Microsoft Elevate

CAPSTONE PROJECT

AI-BASED CROP DISEASE DETECTION SYSTEM

PRESENTED BY

STUDENT NAME: VANITA GURUDAS ABNAVE

COLLEGE NAME: PROGRESSIVE EDUCATION SOCIETY MODERN INSTITUTE OF BUSINESS STUDIES, NIGDI

DEPARTMENT: MCA

EMAIL ID: vanitaa231@gmail.com

OUTLINE

- **Problem Statement** (Should not include solution)
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT:

Farmers face significant crop losses due to delayed or inaccurate identification of plant diseases. In many rural regions, access to agricultural experts is limited, and traditional disease detection methods are time-consuming and unreliable. There is a need for an automated, accurate, and accessible system that can detect crop diseases using leaf images and provide timely recommendations to farmers.

PROPOSED SOLUTION:

- To address the problem of delayed and inaccurate crop disease identification, this project proposes an AI-driven automated crop disease detection system based on image classification using deep learning techniques.
- The system is designed to analyze images of plant leaves and accurately classify them into healthy or diseased categories. By leveraging Convolutional Neural Networks (CNN), the model can automatically extract important visual features such as color variations, texture patterns, and leaf spots that indicate the presence of disease.
- **System Workflow**

The proposed solution follows a structured workflow:

- **Image Acquisition:** Farmers capture or upload an image of a crop leaf using a web-based interface.
- **Image Preprocessing:** The uploaded image is resized to a fixed dimension (e.g., 128x128 pixels) and normalized to improve model performance. Noise reduction and scaling techniques are applied if necessary.
- **Feature Extraction using CNN**
  A Convolutional Neural Network automatically extracts relevant features from the image without manual intervention.
- **Disease Classification**
  The trained model classifies the leaf image into predefined categories such as:
  - Healthy
  - Early Blight
  - Late Blight
  - Leaf Spot
    (Based on selected crops)
- **Result Display and Recommendation**
  The system displays:
  - Predicted disease name
  - Confidence score
  - Basic preventive measures and treatment suggestions

SYSTEM APPROACH:

The Crop Disease Detection System is developed using a deep learning–based approach combined with a web-based interface for user interaction.

The development process follows these stages:

- **Data Collection and Preparation**

The PlantVillage dataset is used for training the model. Images are preprocessed by resizing, normalization, and splitting into training and validation sets to improve model performance.

- **Model Development**

A Convolutional Neural Network (CNN) is implemented using **TensorFlow and Keras** in Python. The CNN architecture includes convolutional layers, pooling layers, and fully connected layers for accurate disease classification.

- **Model Training and Evaluation**

The model is trained using Google Colab with GPU support. Performance is evaluated using accuracy and validation metrics.

- **Deployment and Interface Development**

The trained model is saved and integrated into a simple web interface developed using **Flask, HTML, and CSS**, allowing users to upload leaf images and receive predictions.

- **Technologies Used**

Programming Language: Python

Deep Learning Framework: TensorFlow / Keras

Image Processing: OpenCV

Platform: Google Colab

Web Framework (Optional): Flask

Frontend: HTML, CSS

This approach ensures an efficient, scalable, and user-friendly system for automated crop disease detection.

ALGORITHM

- The Crop Disease Detection System uses a Convolutional Neural Network (CNN) for image classification. The algorithm works as follows:

- **Step 1: Data Input**
  Load the crop leaf image dataset.
  Selected crops: Apple, Grape, Strawberry.
  Label images according to disease categories.

- **Step 2: Data Preprocessing**
  Resize images to a fixed size (224 × 224 pixels).
  Normalize pixel values to range [0,1].
  Split dataset into training and validation sets.

- **Step 3: Model Initialization**
  Initialize CNN model.
  Add Convolution layers with ReLU activation.
  Add MaxPooling layers to reduce dimensions.
  Flatten the output.
  Add Fully Connected (Dense) layers.
  Add Output layer with Softmax activation.

- **Step 4: Model Training**
  Compile model using Adam optimizer.
  Use categorical cross-entropy as loss function.
  Train model using training dataset for specified epochs.
  Validate using validation dataset.

- **Step 5: Model Evaluation**
  Calculate training and validation accuracy.
  Save trained model.

- **Step 6: Prediction Phase**
  Accept new leaf image from user.
  Preprocess image.
  Pass image to trained model.
  Predict disease class.
  Display disease name and recommendations.

DEPLOYMENT

- The deployment phase ensures that the trained model can be used by end users (farmers or agricultural workers).
- **Deployment Process**

1. Model Saving

After training, the CNN model is saved in .h5 format.

2. Backend Integration

The saved model is loaded into a Flask application.

An API endpoint is created to accept image uploads.

3. User Interface

A simple web interface is developed using HTML and CSS.

Users can upload crop leaf images through the interface.

4. Prediction Workflow

Uploaded image is sent to backend.

Backend preprocesses the image.

Model predicts disease.

Result is displayed on webpage.

