PLANT DISEASE DETECTION SYSTEM

Submitted To:
Dr. Neetu Goel
Associate professor
VIPS

Submitted By:
Lavika Khattar
MCA 2A
01717704423

Introduction

This Python project aims to develop a robust “**Plant Disease Detection**” system utilizing the Plant Village dataset, a comprehensive collection of images depicting various diseases across multiple plant species.

The core of the system leverages a pretrained **ResNet50 model**, which has been trained on the Plant Village dataset. ResNet50 is a deep convolutional neural network architecture known for its superior performance in image recognition tasks.

KEY COMPONENTS OF THE PROJECT

01 DATASET

The dataset consists of high resolution images of diseased plants.

02 MODEL

ResNet50 is a pretrained deep CNN architecture.

03 IMPLEMENTATION

The project involves loading the pretrained model and fine tuning it on the dataset.

04 DEPLOYMENT

Deployed as a user friendly application.

OBJECTIVE

- Developing a plant disease detection system using deep learning techniques, specifically leveraging a pretrained convolutional neural network like ResNet50. The primary goal is to aid farmers in promptly identifying and managing disease affecting their crops.

Key Objectives are:

1. **Early Detection:** Spot diseases in plants as soon as possible.
2. **Save Crops**: Help farmers treat sick plants quickly to prevent crop loss.
3. **Increase Food Production**: Protect plants to grow more food.
4. **Farmers' Profitability**: Assist farmers in making more money by preserving their harvests.
5. **Environmental Care**: Minimize environmental impact by using fewer chemicals.

STEPS INVOLVED

1. Data Collection

2. Preprocessing

Resizing and cropping of images.

3. Model Training

The ResNet50 model is trained using preprocessed image dataset.

4. Deployment

The trained model is deployed as an application

MODEL TRAINING CODE

```
import os
import torch
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import torchvision.models as models
import torch.nn as nn
import torch.optim as optim
import multiprocessing

def train_model():
    # Check if GPU is available
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Using device:", device)

    # Define data transformers
    transformation = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])

    # Load the dataset
    training_dataset
=
ImageFolder(root="C:\\Users\\HP\\Desktop\\PlantDisease\\PlantVillage",
transform=transformation)
train_loader = DataLoader(training_dataset, batch_size=32, shuffle=True)

# Load pre-trained ResNet50 model
    model = models.resnet50(pretrained=True)

    # Modify the classifier to match the number of classes in your dataset
    num_ftrs = model.fc.in_features
    model.fc = nn.Linear(num_ftrs, len(training_dataset.classes))

    # Define loss function and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
    # Training loop
    num_epochs = 10
    for epoch in range(num_epochs):
        model.train()
        for inputs, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

    # Save the trained model
    save_dir = 'C:\\Users\\HP\\Desktop\\PlantDisease'
    save_path = os.path.join(save_dir, 'model.pth')

    # Save the model's state dictionary to a .pth file
    torch.save(model.state_dict(), save_path)

if __name__ == '__main__':
    multiprocessing.freeze_support()
    train_model()

train_loader = DataLoader(training_dataset, batch_size=32, shuffle=True)
```

