PLANT DISEASE DETECTION

Install & Import Libraries

```
!pip install tensorflow matplotlib

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import os
import random
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflo
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensor
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorf
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflo
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorf
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from te
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorf
Requirement already satisfied: tensorboard~=2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorf
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensor
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from ten
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotli
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotli
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplo
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunpar
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorfl
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorf
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensor
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from re
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich-
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from ric
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=
```

Upload and Extract Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
```

```
import os, json
from zipfile import ZipFile
```

```
# Load Kaggle credentials
kaggle_dict = json.load(open("kaggle.json"))
os.environ["KAGGLE_USERNAME"] = kaggle_dict["username"]
os.environ["KAGGLE_KEY"] = kaggle_dict["key"]

# Download dataset
!kaggle datasets download -d abdallahalidev/plantvillage-dataset

# Unzip the dataset
with ZipFile("plantvillage-dataset.zip", "r") as zip_ref:
    zip_ref.extractall("/content/plantvillage_dataset")

print("Dataset extracted successfully!")
print(os.listdir("/content/plantvillage_dataset"))
```

```
Dataset URL: https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset
License(s): CC-BY-NC-SA-4.0
plantvillage-dataset.zip: Skipping, found more recently modified local copy (use --force to force download)
Dataset extracted successfully!
['plantvillage dataset']
```

Set random seeds for reproducibility

```
seed_value = 42
tf.random.set_seed(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
```

```
import os

for root, dirs, files in os.walk("/content/plantvillage_dataset"):
    if 'color' in dirs:
        print("Found 'color' folder at:", os.path.join(root, 'color'))
```

```
Found 'color' folder at: /content/plantvillage_dataset/plantvillage dataset/color
```

Prepare Data Generators with Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

base_dir = "/content/plantvillage_dataset/plantvillage dataset/color"

img_size = 128
batch_size = 32
seed_value = 42

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,  # 80% train, 20% validation
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_data = datagen.flow_from_directory(
    base_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    seed=seed_value
)

val_data = datagen.flow_from_directory(
```

```
val_data = datagen.flow_from_directory(
    base_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    seed=seed_value
)

print("Number of Classes:", len(train_data.class_indices))
```

```
Found 43456 images belonging to 38 classes.
Found 10849 images belonging to 38 classes.
Number of Classes: 38
```

Build CNN Model

```
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 3)),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(train_data.class_indices), activation='softmax')
])

model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do no
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3,211,392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 38)	4,902

Total params: 3,309,542 (12.62 MB)
Trainable params: 3,309,542 (12.62 MB)
Non-trainable params: 0 (0.00 B)

Train the model

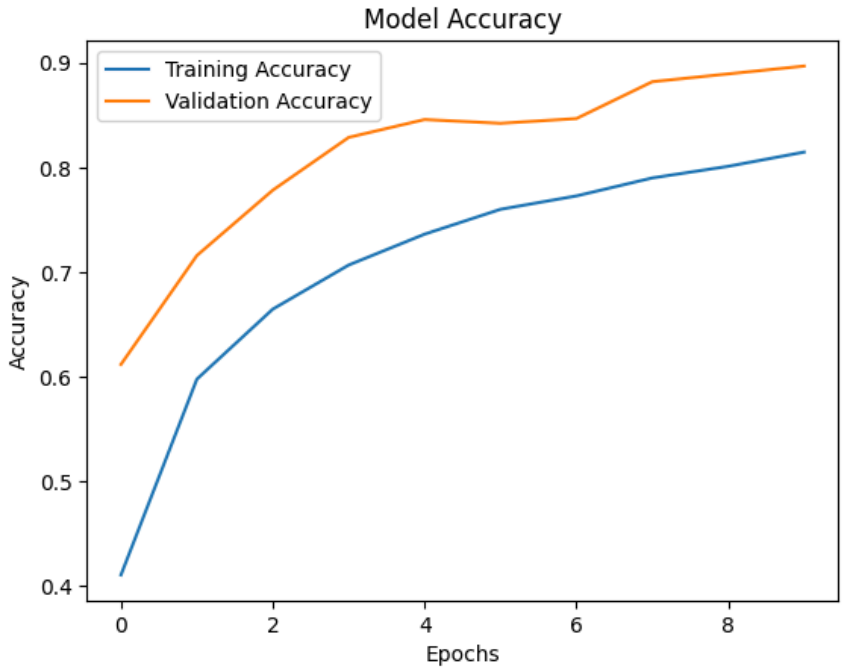
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
```

