import tkinter as tk

from tkinter import filedialog

from PIL import Image, ImageTk # Make sure to install Pillowpackage

import os

from keras.models import load\_model

from PIL import Image

import tensorflow as tf

from tensorflow.keras.preprocessing.image import load\_img,

img\_to\_array

import cv2

import keras

from keras.preprocessing import image

import numpy as np

class BorerClassifierUI:

def \_init\_(self, root):

self.root = root

self.root.title("BORER LEAF CLASSIFICATION")

# Title label

title\_label = tk.Label(root, text="BORER LEAF

CLASSIFICATION", font=("Helvetica", 16, "bold"))

title\_label.pack(pady=10)

self.image\_path = tk.StringVar()

# Browse Button

self.browse\_button = tk.Button(root, text="Browse",

command=self.browse\_image)

self.browse\_button.pack(pady=10)

# Display selected image

self.image\_label = tk.Label(root)

self.image\_label.pack()

# Submit Button

self.submit\_button = tk.Button(root, text="Submit",

command=self.submit\_image)

self.submit\_button.pack(pady=10)

# Result Label

self.result\_label = tk.Label(root, text="", font=

("Helvetica", 12))

self.result\_label.pack(pady=10)

def browse\_image(self):

file\_path = filedialog.askopenfilename(title="Select Image",if file\_path:

self.image\_path.set(file\_path)

self.display\_image(file\_path)

def display\_image(self, file\_path):

image = Image.open(file\_path)

image = image.resize((300, 300)) # Resize the image to fit

the label photo = ImageTk.PhotoImage(image)

# Update label with the selected image

self.image\_label.config(image=photo)

self.image\_label.image = photo

def submit\_image(self):

if self.image\_path.get():

# Call your backend function with the image path

backend\_result = classify\_tomato(self.image\_path.get())

self.result\_label.config(text="Result: " +

backend\_result)

else:

self.result\_label.config(text="Please select an image

first.")

def classify\_tomato(image\_path):

target\_size = (224,224)

model=load\_model(’best\_model.h5’)

print("model loaded")

dic={0:’HEALTHY’,1:’POWDERY’,2:’RUST’

}

test\_image = load\_img(image\_path, target\_size = (224,224))

test\_image = img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

test\_image = test\_image/255

result = model.predict(test\_image)

print(np.exp(result))

print(np.argmax(result))

result = np.argmax(result)

detec=dic[result]

# Placeholder for the backend function

# You can replace this with your actual classification logic

return str(detec)

if \_name\_ == "\_main\_":

root = tk.Tk()

app = BorerClassifierUI(root)

root.mainloop()

The following is the code used for live detection of coffee plant

borer:

import cv2

import numpy as np

from keras.models import load\_model

from tensorflow.keras.preprocessing.image import img\_to\_array

# Load the trained model

model = load\_model(’best\_model.h5’)

print("Model loaded")

size = (224,224)

def classify\_tomato(image):

# Dictionary to map class indices to class labels

dic = {0: ’HEALTHY’, 1: ’POWDERY’, 2: ’RUST’}

image = cv2.resize(image, size)

# Normalize the image

test\_image = image / 255.0

# Add batch dimension

test\_image = np.expand\_dims(test\_image, axis=0)

# Perform classification

result = model.predict(test\_image)

# Get the predicted class index

result = np.argmax(result)

# Get the corresponding class label

detec = dic[result]

return str(detec)

# Function to detect leaf in an image

def detect\_leaf(image):

# Convert the image to HSV color space

hsv = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

# Define the range of green color in HSV

lower\_green = np.array([40, 40, 40])

upper\_green = np.array([80, 255, 255])

# Threshold the HSV image to get only green colors

mask = cv2.inRange(hsv, lower\_green, upper\_green)

# Bitwise-AND mask and original image

res = cv2.bitwise\_and(image, image, mask=mask)

# Convert the masked image to grayscale

gray = cv2.cvtColor(res, cv2.COLOR\_BGR2GRAY)

# Apply a threshold to binarize the grayscale image

\_, binary = cv2.threshold(gray, 50, 255, cv2.THRESH\_BINARY)

# Find contours in the binary image

contours, \_ = cv2.findContours(binary, cv2.RETR\_EXTERNAL, cv2.CHA# Check if any contour is found

if contours:

# Assuming the largest contour corresponds to the leaf

largest\_contour = max(contours, key=cv2.contourArea)

# Calculate the area of the contour

contour\_area = cv2.contourArea(largest\_contour)

# Define a threshold for the contour area (adjust as needed)

min\_contour\_area = 1000

# If the contour area is above the threshold, consider it asif contour\_area > min\_contour\_area:

return True

return False

# Main function to read from live camera and detect leaf

def main():

# Initialize the camera

cap = cv2.VideoCapture(0)

while True:

# Capture frame-by-frame

ret, frame = cap.read()

# Check if the frame is captured successfully

if not ret:

print("Failed to capture frame")

break

# Detect leaf in the frame

if detect\_leaf(frame):

print("Leaf detected")

# Classify the detected leaf

result = classify\_tomato(frame)

# Display the classification result on the frame

cv2.putText(frame, result, (10, 30), cv2.FONT\_HERSHEY\_SIMelse:

print("No leaf")

# Display the captured frame

cv2.imshow(’Frame’, frame)

# Check for key press

key = cv2.waitKey(1)

if key == ord(’q’):

break

# Release the camera and close OpenCV windows

cap.release()

cv2.destroyAllWindows()

if \_name\_ == "\_main\_":

main()