B. TECH 3rd YEAR STUDENT

AGENTIC AI

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

Created By

Aayush Nangia

2023497017

LAB 1- FINE_TUNNING

WORKING CODE-

```
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

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

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

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

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

```
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

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

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

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)

model.save_pretrained("blip_finetuned")
processor.save_pretrained("blip_finetuned")

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)
```