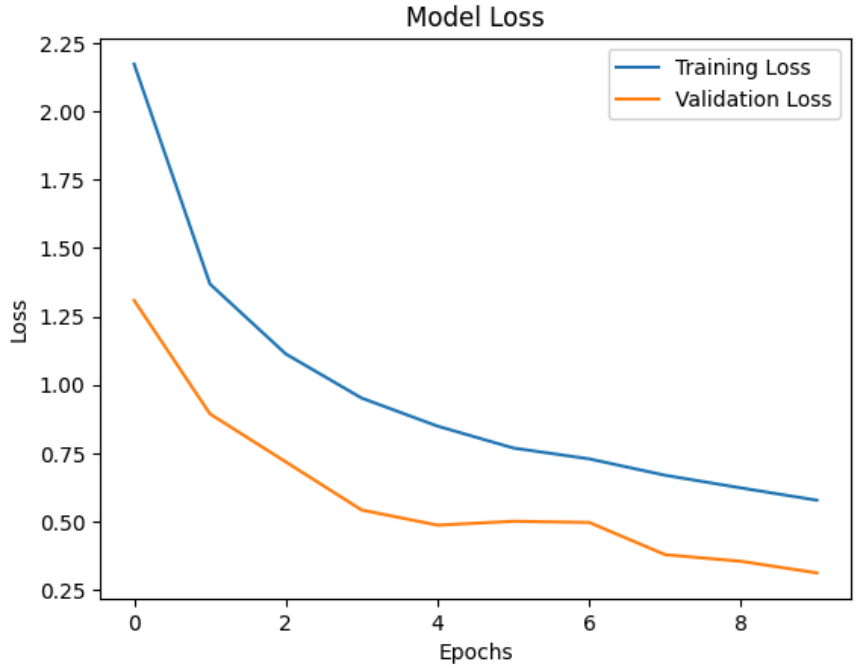
```
                metrics=['accuracy'])

history = model.fit(
    train_data,
    epochs=10,
    validation_data=val_data
)
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWar
  self._warn_if_super_not_called()
Epoch 1/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 266s 191ms/step - accuracy: 0.3087 - loss: 2.6213 - val_accuracy: 0.6119 - v
Epoch 2/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 254s 187ms/step - accuracy: 0.5670 - loss: 1.4841 - val_accuracy: 0.7158 - v
Epoch 3/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 253s 186ms/step - accuracy: 0.6538 - loss: 1.1644 - val_accuracy: 0.7782 - v
Epoch 4/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 252s 185ms/step - accuracy: 0.7024 - loss: 0.9672 - val_accuracy: 0.8286 - v
Epoch 5/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 253s 186ms/step - accuracy: 0.7304 - loss: 0.8729 - val_accuracy: 0.8457 - v
Epoch 6/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 252s 186ms/step - accuracy: 0.7519 - loss: 0.7910 - val_accuracy: 0.8421 - v
Epoch 7/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 251s 185ms/step - accuracy: 0.7706 - loss: 0.7262 - val_accuracy: 0.8466 - v
Epoch 8/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 251s 185ms/step - accuracy: 0.7867 - loss: 0.6819 - val_accuracy: 0.8818 - v
Epoch 9/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 251s 185ms/step - accuracy: 0.7997 - loss: 0.6306 - val_accuracy: 0.8893 - v
Epoch 10/10
1358/1358 ━━━━━━━━━━━━━━━━━━━━ 252s 185ms/step - accuracy: 0.8107 - loss: 0.5907 - val_accuracy: 0.8968 - v
```

Plot Training and Validation Results

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```





Model Evaluation Metrics

```
val_loss, val_accuracy = model.evaluate(val_data)
print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")
print(f"Validation Loss: {val_loss:.4f}")
```

```
340/340 ━━━━━━━━━━━━━━━━━━━━ 51s 148ms/step - accuracy: 0.8955 - loss: 0.3154
Validation Accuracy: 89.88%
Validation Loss: 0.3126
```

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
true_labels = val_data.classes
predictions = model.predict(val_data)
predicted_labels = np.argmax(predictions, axis=1)
```

