProcessor
```

Model

```
model = BlipForConditionalGeneration.from_pretrained(
    "Salesforce/blip-image-captioning-base"
)
```

What this does:

- Loads a **pretrained BLIP image captioning model**.
- BLIP is a **vision-language transformer** capable of understanding images and generating text.

Processor

```
processor = AutoProcessor.from_pretrained(
    "Salesforce/blip-image-captioning-base"
)
```

What the processor does:

- Converts **images → pixel tensors**
- Converts **captions → token IDs**
- Ensures inputs match what BLIP expects

Model + processor must always come from the **same checkpoint**.

5. Data Preprocessing

A preprocessing function is defined that:

- Takes an image and caption
- Processes them using the BLIP processor
- Returns tensors suitable for training

Key steps inside preprocessing:

- Resize and normalize images
- Tokenize text captions
- Align image and text into a single input dictionary

This step is crucial because:

- Raw images/text cannot be fed directly to the model

- Transformers require tensorized inputs

6. Dataset Transformation

```
dataset = dataset.map(preprocess_function,
remove_columns=dataset.column_names)
```

What this does:

- Applies preprocessing to every example in the dataset
- Removes unused raw columns (like raw images or strings)
- Produces a clean, model-ready dataset

7. Defining Training Arguments

```
from transformers import TrainingArguments
```

Training arguments specify:

- Batch size
- Learning rate
- Number of epochs
- Logging frequency
- Where checkpoints are saved

Example concepts used:

- **Per-device batch size** – how many samples per GPU/CPU step
- **Epochs** – how many times the model sees the full dataset
- **FP16 (optional)** – faster training using half-precision

8. Trainer Setup

```
from transformers import Trainer
```

The `Trainer` combines:

- The BLIP model
- Training arguments
- Training dataset
- Data collator (batch formation)

Why Trainer is used:

- Simplifies the training loop
- Handles backpropagation, optimization, logging, and checkpoints automatically

9. Fine-Tuning the Model

```
trainer.train()
```

What happens here:

- Images are passed through the vision encoder
- Captions are learned by the text decoder
- Model weights are updated using gradient descent
- BLIP adapts specifically to **football images and captions**

This is the core learning phase.

10. Saving the Fine-Tuned Model

```
model.save_pretrained("./blip-football-captioning")
processor.save_pretrained("./blip-football-captioning")
```

Why this matters:

- Saves the fine-tuned model locally
- Allows reuse for inference or further training
- Ensures processor and model stay aligned

11. Inference (Caption Generation)

Typical inference flow:

1. Load an image
2. Process image using the processor
3. Call `model.generate()`
4. Decode output tokens into text

Result:

- The model generates football-specific captions for unseen images

12. Overall Pipeline Summary

End-to-end flow:

1. Install dependencies
2. Load dataset
3. Load BLIP model + processor
4. Preprocess images and captions
5. Configure training
6. Fine-tune BLIP
7. Save model
8. Generate captions

13. Why BLIP Is Used Here

BLIP is ideal because:

- It jointly learns **vision + language**
- Pretrained on large image-text pairs
- Can be efficiently adapted to domain-specific datasets

In this notebook, BLIP is adapted to the **football domain**.

Final Outcome

After completing this notebook:

- You have a **custom image captioning model**
- The model performs better on football images than the generic BLIP model
- You can deploy or evaluate it for real-world captioning tasks