5. Hosting Options

The system can be deployed using:

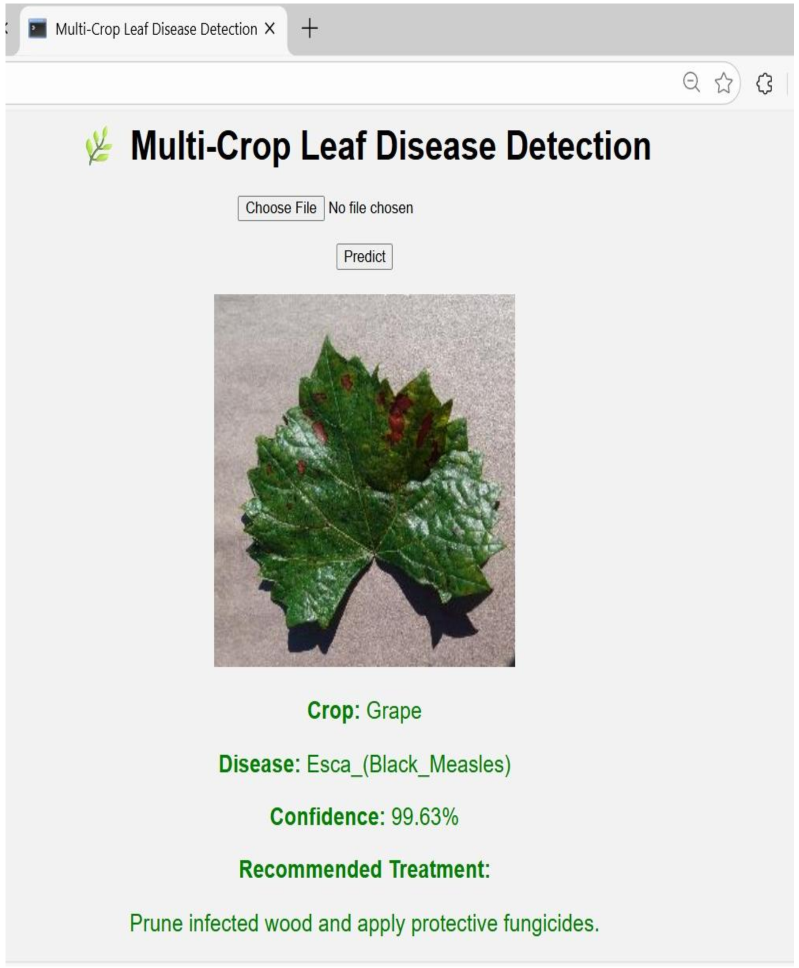
Local server (for demonstration)

Heroku / Render

Google Cloud Platform

AWS (advanced level)

RESULT:



