from yoloDetection import detectObject, displayImage

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from tkinter import simpledialog

from tkinter import filedialog

import os

import cv2

import numpy as np

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten, LSTM

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model\_from\_json

import pickle

main = tkinter.Tk()

main.title("FACIAL EXPRESSIONS IDENTIFICATION BY APPLYING KERAS ") #designing main screen

main. Geometry("1300x1200")

global filename

global X, Y

global classifier

global cnn\_model

global cnn\_layer\_names

names = ['angry','disgusted','fearful','happy','neutral','sad','surprised']

def loadLibraries(): #function to load yolov3 model weight and class labels

global class\_labels

global cnn\_model

global cnn\_layer\_names

class\_labels = open('yolov3model/yolov3-labels').read().strip().split('\n') #reading labels from yolov3 model

cnn\_model = cv2.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights') #reading model

cnn\_layer\_names = cnn\_model.getLayerNames() #getting layers from cnn model

cnn\_layer\_names = [cnn\_layer\_names[i[0] - 1] for i in cnn\_model.getUnconnectedOutLayers()] #assigning all layers

def getID(name):

index = 0

for i in range(len(names)):

if names[i] == name:

index = i

break

return index

def upload():

global filename

filename = filedialog.askopenfilename(initialdir="model")

text.delete('1.0', END)

text.insert(END,filename+" loaded\n");

def yolofaceDetect(image): #function to detect face from images.

#random colors to assign unique color to each label

label\_colors = np.random.randint(0,255,size=(len(class\_labels),3),dtype='uint8')

try:

image\_height, image\_width = image.shape[:2] #converting image to two-dimensional array

except:

raise 'Invalid image path'

finally:

image, \_, \_, \_, \_ = detectObject(cnn\_model, cnn\_layer\_names, image\_height, image\_width, image, label\_colors, class\_labels)#calling detection function

#displayImage(image)#display image with detected objects label

def processDataset():

text.delete('1.0', END)

global X, Y

'''

X = []

Y = []

for root, dirs, directory in os.walk(filename):

for j in range(len(directory)):

name = os.path.basename(root)

print(name+" "+root+"/"+directory[j])

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j])

img = cv2.resize(img, (32,32))

im2arr = np.array(img)

im2arr = im2arr.reshape(32,32,3)

X.append(im2arr)

Y.append(getID(name))

X = np.asarray(X)

Y = np.asarray(Y)

print(Y)

X = X.astype('float32')

X = X/255

test = X[3]

test = cv2.resize(test,(400,400))

cv2.imshow("aa",test)

cv2.waitKey(0)

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

np.save('model/X.txt',X)

np.save('model/Y.txt',Y)

'''

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

yolofaceDetect(X[0])

text.insert(END,"Total number of YOLO Face images found in dataset is : "+str(len(X))+"\n")

text.insert(END,"Total classes found in dataset is : "+str(names)+"\n")

def trainCNN():

global classifier

text.delete('1.0', END)

if os.path.exists('model/cnnmodel.json'):

with open('model/cnnmodel.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

classifier = model\_from\_json(loaded\_model\_json)

classifier.load\_weights("model/cnnmodel\_weights.h5")

classifier.\_make\_predict\_function()

print(classifier.summary())

f = open('model/cnnhistory.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[19] \* 100

text.insert(END,"CNN YOLO Faces Training Model Accuracy = "+str(accuracy))

else:

classifier = Sequential()

classifier.add(Convolution2D(32, 3, 3, input\_shape = (32, 32, 3), activation = 'relu'))

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

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

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

classifier.add(Flatten())

classifier.add(Dense(output\_dim = 256, activation = 'relu'))

classifier.add(Dense(output\_dim = 7, activation = 'softmax'))

print(classifier.summary())

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

hist = classifier.fit(X, Y, batch\_size=32, epochs=20, shuffle=True, verbose=2)

classifier.save\_weights('model/cnnmodel\_weights.h5')

model\_json = classifier.to\_json()

with open("model/cnnmodel.json", "w") as json\_file:

json\_file.write(model\_json)

f = open('model/cnnhistory.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model/cnnhistory.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[19] \* 100

text.insert(END,"CNN YOLO Faces Training Model Accuracy = "+str(accuracy))

def predict():

filename = filedialog.askopenfilename(initialdir="testImages")

image = cv2.imread(filename)

img = cv2.resize(image, (32,32))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,32,32,3)

img = np.asarray(im2arr)

img = img.astype('float32')

img = img/255

preds = classifier.predict(img)

predict = np.argmax(preds)

img = cv2.imread(filename)

img = cv2.resize(img, (600,400))

cv2.putText(img, 'Facial Expression Recognized as : '+names[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (255, 0, 0), 2)

cv2.imshow('Facial Expression Recognized as : '+names[predict], img)

cv2.waitKey(0)

def graph():

f = open('model/cnnhistory.pckl', 'rb')

cnn\_data = pickle.load(f)

f.close()

cnn\_accuracy = cnn\_data['accuracy']

acc = []

for i in range(len(cnn\_accuracy)):

if i > 9:

acc.append(cnn\_accuracy[i])

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('Iterations/Epoch')

plt.ylabel('Accuracy')

plt.plot(acc, 'ro-', color = 'green')

plt.legend(['CNN YOLO Accuracy'], loc='upper left')

#plt.xticks(wordloss.index)

plt.title('CNN with YOLO Accuracy Graph')

plt.show()

def exit():

main.destroy()

font = ('times', 13, 'bold')

title = Label(main, text='FACIAL EXPRESSIONS IDENTIFICATION BY APPLYING KERAS ')

title.config(bg='LightGoldenrod1', fg='medium orchid')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=100)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=480,y=100)

text.config(font=font1)

font1 = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Facial Emotion Dataset", command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess Dataset", command=processDataset)

processButton.place(x=50,y=150)

processButton.config(font=font1)

cnnButton = Button(main, text="Train CNN Algorithm with YOLO Faces", command=trainCNN)

cnnButton.place(x=50,y=200)

cnnButton.config(font=font1)

graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)

graphButton.place(x=50,y=250)

graphButton.config(font=font1)

predictButton = Button(main, text="Predict Facial Expression", command=predict)

predictButton.place(x=50,y=300)

predictButton.config(font=font1)

exitButton = Button(main, text="Exit", command=exit)

exitButton.place(x=50,y=350)

exitButton.config(font=font1)

loadLibraries()

main.config(bg='OliveDrab2')

main.mainloop()