######## face\_recognition\_knn.py 🡨 file name

import math

from sklearn import neighbors

import os

import os.path

import pickle

from PIL import Image

import face\_recognition

from face\_recognition.face\_recognition\_cli import image\_files\_in\_folder

import cv2

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg', 'JPG', 'JPEG', 'PNG'}

def train(train\_dir, model\_save\_path=None, n\_neighbors=None, knn\_algo='ball\_tree', verbose=True):

X = []

y = []

# Loop through each person in the training set

for class\_dir in os.listdir(train\_dir):

if not os.path.isdir(os.path.join(train\_dir, class\_dir)):

continue

# Loop through each training image for the current person

n = 0

for img\_path in image\_files\_in\_folder(os.path.join(train\_dir, class\_dir)):

image = face\_recognition.load\_image\_file(img\_path)

img = Image.open(img\_path) ######### OWN

face\_bounding\_boxes=[(0,img.width,img.height,0)] ######## OWN

# Add face encoding for current image to the training set

X.append(face\_recognition.face\_encodings(image, known\_face\_locations=face\_bounding\_boxes)[0])

y.append(class\_dir)

n = n+1

if verbose:

print(class\_dir, f"{n} DONE ")

if n == 34:

break

# Determine how many neighbors to use for weighting in the KNN classifier

if n\_neighbors is None:

n\_neighbors = int(round(math.sqrt(len(X))))

if verbose:

print("Chose n\_neighbors automatically:", n\_neighbors)

# Create and train the KNN classifier

knn\_clf = neighbors.KNeighborsClassifier(n\_neighbors=n\_neighbors, algorithm=knn\_algo, weights='distance')

knn\_clf.fit(X, y)

# Save the trained KNN classifier

if model\_save\_path is not None:

with open(model\_save\_path, 'wb') as f:

pickle.dump(knn\_clf, f)

return knn\_clf

def predict(X\_img\_path, knn\_clf=None, model\_path=None, distance\_threshold=0.45):

if not os.path.isfile(X\_img\_path) or os.path.splitext(X\_img\_path)[1][1:] not in ALLOWED\_EXTENSIONS:

raise Exception("Invalid image path: {}".format(X\_img\_path))

if knn\_clf is None and model\_path is None:

raise Exception("Must supply knn classifier either thourgh knn\_clf or model\_path")

# Load a trained KNN model (if one was passed in)

if knn\_clf is None:

with open(model\_path, 'rb') as f:

knn\_clf = pickle.load(f)

# Load image file and find face locations

X\_img = face\_recognition.load\_image\_file(X\_img\_path)

X\_face\_locations = face\_recognition.face\_locations(X\_img)

# If no faces are found in the image, return an empty result.

if len(X\_face\_locations) == 0:

return []

# Find encodings for faces in the test iamge

faces\_encodings = face\_recognition.face\_encodings(X\_img, known\_face\_locations=X\_face\_locations)

# Use the KNN model to find the best matches for the test face

closest\_distances = knn\_clf.kneighbors(faces\_encodings, n\_neighbors=1)

are\_matches = [closest\_distances[0][i][0] <= distance\_threshold for i in range(len(X\_face\_locations))]

# Predict classes and remove classifications that aren't within the threshold

predictions =[]

for pred, loc, rec in zip(knn\_clf.predict(faces\_encodings), X\_face\_locations, are\_matches):

if rec:

predictions.append((pred, loc))

else:

predictions.append(("---------", loc))

return predictions

#return [(pred, loc) if rec else ("unknown", loc) for pred, loc, rec in zip(knn\_clf.predict(faces\_encodings), X\_face\_locations, are\_matches)]

def show\_prediction\_labels\_on\_image(img\_path, predictions, img\_save\_path, n):

image = cv2.imread(img\_path)

for name, (top, right, bottom, left) in predictions:

cv2.putText(image, name, (left, top-10), cv2.FONT\_HERSHEY\_SIMPLEX,fontScale=1, color=[0, 0, 0], thickness=8)

cv2.putText(image, name, (left, top-10), cv2.FONT\_HERSHEY\_SIMPLEX,fontScale=1, color=[255, 255, 255], thickness=3)

cv2.rectangle(image, (left, top), (right, bottom), (0, 0, 0), 6)

cv2.rectangle(image, (left, top), (right, bottom), (225, 225, 225), 4)

status = cv2.imwrite(img\_save\_path + "image\_" + str(n) + '.jpg', image)

print(status)

##### train.py 🡨 file name

import face\_recognition\_knn as fr

# STEP 1: Train the KNN classifier and save it to disk

# Once the model is trained and saved, you can skip this step next time.

print("Training KNN classifier...")

train\_dir\_path = "G:/Dataset/Realtime\_only\_faces"

save\_path = "G:/KNN Face\_Recognisation/Trained\_models/trained\_knn\_model\_13.clf"

classifier = fr.train(train\_dir\_path, model\_save\_path=save\_path, n\_neighbors=4)

print("Training complete!")

###### predict.py 🡨 file name

import face\_recognition\_knn as fr

from datetime import datetime

import os

import pandas as pd

all\_images\_present = []

# STEP 2: Using the trained classifier, make predictions for unknown images

n=0

today\_dir = "G:/KNN Face\_Recognisation/output excel and images/" + datetime.now().strftime("%d-%m-%Y")

time\_now\_dir = today\_dir + "/At\_Time\_" + datetime.now().strftime("%H\_%M\_%S")

os.makedirs(time\_now\_dir)

indvidual\_out\_images\_path = time\_now\_dir + "/" + "individual\_images\_output/"

os.makedirs(indvidual\_out\_images\_path)

indvidual\_out\_excel\_path = time\_now\_dir + "/" + "individual\_excel\_output/"

os.makedirs(indvidual\_out\_excel\_path)

for image\_file in os.listdir("G:/KNN Face\_Recognisation/Testing\_Images/Image\_one"):

full\_file\_path = os.path.join("G:/KNN Face\_Recognisation/Testing\_Images/Image\_one/", image\_file)

print("Looking for faces in {}".format(image\_file))

# Find all people in the image using a trained classifier model

# Note: You can pass in either a classifier file name or a classifier model instance

model\_path = 'G:/KNN Face\_Recognisation/Trained\_models/trained\_knn\_model\_13.clf'

predictions = fr.predict(full\_file\_path, model\_path = model\_path, distance\_threshold = 0.47)

# Print results on the console

single\_img\_present = []

for name, (top, right, bottom, left) in predictions:

#print("- Found {} at ({}, {})".format(name, left, top))

single\_img\_present.append(name)

print(image\_file + "\_complete")

# Display results overlaid on an image

#img\_save\_path = "G:/KNN Face\_Recognisation/output/model12-3/"

n+=1

fr.show\_prediction\_labels\_on\_image(full\_file\_path, predictions, indvidual\_out\_images\_path, n)

single\_img\_present = sorted(list(set(single\_img\_present)))

if(single\_img\_present[0]=="---------"):

single\_img\_present = single\_img\_present[1:]

df = pd.DataFrame(single\_img\_present)

all\_images\_present += single\_img\_present

df.to\_csv(indvidual\_out\_excel\_path + "excel\_of\_image\_" +str(n) + ".csv")

all\_images\_present = sorted(list(set(all\_images\_present)))

tdf = pd.DataFrame(all\_images\_present)

tdf.columns=['Attended\_Students']

#csv\_save\_path = "G:/KNN Face\_Recognisation/Excel\_output/total\_attendence\_report.csv"

tdf.to\_csv(time\_now\_dir+ "/" + "Total\_attendees\_report.csv")