Multi-Crop Leaf Disease Detection

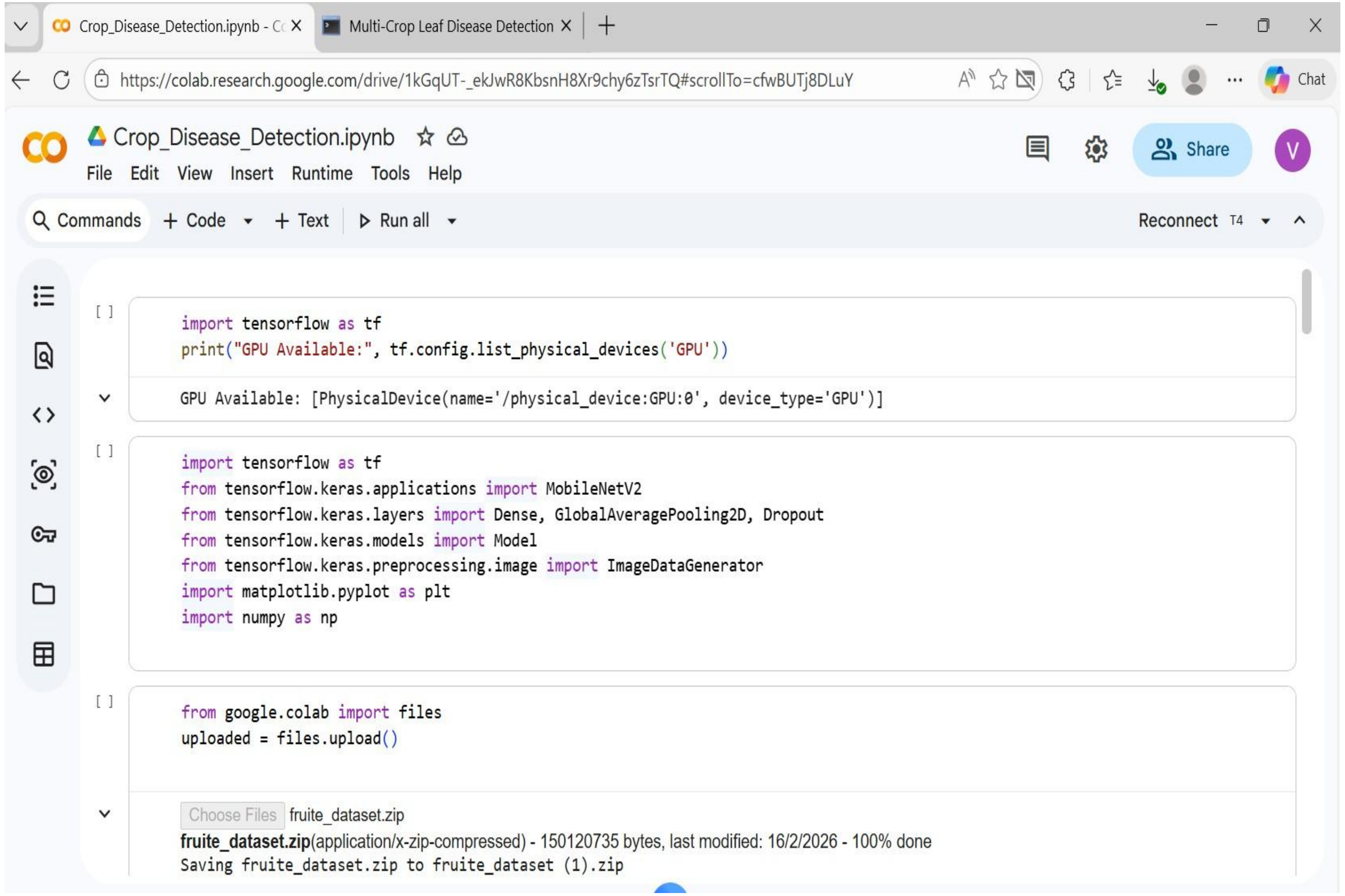
Crop: Grape

Disease: Esca_(Black_Measles)

Confidence: 99.63%

Recommended Treatment:

Prune infected wood and apply protective fungicides.


Crop_Disease_Detection.ipynb
Multi-Crop Leaf Disease Detection
https://colab.research.google.com/drive/1kGqUT-_ekJwR8KbsnH8Xr9chy6zTsrTQ#scrollTo=cfwBUTj8DLuY
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Reconnect T4
Share

```
import tensorflow as tf
print("GPU Available:", tf.config.list_physical_devices('GPU'))
```

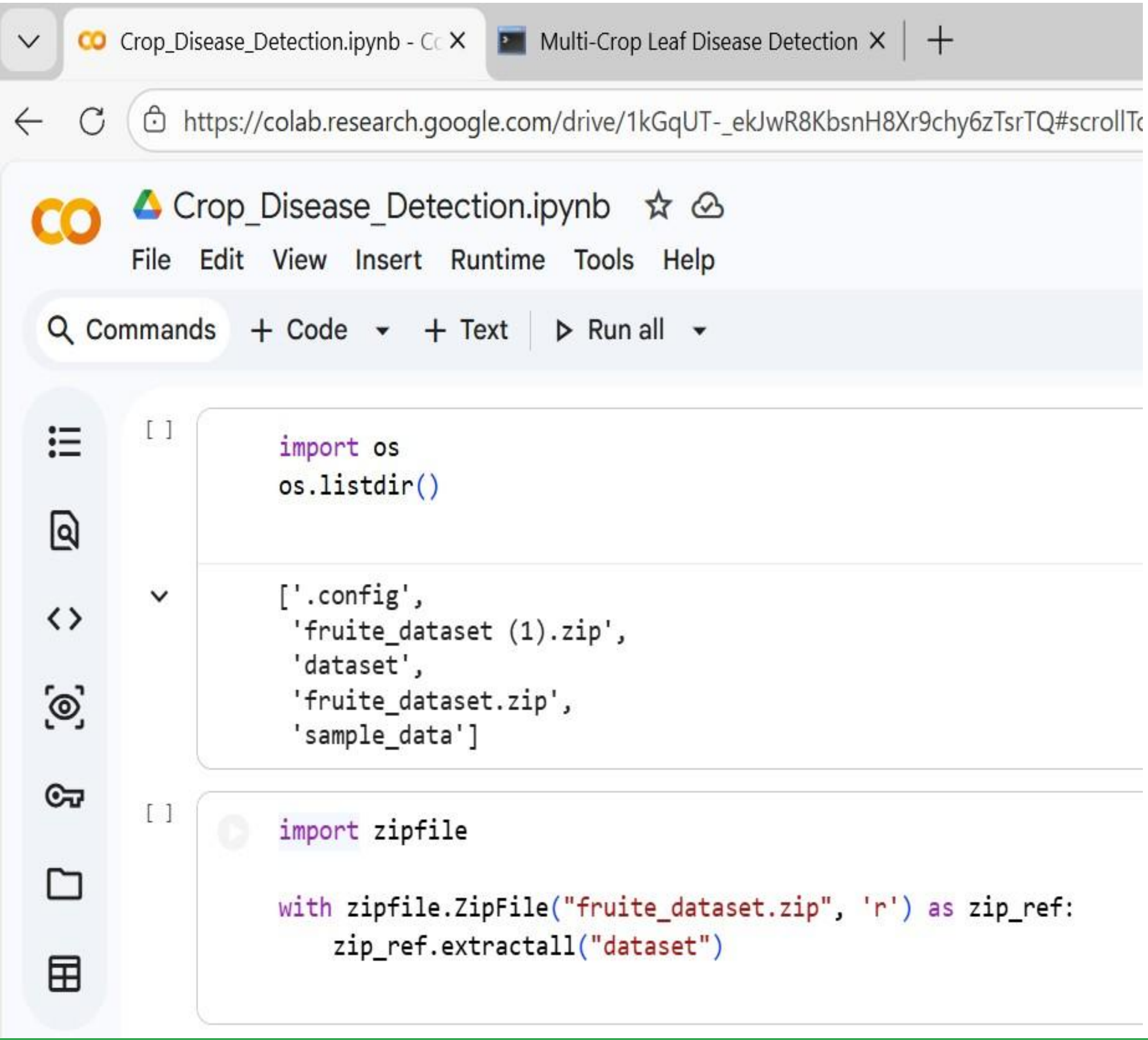
```
GPU Available: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files fruite_dataset.zip
fruite_dataset.zip(application/x-zip-compressed) - 150120735 bytes, last modified: 16/2/2026 - 100% done
Saving fruite_dataset.zip to fruite_dataset (1).zip
```