DISEASE DETECTION

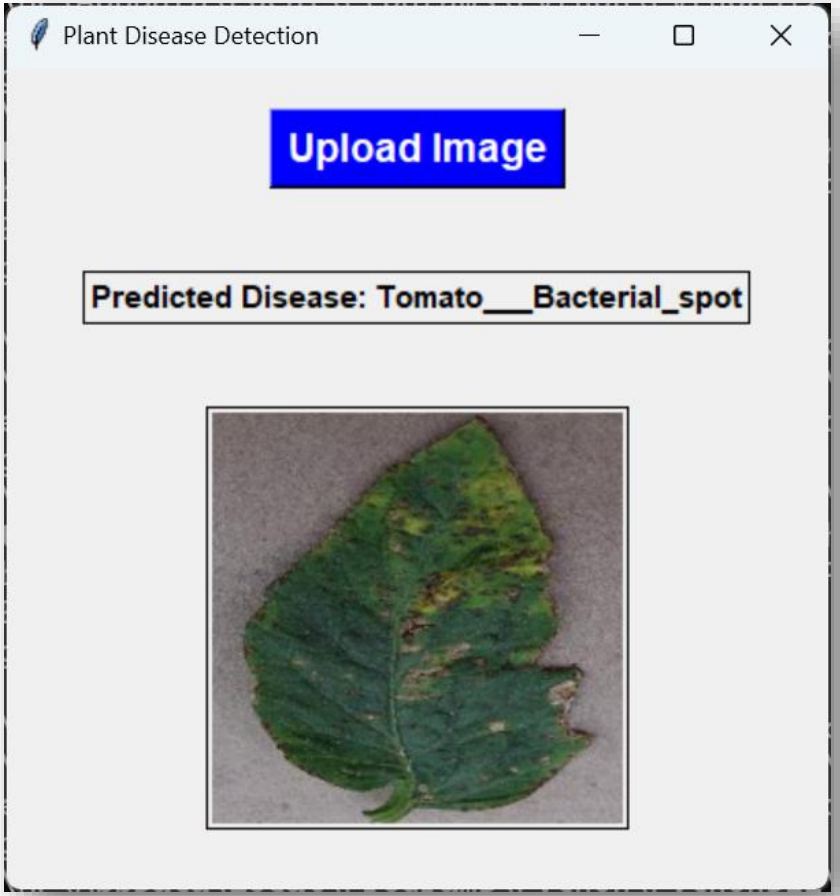
```
import torch
import torch.nn
import torchvision.models as models
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from torchvision.datasets import ImageFolder
from PIL import Image,ImageTk
import tkinter as tk
from tkinter import filedialog,messagebox
#define class labels
class_labels=[
'Pepper,_bell___Bacterial_spot',
'Pepper,_bell___healthy',
'Potato___Early_blight',
'Potato__healthy',
'Tomato___Bacterial_spot',
'Tomato__healthy',
'Tomato___Leaf_Mold',
'Tomato___Septoria_leaf_spot',
'Tomato___Spider_mites_Two-spotted_spider_mite']
```

```
#data transformation
transformation=transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485,0.456,0.406],std=[0.229,0.224,0.225])
])
model = models.resnet50(pretrained=True)
num_ftrs = model.fc.in_features
model.fc = torch.nn.Linear(num_ftrs, len(class_labels))  # Adjust for the number of classes
model.load_state_dict(torch.load('model.pth'))  # Load your trained model
model.eval()

#Loading the dataset
#dataset=ImageFolder(root='C:\\Users\\HP\\Desktop\\PlantDisease\\PlantVillage',transform
=transformation)
#data_loader=DataLoader(dataset,batch_size=1,shuffle=True)

def predict_disease(image_path):
  image= Image.open(image_path)
  image=transformation(image).unsqueeze(0)
  with torch.no_grad():
    outputs=model(image)
  _, predicted=torch.max(outputs,1)
  return class_labels[predicted.item()]
```

DISEASE DETECTION

```
def upload_image():
  file_path=filedialog.askopenfilename()
  image = Image.open(file_path)
  image = image.resize((200, 200))
# Convert the Image object into a Tkinter PhotoImage object
  photo = ImageTk.PhotoImage(image)
# Create a label to display the image
  selected_image.config(image=photo)
  selected_image.image = photo
#add border to image
  selected_image.config(highlightthickness=1,highlightbackground='black')
  if file_path:
    try:
      disease_name=predict_disease(file_path)
      result.config(text=f'Predicted Disease:
{disease_name}',font=('Arial',11,'bold'))
    except Exception as e:
      messagebox.showerror("Error",f"Error occured:{str(e)}")


root=tk.Tk()
root.title("Plant Disease Detection")
root.geometry("400x400")

upload_button=tk.Button(root,text="Upload
Image",command=upload_image,font=('Arial',14,'bold'),bg='blue',fg='white')
upload_button.pack(pady=20)

result=tk.Label(root,highlightthickness=1,highlightbackground='black')
result.pack(pady=20)

selected_image = tk.Label(root)
selected_image.pack(pady=20)

root.mainloop()
```

OUTPUT


Plant Disease Detection
Upload Image
Predicted Disease: Tomato___Bacterial_spot