#leaf.py

#Import necessary libraries

from flask import Flask, render\_template, request

import numpy as np

import os

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

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

from tensorflow.keras.models import load\_model

filepath = 'C:\\Users\\saisw\\OneDrive\\Desktop\\Plant\_Leaf\_Disease\_Detection\\model.h5'

model = load\_model(filepath)

print(model)

print("Model Loaded Successfully")

def pred\_tomato\_dieas(tomato\_plant):

test\_image = load\_img(tomato\_plant, target\_size = (128, 128)) # load image

print("@@ Got Image for prediction")

test\_image = img\_to\_array(test\_image)/255 # convert image to np array and normalize

test\_image = np.expand\_dims(test\_image, axis = 0) # change dimention 3D to 4D

result = model.predict(test\_image) # predict diseased palnt or not

print('@@ Raw result = ', result)

pred = np.argmax(result, axis=1)

print(pred)

if pred==0:

return "Tomato - Bacteria Spot Disease", 'Tomato-Bacteria Spot.html'

elif pred==1:

return "Tomato - Early Blight Disease", 'Tomato-Early\_Blight.html'

elif pred==2:

return "Tomato - Healthy and Fresh", 'Tomato-Healthy.html'

elif pred==3:

return "Tomato - Late Blight Disease", 'Tomato - Late\_blight.html'

elif pred==4:

return "Tomato - Leaf Mold Disease", 'Tomato - Leaf\_Mold.html'

elif pred==5:

return "Tomato - Septoria Leaf Spot Disease", 'Tomato - Septoria\_leaf\_spot.html'

elif pred==6:

return "Tomato - Target Spot Disease", 'Tomato - Target\_Spot.html'

elif pred==7:

return "Tomato - Tomoato Yellow Leaf Curl Virus Disease", 'Tomato - Tomato\_Yellow\_Leaf\_Curl\_Virus.html'

elif pred==8:

return "Tomato - Tomato Mosaic Virus Disease", 'Tomato - Tomato\_mosaic\_virus.html'

elif pred==9:

return "Tomato - Two Spotted Spider Mite Disease", 'Tomato - Two-spotted\_spider\_mite.html'

# Create flask instance

app = Flask(\_\_name\_\_)

# render index.html page

@app.route("/", methods=['GET', 'POST'])

def home():

return render\_template('index.html')

# get input image from client then predict class and render respective .html page for solution

@app.route("/predict", methods = ['GET','POST'])

def predict():

if request.method == 'POST':

file = request.files['image'] # fet input

filename = file.filename

print("@@ Input posted = ", filename)

file\_path = os.path.join('C:\\Users\\saisw\\OneDrive\\Desktop\\Plant\_Leaf\_Disease\_Detection\\static\\upload', filename)

file.save(file\_path)

print("@@ Predicting class......")

pred, output\_page = pred\_tomato\_dieas(tomato\_plant=file\_path)

return render\_template(output\_page, pred\_output = pred, user\_image = file\_path)

# For local system & cloud

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

app.run(threaded=False,port=8081)

#training.py

from tensorflow.compat.v1 import ConfigProto

from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()

config.gpu\_options.allow\_growth = True

session = InteractiveSession(config=config)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import tensorflow as tf

tf.compat.v1.disable\_eager\_execution()

import matplotlib.pyplot as plt

import numpy as np

import os

#basic cnn

# Initialising the CNN

classifier = Sequential()

# Step 1 - Convolution

classifier.add(Conv2D(32, (3, 3), input\_shape = (128, 128, 3), activation = 'relu'))

# Step 2 - Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a second convolutional layer

classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 10, activation = 'sigmoid'))

# Compiling the CNN

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

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

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

training\_set = train\_datagen.flow\_from\_directory('C:\\Users\\saisw\\OneDrive\\Desktop\\Plant\_Leaf\_Disease\_Detection\\Dataset\\train', # relative path from working directoy

target\_size = (128, 128),

batch\_size = 6, class\_mode = 'categorical')

valid\_set = test\_datagen.flow\_from\_directory('C:\\Users\\saisw\\OneDrive\\Desktop\\Plant\_Leaf\_Disease\_Detection\\Dataset\\val', # relative path from working directoy

target\_size = (128, 128),

batch\_size = 3, class\_mode = 'categorical')

labels = (training\_set.class\_indices)

print(labels)

classifier.fit\_generator(training\_set,

steps\_per\_epoch = 20,

epochs = 50,

validation\_data=valid\_set

)

classifier\_json=classifier.to\_json()

with open("model1.json", "w") as json\_file:

json\_file.write(classifier\_json)

# serialize weights to HDF5

classifier.save\_weights("my\_model\_weights.h5")

classifier.save("model.h5")

print("Saved model to disk")

'''

import cv2

from matplotlib import pyplot as plt

import os

import numpy as np

img = cv2.imread('C:/Users//AppData/Local/Programs/Python/Python38/Leaf\_disease/data/d (7)\_iaip.jpg')

img\_resize = cv2.resize(img, (128,128))

CV2 reads an image in BGR format. We need to convert it to RGB

b,g,r = cv2.split(img\_resize) # get b,g,r

rgb\_img = cv2.merge([r,g,b]) # switch it to rgb

plt.imshow(rgb\_img)

label\_map = (training\_set.class\_indices)

print(label\_map)

img\_rank4 = np.expand\_dims(rgb\_img/255, axis=0)

classifier.predict(img\_rank4)

h = list(label\_map.keys())[classifier.predict\_classes(img\_rank4)[0]]

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(img, h, (10, 30), font, 1.0, (0, 0, 255), 1)

cv2.imshow(h,img)

print(h)

'''