RESULT:

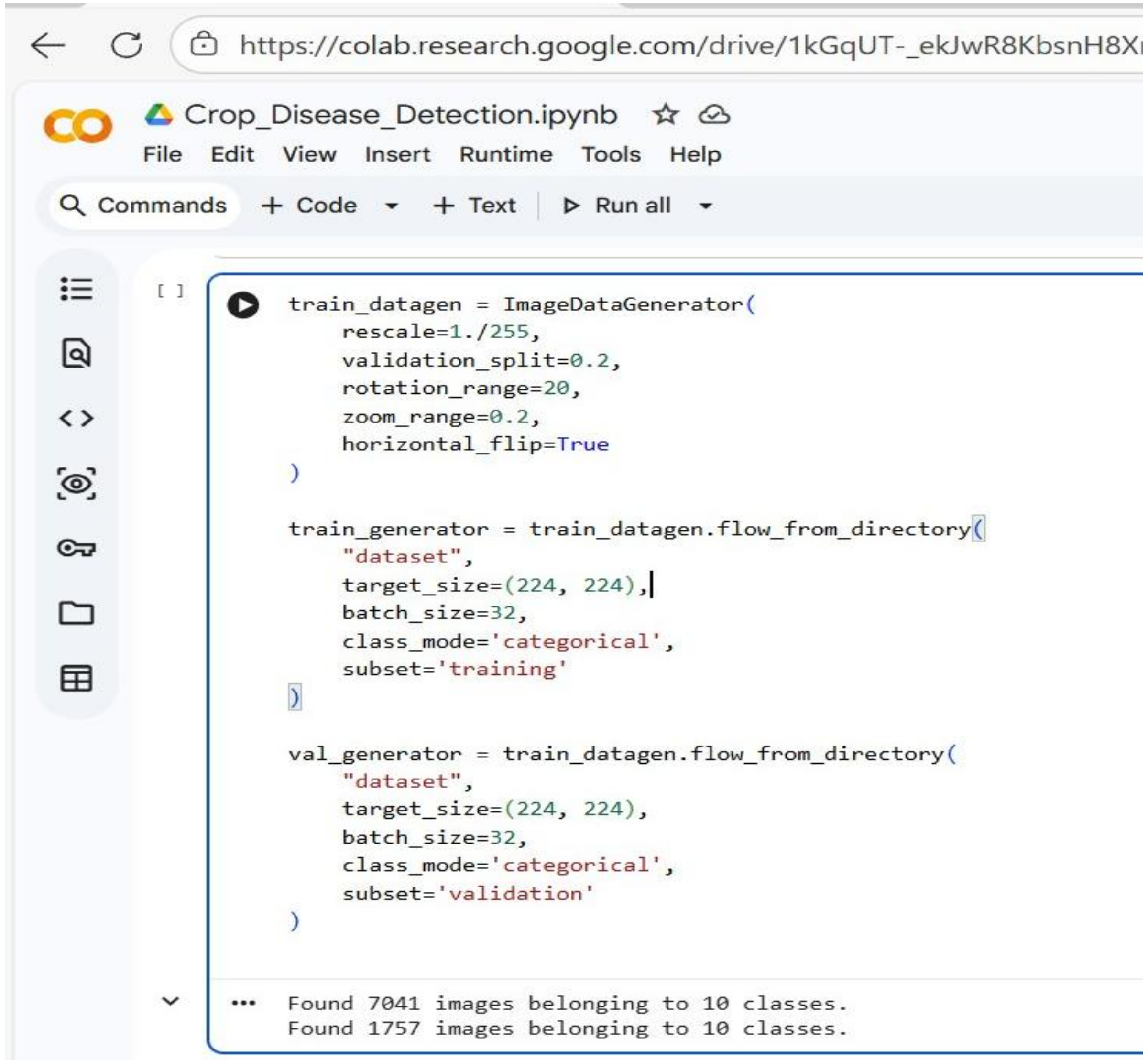


Crop_Disease_Detection.ipynb
Multi-Crop Leaf Disease Detection
https://colab.research.google.com/drive/1kGqUT-_ekJwR8KbsnH8Xr9chy6zTsrTQ#scrollT
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
import os
os.listdir()
```

```
['.config',
 'fruite_dataset (1).zip',
 'dataset',
 'fruite_dataset.zip',
 'sample_data']
```

