import os

import torch

import torch.nn as nn

from torchvision import datasets, transforms, models

from torch.utils.data import DataLoader

from sklearn.metrics import classification\_report

# Set device

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

# Directories

data\_dir = "poultry\_dataset"

batch\_size = 32

num\_classes = len(os.listdir(data\_dir))

# Data transformations

transform = transforms.Compose([

transforms.Resize((224, 224)),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406],

std=[0.229, 0.224, 0.225])

])

# Datasets

dataset = datasets.ImageFolder(data\_dir, transform=transform)

train\_size = int(0.8 \* len(dataset))

test\_size = len(dataset) - train\_size

train\_ds, test\_ds = torch.utils.data.random\_split(dataset, [train\_size, test\_size])

train\_loader = DataLoader(train\_ds, batch\_size=batch\_size, shuffle=True)

test\_loader = DataLoader(test\_ds, batch\_size=batch\_size, shuffle=False)

# Load pre-trained model

model = models.resnet50(pretrained=True)

# Freeze base layers

for param in model.parameters():

param.requires\_grad = False

# Modify classifier

model.fc = nn.Linear(model.fc.in\_features, num\_classes)

model.to(device)

# Loss and optimizer

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.fc.parameters(), lr=0.001)

# Training loop

epochs = 10

for epoch in range(epochs):

model.train()

running\_loss = 0

for images, labels in train\_loader:

images, labels = images.to(device), labels.to(device)

optimizer.zero\_grad()

outputs = model(images)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

running\_loss += loss.item()

print(f"Epoch [{epoch+1}/{epochs}], Loss: {running\_loss/len(train\_loader):.4f}")

# Evaluation

model.eval()

all\_preds, all\_labels = [], []

with torch.no\_grad():

for images, labels in test\_loader:

images = images.to(device)

outputs = model(images)

\_, preds = torch.max(outputs, 1)

all\_preds.extend(preds.cpu())

all\_labels.extend(labels)

print(classification\_report(all\_labels, all\_preds, target\_names=dataset.classes))