**Plant Disease Detection Using Computer Vision**

In this project, we will build a **Plant Disease Detection System** using **Convolutional Neural Networks (CNNs)**. The system will classify plant leaves into healthy or diseased categories, helping in early diagnosis and improving agricultural productivity.

**Steps to Build the Project**

**1. Problem Definition**

* **Goal**: Detect diseases in plants by analyzing images of their leaves.
* **Output**: Predict whether a leaf is healthy or diseased and classify the specific disease type.

**2. Dataset**

* Commonly used dataset: [**PlantVillage Dataset**](https://www.kaggle.com/emmarex/plantdisease).
* The dataset contains images of healthy and diseased plant leaves categorized by plant species and disease type.

**3. Prerequisites**

* **Tools**: Python, TensorFlow/Keras or PyTorch.
* **Libraries**: NumPy, Pandas, Matplotlib, OpenCV, scikit-learn.
* **Environment**: Jupyter Notebook, Google Colab, or any Python IDE.

**Implementation Steps**

**Step 1: Import Required Libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import EarlyStopping

**Step 2: Load and Preprocess Data**

* Organize data into train, validation, and test sets.
* Resize images for uniformity and normalize pixel values.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Data augmentation and normalization

datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=30,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

validation\_split=0.2

)

# Load training and validation data

train\_generator = datagen.flow\_from\_directory(

'path\_to\_dataset',

target\_size=(128, 128),

batch\_size=32,

class\_mode='categorical',

subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

'path\_to\_dataset',

target\_size=(128, 128),

batch\_size=32,

class\_mode='categorical',

subset='validation'

)

**Step 3: Build the CNN Model**

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(128, 128, 3)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(train\_generator.num\_classes, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

**Step 4: Train the Model**

* Use early stopping to prevent overfitting.

early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)

history = model.fit(

train\_generator,

validation\_data=validation\_generator,

epochs=20,

callbacks=[early\_stopping]

)

**Step 5: Evaluate the Model**

* Evaluate on test data to assess performance.

# Test dataset generator

test\_generator = datagen.flow\_from\_directory(

'path\_to\_test\_dataset',

target\_size=(128, 128),

batch\_size=32,

class\_mode='categorical',

shuffle=False

)

# Evaluate model

test\_loss, test\_accuracy = model.evaluate(test\_generator)

print(f"Test Accuracy: {test\_accuracy \* 100:.2f}%")

**Step 6: Visualize Results**

* Plot training and validation accuracy and loss.

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

plt.title('Accuracy')

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.legend()

plt.title('Loss')

plt.show()

**Optional Enhancements**

1. **Use Pre-trained Models**: Use models like VGG16, ResNet, or MobileNet for Transfer Learning to improve accuracy.
2. from tensorflow.keras.applications import VGG16
3. base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(128, 128, 3))
4. model = Sequential([
5. base\_model,
6. Flatten(),
7. Dense(128, activation='relu'),
8. Dropout(0.5),
9. Dense(train\_generator.num\_classes, activation='softmax')
10. ])
11. **Add Explainability**:
    * Use Grad-CAM to visualize which parts of the image the model focuses on.
12. **Deploy the Model**:
    * Use **Flask** or **Streamlit** to create a web interface for uploading images and predicting diseases.

**Expected Outcome**

* A trained model capable of predicting whether a plant leaf is healthy or diseased.
* Achievable accuracy: 85–95% depending on dataset size and model complexity.

Let me know if you'd like to dive deeper into any section or need further clarification!

https://github.com/manthan89-py/Plant-Disease-Detection/blob/main/Model/Plant%20Disease%20Detection%20Code.ipynb