```
import zipfile

with zipfile.ZipFile("fruite_dataset.zip", 'r') as zip_ref:
    zip_ref.extractall("dataset")
```



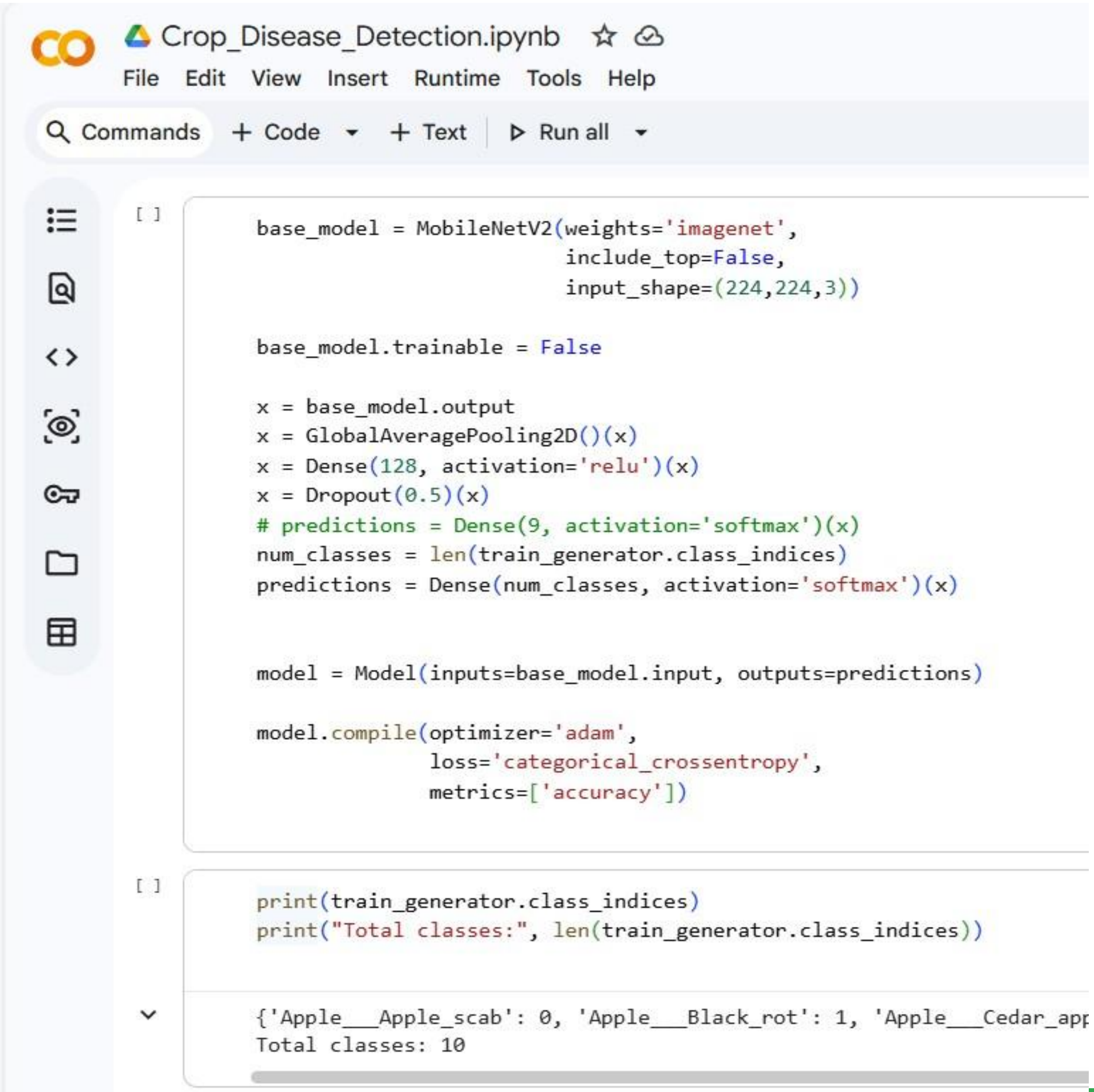
https://colab.research.google.com/drive/1kGqUT-_ekJwR8KbsnH8X
Crop_Disease_Detection.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    "dataset",
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    "dataset",
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 7041 images belonging to 10 classes.
Found 1757 images belonging to 10 classes.
```

