```
!pip install google-api-python-client
!pip install oauth2client



!pip install oauth2client gspread google-api-python-client

import tensorflow as tf
print("TensorFlow version:", tf.__version__)
import gspread
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload

import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from oauth2client.service_account import ServiceAccountCredentials


# Path to the dataset
data_dir =
'C:/Users/MALAYAPPAN/OneDrive/Desktop/all/aiot/computervision/plantDisease_monitoring
/PlantVillage/PlantVillage'

# Parameters
img_size = 128  # You can adjust this size

# Function to load and preprocess images
def load_data(data_dir, img_size):
    images = []
    labels = []
    categories = os.listdir(data_dir)
    for category in categories:
        category_path = os.path.join(data_dir, category)
        if not os.path.isdir(category_path):
            continue
        for img in os.listdir(category_path):
            img_path = os.path.join(category_path, img)
            try:
                img_array = cv2.imread(img_path)
                if img_array is None:
                    print(f"Warning: Could not read image {img_path}")
                    continue
                resized_img = cv2.resize(img_array, (img_size, img_size))
                images.append(resized_img)
                labels.append(category)
            except Exception as e:
                print(f"Error processing image {img_path}: {e}")
                continue
    return np.array(images), np.array(labels)

# Load data
images, labels = load_data(data_dir, img_size)

# Encode labels
le = LabelEncoder()
labels_encoded = le.fit_transform(labels)

# Split data
X_train, X_test, y_train, y_test = train_test_split(images, labels_encoded, test_size=0.2,
random_state=42)

# Normalize images
X_train = X_train / 255.0
X_test = X_test / 255.0

print("Data loaded and preprocessed successfully.")
print(f"Number of training samples: {len(X_train)}")
print(f"Number of testing samples: {len(X_test)}")


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

# Build the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 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(len(np.unique(labels_encoded)), activation='softmax')
])

# Compile the model
```

```
model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Print the model summary
model.summary()


# Early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
history = model.fit(
    X_train, y_train,
    epochs=25,
    validation_data=(X_test, y_test),
    callbacks=[early_stopping]
)

 # Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc}")

# Save the model
model.save('plant_disease_model.h5')

def predict_disease(image_path, model):
    img = cv2.imread(image_path)
    img = cv2.resize(img, (img_size, img_size))
    img = img / 255.0
    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)
    predicted_class = np.argmax(prediction, axis=1)
    return le.inverse_transform(predicted_class)[0]

# Load the model
model = tf.keras.models.load_model('plant_disease_model.h5')

# Predict the disease
image_path =
r'C:\Users\MALAYAPPAN\OneDrive\Desktop\all\aiot\computervision\'Cercospora_capsici.jpg'
predicted_disease = predict_disease(image_path, model)
print(f"The predicted disease is: {predicted_disease}")

# Load the model
model = tf.keras.models.load_model('plant_disease_model.h5')

# Predict the disease
```

```
image_path =
r'C:\Users\MALAYAPPAN\OneDrive\Desktop\all\aiot\computervision\WhatsApp Image
2024-06-16 at 11.58.55_0ad205aa.jpg'
predicted_disease = predict_disease(image_path, model)
print(f"The predicted disease is: {predicted_disease}")


# Load the model
model = tf.keras.models.load_model('plant_disease_model.h5')

# Predict the disease
image_path = r'C:\Users\MALAYAPPAN\OneDrive\Pictures\banyan.jpg'
predicted_disease = predict_disease(image_path, model)
print(f"The predicted disease is: {predicted_disease}")


# Load the model
model = tf.keras.models.load_model('plant_disease_model.h5')

# Predict the disease
image_path = r'C:\Users\MALAYAPPAN\OneDrive\Pictures\bilwa.jpg'
predicted_disease = predict_disease(image_path, model)
print(f"The predicted disease is: {predicted_disease}")


def update_google_sheet(predicted_disease, sheet_name):
    # Define the scope and create credentials
    scope = ["https://spreadsheets.google.com/feeds",
"https://www.googleapis.com/auth/drive"]
    creds =
ServiceAccountCredentials.from_json_keyfile_name(r"C:\Users\MALAYAPPAN\plant-426909
-8c6454e8fafb.json", scope)
    client = gspread.authorize(creds)

    # Open the Google Sheet and select the first worksheet
    sheet = client.open(sheet_name).sheet1

    # Append the predicted disease to the sheet
    sheet.append_row([predicted_disease])

# Update the Google Sheet with the predicted disease
sheet_name = "live plant detection"
update_google_sheet(predicted_disease, sheet_name)
print("Google Sheet updated with the predicted disease.")
```