**Python for AI**

Name : Surya kiran

Reg.no : 20BCD7110

Github Repository Link : https://github.com/Suryakiran09/Python-for-AI---Surya-kiran.git

Code :

import os

import cv2

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the input and output directories for each class

class1\_input\_dir = "/content/drive/MyDrive/StudentProjects/CELOSIA ARGENTEA L"

class1\_output\_dir ="/content/drive/MyDrive/StudentProjects/celosia\_out"

class2\_input\_dir = "/content/drive/MyDrive/StudentProjects/CROWFOOT GRASS"

class2\_output\_dir = "/content/drive/MyDrive/StudentProjects/crow\_out"

class3\_input\_dir = "/content/drive/MyDrive/StudentProjects/PURPLE CHLORIS"

class3\_output\_dir = "/content/drive/MyDrive/StudentProjects/purplr\_out"

# Define the target image size

target\_size = (224, 224) # Change the size to whatever you need

# Preprocess images in class 1

for filename in os.listdir(class1\_input\_dir):

img = cv2.imread(os.path.join(class1\_input\_dir, filename))

resized\_img = cv2.resize(img, target\_size)

height, width = resized\_img.shape[:2]

start\_row, start\_col = int(height \* 0.25), int(width \* 0.25)

end\_row, end\_col = int(height \* 0.75), int(width \* 0.75)

cropped\_img = resized\_img[start\_row:end\_row, start\_col:end\_col]

normalized\_img = cropped\_img / 255.0

output\_filename = os.path.join(class1\_output\_dir, filename)

cv2.imwrite(output\_filename, normalized\_img)

# Preprocess images in class 2

for filename in os.listdir(class2\_input\_dir):

img = cv2.imread(os.path.join(class2\_input\_dir, filename))

resized\_img = cv2.resize(img, target\_size)

height, width = resized\_img.shape[:2]

start\_row, start\_col = int(height \* 0.25), int(width \* 0.25)

end\_row, end\_col = int(height \* 0.75), int(width \* 0.75)

cropped\_img = resized\_img[start\_row:end\_row, start\_col:end\_col]

normalized\_img = cropped\_img / 255.0

output\_filename = os.path.join(class2\_output\_dir, filename)

cv2.imwrite(output\_filename, normalized\_img)

# Preprocess images in class 3

for filename in os.listdir(class3\_input\_dir):

img = cv2.imread(os.path.join(class3\_input\_dir, filename))

resized\_img = cv2.resize(img, target\_size)

height, width = resized\_img.shape[:2]

start\_row, start\_col = int(height \* 0.25), int(width \* 0.25)

end\_row, end\_col = int(height \* 0.75), int(width \* 0.75)

cropped\_img = resized\_img[start\_row:end\_row, start\_col:end\_col]

normalized\_img = cropped\_img / 255.0

output\_filename = os.path.join(class3\_output\_dir, filename)

cv2.imwrite(output\_filename, normalized\_img)

# Define the directories for the dataset

train\_dir = '/content/drive/MyDrive/student\_output/celosia\_out-20230416T064611Z-001'

val\_dir = '/content/drive/MyDrive/student\_output/celosia\_out-20230416T064611Z-001'

test\_dir = '/content/drive/MyDrive/student\_output/celosia\_out-20230416T064611Z-001'

# Define the parameters for image resizing and batch size

img\_height = 224

img\_width = 224

batch\_size = 32

# Create data generators for training, validation, and test data

train\_datagen = ImageDataGenerator(rescale=1./255, shear\_range=0.2, zoom\_range=0.2, horizontal\_flip=True)

val\_datagen = ImageDataGenerator(rescale=1./255)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_data = train\_datagen.flow\_from\_directory(train\_dir, target\_size=(img\_height, img\_width), batch\_size=batch\_size, class\_mode='categorical')

val\_data = val\_datagen.flow\_from\_directory(val\_dir, target\_size=(img\_height, img\_width), batch\_size=batch\_size, class\_mode='categorical')

test\_data = test\_datagen.flow\_from\_directory(test\_dir, target\_size=(img\_height, img\_width), batch\_size=batch\_size, class\_mode='categorical')

# Define the model architecture

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(16, (3,3), activation='relu', input\_shape=(img\_height, img\_width, 3)),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(512, activation='relu'),

tf.keras.layers.Dense(3, activation='softmax')

])

# Compile the model with an appropriate loss and optimizer

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

# Train the model

history = model.fit(train\_data, epochs=25, validation\_data=val\_data)

# Evaluate the model on the test data

test\_loss, test\_acc = model.evaluate(test\_data)

print('Test accuracy:', test\_acc)

#save the model

model.save('weed\_classification\_model.h')

from pickle import NONE

import cv2

import numpy as np

import tensorflow as tf

# Load the trained model

model = tf.keras.models.load\_model('/content/weed\_classification\_model.h')

# Define the parameters for image resizing

img\_height = 224

img\_width = 224

# Define the lower and upper bounds for the color of the weed

lower = np.array([0, 0, 0])

upper = np.array([50, 50, 50])

# Open a video capture device

cap = cv2.VideoCapture(0)

while True:

# Capture the frame from the video stream

ret, frame = cap.read()

# Convert the frame to grayscale

if(frame is not None):

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

# Remove the background noise using the GaussianBlur function

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Threshold the image to binarize it

\_, thresh = cv2.threshold(blurred, 120, 255, cv2.THRESH\_BINARY)

# Apply a mask to the image to remove the background

mask = cv2.inRange(frame, lower, upper)

masked = cv2.bitwise\_and(thresh, thresh, mask=mask)

# Resize the image to match the input size of the trained model

resized = cv2.resize(masked, (img\_height, img\_width))

# Normalize the pixel values

normalized = resized / 255.0

# Add a batch dimension to the image

img = np.expand\_dims(normalized, axis=0)

# Use the trained model to predict the type of weed

predictions = model.predict(img)

class\_index = np.argmax(predictions[0])

if class\_index == 0:

label = "Weed Type A"

elif class\_index == 1:

label = "Weed Type B"

else:

label = "Weed Type C"

# Display the label on the frame

cv2.putText(frame, label, (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 255, 0), 2)

# Display the frame

cv2.imshow("Frame", frame)

# Press the 'q' key to exit the loop

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Release the video capture device and close all windows

cap.release()

cv2.destroyAllWindows()

import matplotlib.pyplot as plt

def plot\_hist(history):

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.plot(acc, label='Training Accuracy')

plt.plot(val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.grid()

plt.subplot(1, 2, 2)

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

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

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.grid()

plt.show()

plot\_hist(history)

Output :