Microsoft Elevate

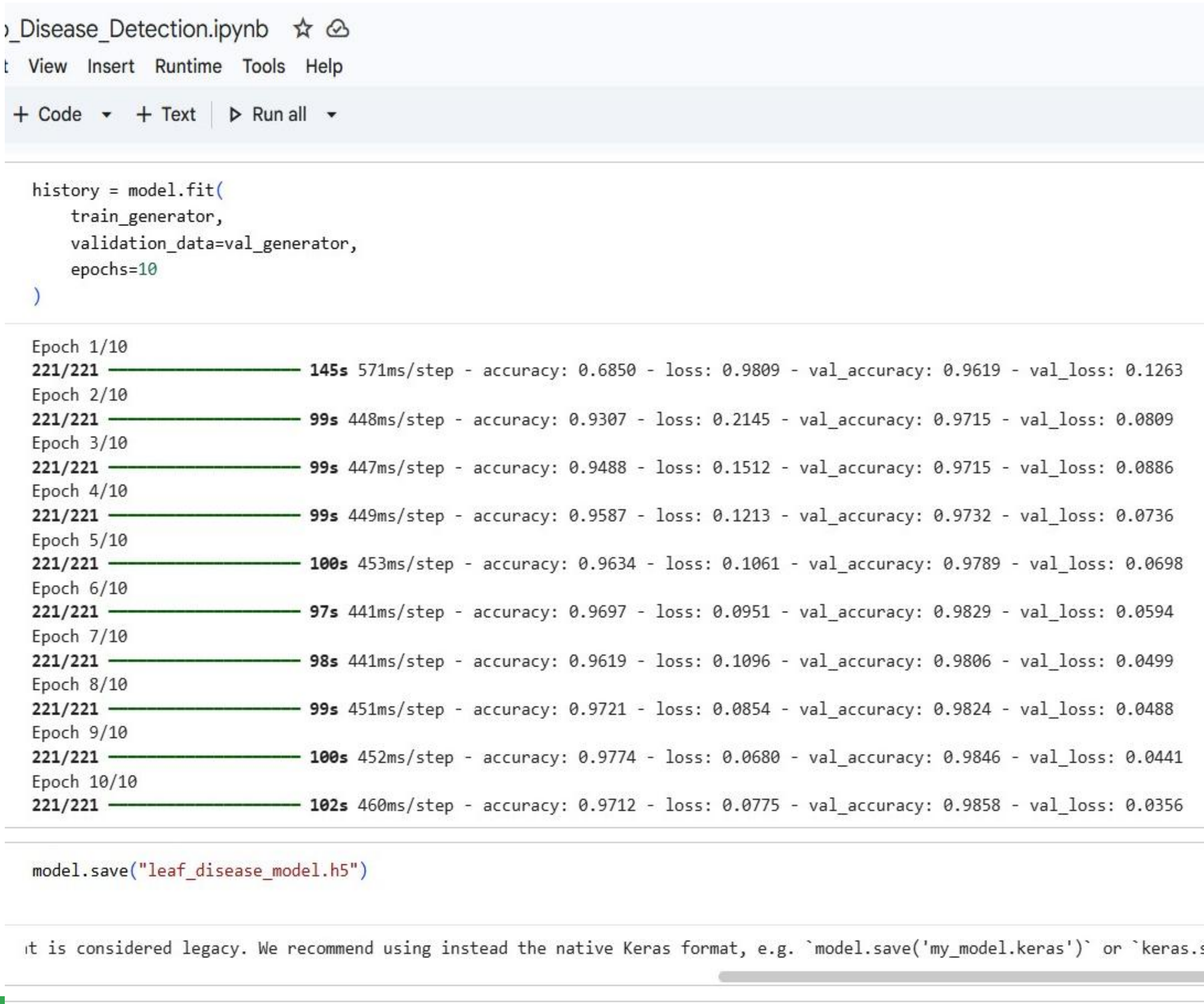
RESULT:


Crop_Disease_Detection.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
base_model = MobileNetV2(weights='imagenet',
                         include_top=False,
                         input_shape=(224,224,3))

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
# predictions = Dense(9, activation='softmax')(x)
num_classes = len(train_generator.class_indices)
predictions = Dense(num_classes, activation='softmax')(x)


model = Model(inputs=base_model.input, outputs=predictions)

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
print(train_generator.class_indices)
print("Total classes:", len(train_generator.class_indices))
```

```
{'Apple___Apple_scab': 0, 'Apple___Black_rot': 1, 'Apple___Cedar_app
Total classes: 10
```



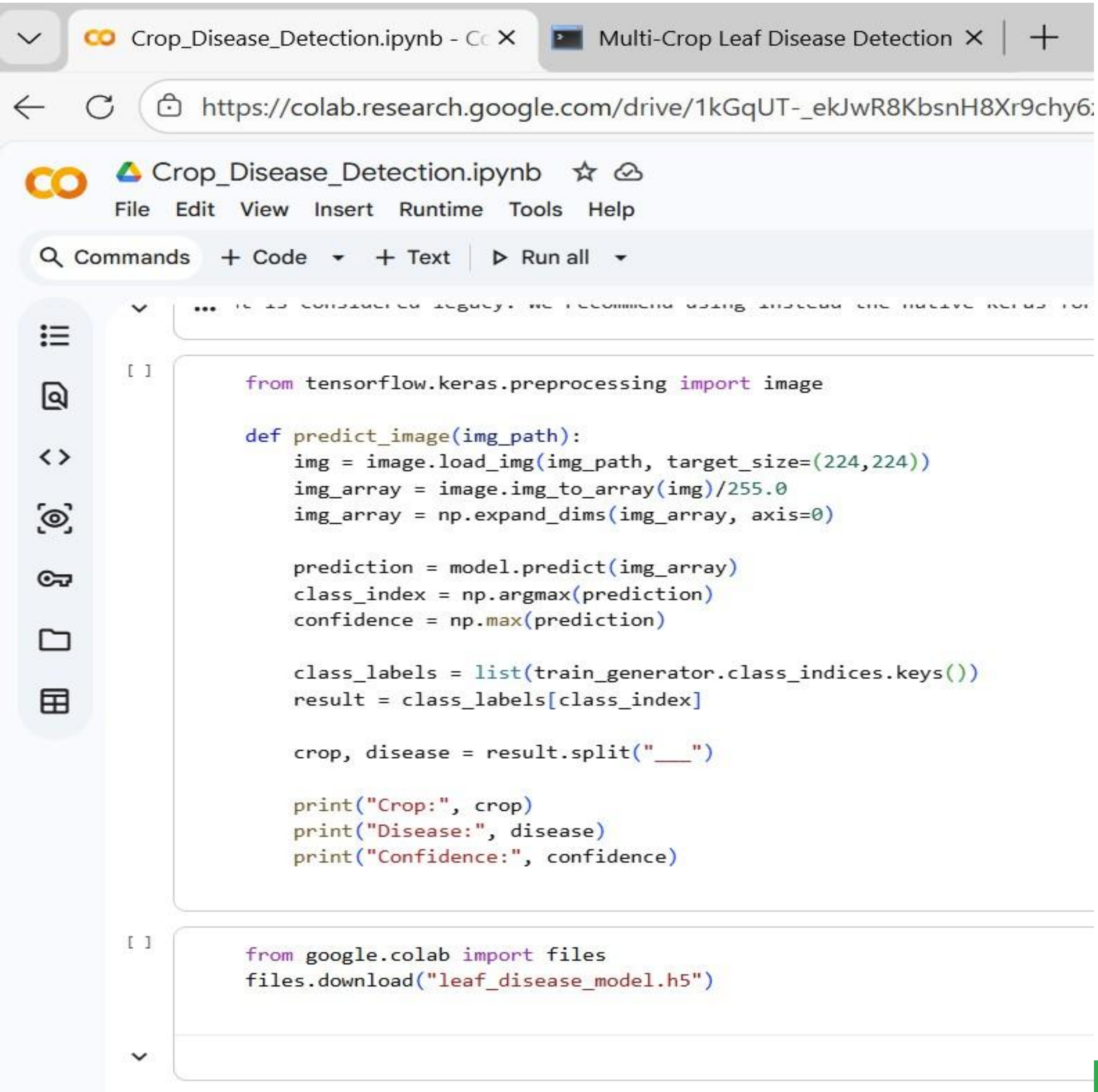
```
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=10
)
```

