import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras import layers

from google.colab import drive

*# Mount Google Drive*

drive.mount('/content/drive')

*# Define batch specifications*

batch\_size = 32 *# You can adjust this based on your preference*

img\_height, img\_width = 224, 224 *# Adjust these dimensions based on your images*

*# Define paths to the training and validation datasets*

train\_dataset\_path = '/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)/train'

valid\_dataset\_path = '/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)/valid'

*# Load and preprocess the training dataset with error handling*

training\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

train\_dataset\_path,

seed=42,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size,

label\_mode='int', *# Use 'int' labels for sparse categorical\_crossentropy*

validation\_split=0.2,

subset='training',

smart\_resize=True *# Resize the images preserving their aspect ratio*

)

*# Load and preprocess the validation dataset with error handling*

validation\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

valid\_dataset\_path,

seed=42,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size,

label\_mode='int', *# Use 'int' labels for sparse categorical\_crossentropy*

validation\_split=0.2,

subset='validation',

smart\_resize=True *# Resize the images preserving their aspect ratio*

)

*# Get class names from the training dataset*

class\_names = training\_ds.class\_names

*# Define the CNN model*

MyCnn = tf.keras.models.Sequential([

layers.BatchNormalization(),

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

layers.MaxPooling2D(),

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

layers.MaxPooling2D(),

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

layers.MaxPooling2D(),

layers.Flatten(),

layers.Dense(256, activation='relu'),

layers.Dense(len(class\_names), activation='softmax')

])

*# Compile the model*

MyCnn.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

*# Train the model*

retVal = MyCnn.fit(training\_ds, validation\_data=validation\_ds, epochs=2)

--

plt.plot(retVal.history['loss'], label = 'training loss')

plt.plot(retVal.history['accuracy'], label = 'training accuracy')

plt.legend()

--

AccuracyVector = []

plt.figure(figsize=(30, 30))

*# Limit the loop to the number of images in the batch*

num\_images\_to\_visualize = min(40, validation\_ds.cardinality().numpy() \* batch\_size)

for images, labels in validation\_ds.take(1):

predictions = MyCnn.predict(images)

predlabel = []

prdlbl = []

for mem in predictions:

predlabel.append(class\_names[np.argmax(mem)])

prdlbl.append(np.argmax(mem))

AccuracyVector = np.array(prdlbl) == labels

*# Print the shape of the 'images' tensor*

print("Shape of 'images' tensor:", images.shape)

*# Limit the loop to the actual number of images in the batch*

for i in range(min(num\_images\_to\_visualize, images.shape[0])):

ax = plt.subplot(10, 4, i + 1)

plt.imshow(images[i].numpy().astype("uint8"))

plt.title('Pred: '+ predlabel[i]+' actl:'+class\_names[labels[i]] )

plt.axis('off')

plt.grid(True)

plt.show()

--

plt.plot(retVal.history['val\_loss'], label = 'validation loss')

plt.plot(retVal.history['val\_accuracy'], label = 'validation accuracy')

plt.legend()