AGENTIC AI LAB

CSCR3215

B.Tech. (CSE)-VI Semester

School of Engineering & Technology
Department of Computer Science & Engineering

Submitted to:
Mr. Ayush Singh

Submitted by:
Sumit Shrestha
2023850898

1. Aim of the Experiment

The aim of this experiment is to fine-tune a pretrained BLIP (Bootstrapped Language–Image Pretraining) model on a football player image captioning dataset, so that the model can generate accurate and context-aware captions related specifically to football players, match situations, and training scenes.

2. Problem Statement

Generic image captioning models often generate vague captions such as "a person standing on a field" when applied to sports images. Such captions fail to capture:
The presence of a football player
The action being performed (dribbling, passing, shooting)
The football context (match or training)
The problem addressed in this lab is to improve caption specificity and semantic accuracy for football-related images using domain-specific fine-tuning.

3. Theory / Background

3.1 Image Captioning

Image captioning is a multimodal task that combines:
Computer Vision – to understand image content
Natural Language Processing (NLP) – to generate human-readable descriptions
Modern image captioning systems use transformer-based architectures that rely on attention mechanisms instead of traditional CNN–RNN pipelines.

3.2 BLIP (Bootstrapped Language–Image Pretraining)

BLIP is a vision–language model developed by Salesforce. It consists of:
Vision Encoder (ViT): Extracts visual features from images
Text Decoder: Generates captions from visual embeddings
Cross-Attention Layers: Align visual and textual representations
BLIP is pretrained on large-scale image–text pairs, making it suitable for transfer learning through fine-tuning.

4. Dataset Description

4.1 Dataset Domain

The dataset used in this experiment contains images of football players captured during:
Matches
Training sessions
On-field actions involving the football
Each image is paired with a caption describing the player and the action being performed.

4.2 Dataset Format
The dataset is stored in JSON format, where each entry contains:

```
{
  "image": "player_image.jpg",
  "caption": "A football player dribbles the ball during a match"
}
```

Images are stored in a directory, and captions are used as supervision during training.

5. Tools and Libraries Used
Python
PyTorch
Hugging Face Transformers
Hugging Face Datasets
PIL (Python Imaging Library)
TQDM
CUDA (for GPU acceleration, if available)

6. System Configuration
The system automatically selects the computation device:
GPU (CUDA) if available
CPU otherwise
This ensures optimal performance during training.

7. Implementation Details (With Code)
7.1 Importing Required Libraries

```
import torch
from torch.utils.data import DataLoader, Dataset
from datasets import load_dataset
from transformers import BlipProcessor, BlipForConditionalGeneration, AdamW
from PIL import Image
from tqdm import tqdm
import os
```

These libraries are required for deep learning, dataset handling, model loading, image processing, and training visualization.

7.2 Device Selection

```
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Using device:", device)
```

The code checks whether a GPU is available and assigns the appropriate device for training.

7.3 Loading Pretrained BLIP Model

```
model_name = "Salesforce/blip-image-captioning-base"

processor = BlipProcessor.from_pretrained(model_name)
model = BlipForConditionalGeneration.from_pretrained(model_name)
model.to(device)
```

A pretrained BLIP model and its processor are loaded from Hugging Face. The model is moved to the selected device.

7.4 Loading the Football Caption Dataset

```
dataset = load_dataset(
    "json",
    data_files={"train": "dataset/train/captions.json"}
)
```

The dataset is loaded from a JSON file containing football image–caption pairs.

7.5 Custom Dataset Class

```
class ImageCaptionDataset(Dataset):
    def __init__(self, dataset, image_folder, processor):
        self.dataset = dataset
        self.image_folder = image_folder
        self.processor = processor

    def __len__(self):
        return len(self.dataset)

    def __getitem__(self, idx):
        item = self.dataset[idx]
        image_path = os.path.join(self.image_folder, item["image"])
        image = Image.open(image_path).convert("RGB")
        caption = item["caption"]

        encoding = self.processor(
            images=image,
            text=caption,
            padding="max_length",
            truncation=True,
            return_tensors="pt"
        )

        encoding = {k: v.squeeze(0) for k, v in encoding.items()}
        encoding["labels"] = encoding["input_ids"]
        return encoding
```

This custom dataset class:
Loads football images
Loads corresponding captions
Applies BLIP preprocessing
Returns tensors suitable for training

7.6 DataLoader Creation

```
train_dataset = ImageCaptionDataset(
    dataset=dataset["train"],
    image_folder="dataset/train",
    processor=processor
)

train_dataloader = DataLoader(
    train_dataset,
    batch_size=4,
    shuffle=True
)
```

The DataLoader handles batching and shuffling during training.

7.7 Model Training

```
optimizer = AdamW(model.parameters(), lr=5e-5)

epochs = 3
model.train()

for epoch in range(epochs):
    print(f"Epoch {epoch + 1}/{epochs}")
    total_loss = 0
    for batch in tqdm(train_dataloader):
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
        total_loss += loss.item()

    avg_loss = total_loss / len(train_dataloader)
    print("Average Loss:", avg_loss)
```
The model is fine-tuned using supervised learning. Loss decreases across epochs, indicating learning.

7.8 Saving the Fine-Tuned Model
```
model.save_pretrained("blip_finetuned")
processor.save_pretrained("blip_finetuned")
```
The fine-tuned model and processor are saved for later use.

7.9 Caption Generation (Inference)
```
model.eval()

test_image = Image.open("test.jpg").convert("RGB")
inputs = processor(images=test_image, return_tensors="pt").to(device)

with torch.no_grad():
    generated_ids = model.generate(**inputs)

caption = processor.decode(generated_ids[0], skip_special_tokens=True)
print("Generated Caption:", caption)
```
The trained model generates a caption for a new football image.

8. Results and Output

The fine-tuned BLIP model generates captions that correctly describe:
Football players
On-field actions
Match or training context
The captions are more specific and meaningful compared to a generic pretrained model.

9. Conclusion

This experiment successfully demonstrates fine-tuning of a BLIP model for football player image captioning. The results show that domain-specific fine-tuning significantly improves caption quality and relevance.