AI-Powered Automated Plant Health Diagnosis System

Abstract:

This project presents an AI-powered system that detects and diagnoses plant diseases using deep learning and computer vision. By analyzing images of plant leaves, the system identifies diseases and suggests remedies. The tool is especially helpful for farmers, agronomists, and researchers aiming to monitor crop health effectively.

1. Introduction:

Manual disease identification is time-consuming and requires expert knowledge. Our proposed solution automates this process using a deep learning model trained on the PlantVillage dataset. We employ transfer learning using MobileNetV2 to achieve accurate predictions while keeping the model lightweight.

2. Dataset:

We use the publicly available PlantVillage dataset, which includes images of both healthy and diseased leaves across various plant species. The dataset is split into training and validation folders.

3. Preprocessing:

- Images are resized to 224x224 pixels.
- Normalized to range [0,1].
- Augmentation includes rotation, flipping, zooming.
- Labels are one-hot encoded for multi-class classification.

4. Model Architecture (MobileNetV2):

MobileNetV2 is a lightweight convolutional neural network designed for mobile and edge devices. We use it as the base model, excluding its top layers, and add a global average pooling layer, dropout, and dense softmax layer.

Code Snippet - Model Training:

```
----------------------------------------------------
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

IMG_SIZE = 224
BATCH_SIZE = 32
EPOCHS = 10

train_aug = ImageDataGenerator(rescale=1./255, rotation_range=20, zoom_range=0.15, horizontal_flip=True)
val_aug = ImageDataGenerator(rescale=1./255)

train_gen = train_aug.flow_from_directory('dataset/train', target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE, class_mode='categorical')
val_gen = val_aug.flow_from_directory('dataset/val', target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE, class_mode='categorical')

base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
output = Dense(len(train_gen.class_indices), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_gen, validation_data=val_gen, epochs=EPOCHS)
model.save('plant_disease_model.h5')
```

----

5. Web Deployment Using Flask:

We use Flask to build a web interface where users can upload leaf images and receive disease predictions.

Code Snippet - Flask App:

```
----------------------------------------------------
from flask import Flask, render_template, request
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np, os

app = Flask(__name__)
model = tf.keras.models.load_model(‘model/plant_disease_model.h5')
class_names = ['Healthy', 'Powdery mildew', 'Leaf spot', 'Rust']

@app.route(‘/')
def index():
    return render_template(‘index.html')

@app.route(‘/predict', methods=['POST'])
def upload():
    file = request.files['file']
    path = os.path.join('static', file.filename)
    file.save(path)
    img = image.load_img(path, target_size=(224, 224))
    img_array = np.expand_dims(image.img_to_array(img) / 255.0, axis=0)
    prediction = class_names[np.argmax(model.predict(img_array))]
    return render_template(‘index.html', prediction=prediction, img_path=path)
----------------------------------------------------
```

6. Output:

- Users upload a leaf image via browser.
- Model predicts and displays the disease class.
- Webpage shows uploaded image and the result.

7. Deployment Options:

- Run locally using `python app.py`.
- Host on cloud platforms like Heroku, Render.

- Use Docker for scalable deployment.

8. Future Scope:

- Extend support for more crops and regional diseases.
- Integrate remedies and expert suggestions.
- Convert to mobile app using TensorFlow Lite.

9. References:

- PlantVillage Dataset (https://www.kaggle.com/emmarex/plantdisease)
- TensorFlow Documentation
- MobileNetV2 Paper
- Flask Web Framework