```
Epoch 1/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 145s 571ms/step - accuracy: 0.6850 - loss: 0.9809 - val_accuracy: 0.9619 - val_loss: 0.1263
Epoch 2/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 99s 448ms/step - accuracy: 0.9307 - loss: 0.2145 - val_accuracy: 0.9715 - val_loss: 0.0809
Epoch 3/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 99s 447ms/step - accuracy: 0.9488 - loss: 0.1512 - val_accuracy: 0.9715 - val_loss: 0.0886
Epoch 4/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 99s 449ms/step - accuracy: 0.9587 - loss: 0.1213 - val_accuracy: 0.9732 - val_loss: 0.0736
Epoch 5/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 100s 453ms/step - accuracy: 0.9634 - loss: 0.1061 - val_accuracy: 0.9789 - val_loss: 0.0698
Epoch 6/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 97s 441ms/step - accuracy: 0.9697 - loss: 0.0951 - val_accuracy: 0.9829 - val_loss: 0.0594
Epoch 7/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 98s 441ms/step - accuracy: 0.9619 - loss: 0.1096 - val_accuracy: 0.9806 - val_loss: 0.0499
Epoch 8/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 99s 451ms/step - accuracy: 0.9721 - loss: 0.0854 - val_accuracy: 0.9824 - val_loss: 0.0488
Epoch 9/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 100s 452ms/step - accuracy: 0.9774 - loss: 0.0680 - val_accuracy: 0.9846 - val_loss: 0.0441
Epoch 10/10
221/221 ━━━━━━━━━━━━━━━━━━━━ 102s 460ms/step - accuracy: 0.9712 - loss: 0.0775 - val_accuracy: 0.9858 - val_loss: 0.0356
```