```
class_names = list(val_data.class_indices.keys())
```

```
340/340 ━━━━━━━━━━━━━━━━━━━━ 62s 181ms/step
```

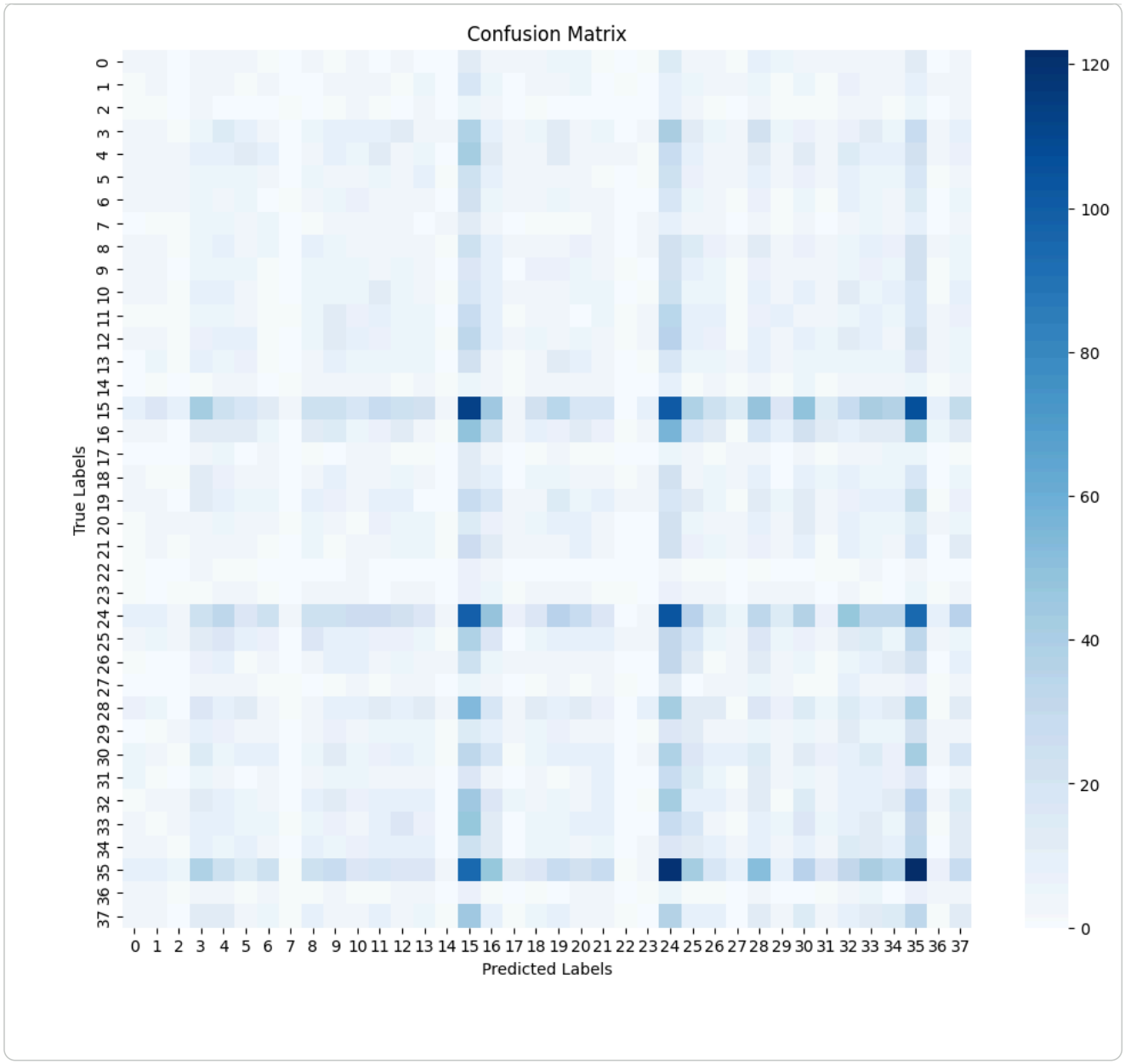
Classification Report

```
report = classification_report(true_labels, predicted_labels, target_names=class_names)
print("Classification Report:\n")
print(report)
```

```
Classification Report:

                                                precision    recall  f1-score   support

                                Apple___Apple_scab       0.03      0.02      0.03       126
                                 Apple___Black_rot       0.03      0.02      0.02       124
                          Apple___Cedar_apple_rust       0.00      0.00      0.00        55
                                   Apple___healthy       0.01      0.02      0.01       329
                                Blueberry___healthy       0.03      0.03      0.03       300
           Cherry_(including_sour)___Powdery_mildew       0.02      0.02      0.02       210
                  Cherry_(including_sour)___healthy       0.00      0.00      0.00       170
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.00      0.00      0.00       102
                         Corn_(maize)___Common_rust_       0.04      0.04      0.04       238
                Corn_(maize)___Northern_Leaf_Blight       0.02      0.03      0.02       197
                             Corn_(maize)___healthy       0.02      0.02      0.02       232
                                  Grape___Black_rot       0.04      0.04      0.04       236
                       Grape___Esca_(Black_Measles)       0.02      0.02      0.02       276
         Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       0.03      0.02      0.02       215
                                   Grape___healthy       0.03      0.01      0.02        84
           Orange___Haunglongbing_(Citrus_greening)       0.10      0.10      0.10      1101
                             Peach___Bacterial_spot       0.05      0.05      0.05       459
                                    Peach___healthy       0.00      0.00      0.00        72
                      Pepper,_bell___Bacterial_spot       0.03      0.03      0.03       199
                             Pepper,_bell___healthy       0.05      0.05      0.05       295
                              Potato___Early_blight       0.04      0.04      0.04       200
                               Potato___Late_blight       0.01      0.01      0.01       200
                                   Potato___healthy       0.00      0.00      0.00        30
                                Raspberry___healthy       0.00      0.00      0.00        74
                                  Soybean___healthy       0.09      0.10      0.10      1018
                            Squash___Powdery_mildew       0.05      0.05      0.05       367
                           Strawberry___Leaf_scorch       0.00      0.00      0.00       221
                               Strawberry___healthy       0.02      0.02      0.02        91
                             Tomato___Bacterial_spot       0.04      0.04      0.04       425
                               Tomato___Early_blight       0.02      0.01      0.02       200
                                Tomato___Late_blight       0.03      0.03      0.03       381
                                  Tomato___Leaf_Mold       0.02      0.02      0.02       190
                         Tomato___Septoria_leaf_spot       0.02      0.02      0.02       354
     Tomato___Spider_mites Two-spotted_spider_mite       0.03      0.04      0.03       335
                                Tomato___Target_Spot       0.03      0.03      0.03       280
              Tomato___Tomato_Yellow_Leaf_Curl_Virus       0.12      0.11      0.12      1071
                        Tomato___Tomato_mosaic_virus       0.03      0.03      0.03        74
                                    Tomato___healthy       0.04      0.04      0.04       318