```
model.save("leaf_disease_model.h5")
```

```
it is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.s
```

RESULT:

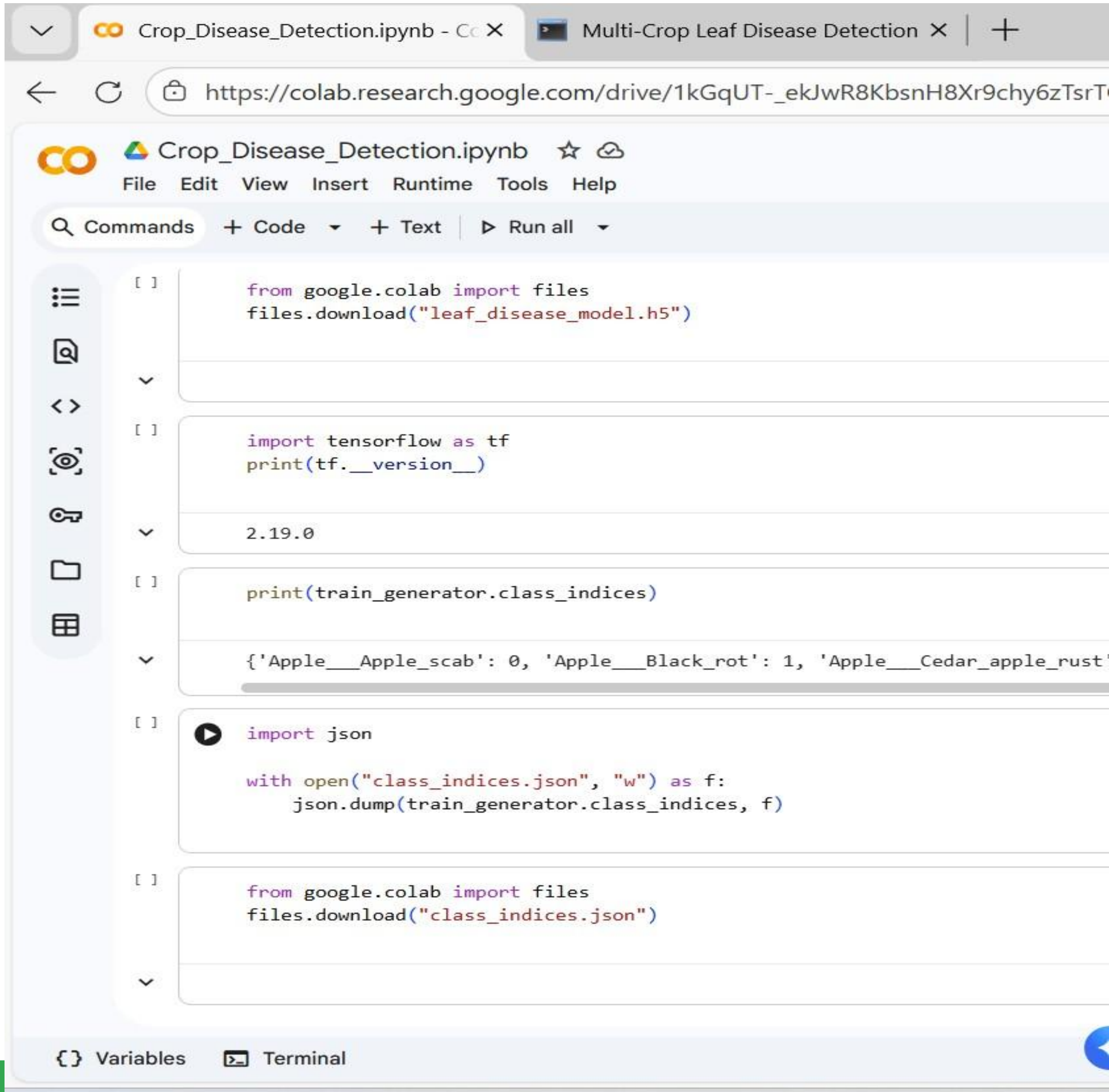


Crop_Disease_Detection.ipynb - Cc ✕ | Multi-Crop Leaf Disease Detection ✕

https://colab.research.google.com/drive/1kGqUT-_ekJwR8KbsnH8Xr9chy6

```
from tensorflow.keras.preprocessing import image

def predict_image(img_path):
    img = image.load_img(img_path, target_size=(224,224))
    img_array = image.img_to_array(img)/255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    class_index = np.argmax(prediction)
    confidence = np.max(prediction)

    class_labels = list(train_generator.class_indices.keys())
    result = class_labels[class_index]

    crop, disease = result.split("___")

    print("Crop:", crop)
    print("Disease:", disease)
    print("Confidence:", confidence)
```

```
from google.colab import files
files.download("leaf_disease_model.h5")
```



Crop_Disease_Detection.ipynb - Cc ✕ | Multi-Crop Leaf Disease Detection ✕

https://colab.research.google.com/drive/1kGqUT-_ekJwR8KbsnH8Xr9chy6zTsrT

```
from google.colab import files
files.download("leaf_disease_model.h5")
```

```
import tensorflow as tf
print(tf.__version__)
```

```
2.19.0
```

```
print(train_generator.class_indices)
```

```
{'Apple___Apple_scab': 0, 'Apple___Black_rot': 1, 'Apple___Cedar_apple_rust'
```

```
import json

with open("class_indices.json", "w") as f:
    json.dump(train_generator.class_indices, f)
```

```
from google.colab import files
files.download("class_indices.json")
```

CONCLUSION

- The Multi-Crop Leaf Identification and Disease Detection System successfully demonstrates the application of Artificial Intelligence and Deep Learning in agriculture. By using Transfer Learning with MobileNetV2, the system accurately identifies the crop type and detects associated diseases from leaf images.
- The developed model provides reliable predictions along with treatment recommendations, making it practically useful for farmers and agricultural stakeholders. The system reduces the need for manual inspection, saves time, and supports early disease detection, which can help improve crop yield and reduce losses.
- This project highlights how AI and Machine Learning can contribute to smart farming and sustainable agriculture practices.

FUTURE SCOPE:

- The proposed system can be further enhanced to support a larger variety of crops and plant diseases to make it more comprehensive. In the future, the model can be trained on real-field images instead of controlled dataset images to improve real-world accuracy.
- The system can also be integrated into a mobile application, allowing farmers to capture and analyze leaf images directly from their smartphones. Additionally, integrating weather data and soil analysis can help provide preventive recommendations along with treatment suggestions.
- Deployment of the system on cloud platforms can enable large-scale access and contribute to the development of smart and sustainable agriculture solutions.

REFERENCES:

GitHub Link: vanitaa231/AICTE_CDD_VanitaAbnave: AICTE project of Crop Disease Detection system for Fruit crops leaf

Dataset : PlantVillage Dataset (Use Fruit classs and color images only).

Microsoft Elevate

Thank You