                                            accuracy                           0.05     10849
                                           macro avg       0.03      0.03      0.03     10849
                                        weighted avg       0.05      0.05      0.05     10849
```

Confusion Matrix

```
cm = confusion_matrix(true_labels, predicted_labels)

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=False, cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```




```
# overall metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(true_labels, predicted_labels)
precision = precision_score(true_labels, predicted_labels, average='macro')
recall = recall_score(true_labels, predicted_labels, average='macro')
f1 = f1_score(true_labels, predicted_labels, average='macro')

print(f"Overall Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

```
Overall Accuracy: 5.12%
Precision: 0.03
Recall: 0.03
F1 Score: 0.03
```

Save the model

```
model.save("plant_disease_cnn.h5")
print("Model saved successfully!")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model
Model saved successfully!
```

Test the Model

Prediction

```
from tensorflow.keras.preprocessing import image
from google.colab import files
import matplotlib.pyplot as plt
import numpy as np

# --- Define prediction function ---
def predict_plant_disease(model, train_data):
    uploaded = files.upload()
    for img_name in uploaded.keys():
        img_path = img_name

        # Load and preprocess
        img = image.load_img(img_path, target_size=(128, 128))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0) / 255.0

        # Predict
        pred = model.predict(img_array)
        predicted_class = np.argmax(pred, axis=1)[0]
        confidence = np.max(pred) * 100
        class_labels = list(train_data.class_indices.keys())

        # Display image and prediction
        plt.figure(figsize=(4, 4))
        plt.imshow(image.load_img(img_path))
        plt.axis('off')
        plt.title(f"Prediction: {class_labels[predicted_class]}\nConfidence: {confidence:.2f}%", fontsize=
        plt.show()

        # Print result
        print(f"Predicted Class: {class_labels[predicted_class]}")
        print(f"Model Confidence: {confidence:.2f}%\n")


# Call the function
predict_plant_disease(model, train_data)
```

Choose Files Screenshot... 141331.png

Screenshot 2025-11-08 141331.png(image/png) - 308214 bytes, last modified: 8/11/2025 - 100% done

```
Saving Screenshot 2025-11-08 141331.png to Screenshot 2025-11-08 141331 (3).png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```

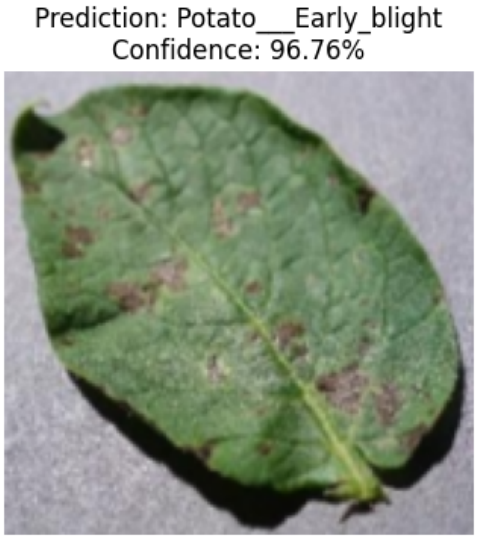

```
Predicted Class: Potato___Late_blight
Model Confidence: 80.53%
```

```
predict_plant_disease(model, train_data)
```

Choose Files Screenshot… 144227.png

Screenshot 2025-11-08 144227.png(image/png) - 55778 bytes, last modified: 8/11/2025 - 100% done

```
Saving Screenshot 2025-11-08 144227.png to Screenshot 2025-11-08 144227.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
```




```
Predicted Class: Potato___Early_blight
Model Confidence: 96.76%
```