import sys

import argparse

import cv2

from libfaceid.detector import FaceDetectorModels, FaceDetector

from libfaceid.encoder import FaceEncoderModels, FaceEncoder

from libfaceid.pose import FacePoseEstimatorModels, FacePoseEstimator

from libfaceid.age import FaceAgeEstimatorModels, FaceAgeEstimator

from libfaceid.gender import FaceGenderEstimatorModels, FaceGenderEstimator

from libfaceid.emotion import FaceEmotionEstimatorModels, FaceEmotionEstimator

# Use flask for web app

from flask import Flask, render\_template, Response

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

# Set the input directories

INPUT\_DIR\_DATASET = "datasets"

INPUT\_DIR\_MODEL\_DETECTION = "models/detection/"

INPUT\_DIR\_MODEL\_ENCODING = "models/encoding/"

INPUT\_DIR\_MODEL\_TRAINING = "models/training/"

INPUT\_DIR\_MODEL\_ESTIMATION = "models/estimation/"

# Set width and height

RESOLUTION\_QVGA = (320, 240)

RESOLUTION\_VGA = (640, 480)

RESOLUTION\_HD = (1280, 720)

RESOLUTION\_FULLHD = (1920, 1080)

def cam\_init(cam\_index, width, height):

cap = cv2.VideoCapture(cam\_index)

if sys.version\_info < (3, 0):

cap.set(cv2.cv.CV\_CAP\_PROP\_FPS, 60)

cap.set(cv2.cv.CV\_CAP\_PROP\_FRAME\_WIDTH, width)

cap.set(cv2.cv.CV\_CAP\_PROP\_FRAME\_HEIGHT, height)

else:

cap.set(cv2.CAP\_PROP\_FPS, 60)

cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, width)

cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, height)

return cap

def label\_face(frame, face\_rect, face\_id, confidence):

(x, y, w, h) = face\_rect

cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 255, 255), 1)

if face\_id is not None:

cv2.putText(frame, "{} {:.2f}%".format(face\_id, confidence),

(x+5,y+h-5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE\_AA)

def process\_facedetection():

cam\_index = 0

cam\_resolution = RESOLUTION\_QVGA

model\_detector=FaceDetectorModels.HAARCASCADE

model\_poseestimator=FacePoseEstimatorModels.DEFAULT

model\_ageestimator=FaceAgeEstimatorModels.DEFAULT

model\_genderestimator=FaceGenderEstimatorModels.DEFAULT

model\_emotionestimator=FaceEmotionEstimatorModels.DEFAULT

# Initialize the camera

camera = cam\_init(cam\_index, cam\_resolution[0], cam\_resolution[1])

try:

# Initialize face detection

face\_detector = FaceDetector(model=model\_detector, path=INPUT\_DIR\_MODEL\_DETECTION)#, optimize=True)

# Initialize face pose/age/gender estimation

face\_pose\_estimator = FacePoseEstimator(model=model\_poseestimator, path=INPUT\_DIR\_MODEL\_ESTIMATION)

face\_age\_estimator = FaceAgeEstimator(model=model\_ageestimator, path=INPUT\_DIR\_MODEL\_ESTIMATION)

face\_gender\_estimator = FaceGenderEstimator(model=model\_genderestimator, path=INPUT\_DIR\_MODEL\_ESTIMATION)

face\_emotion\_estimator = FaceEmotionEstimator(model=model\_emotionestimator, path=INPUT\_DIR\_MODEL\_ESTIMATION)

except:

print("Warning, check if models and trained dataset models exists!")

(age, gender, emotion) = (None, None, None)

while (True):

# Capture frame from webcam

ret, frame = camera.read()

if frame is None:

print("Error, check if camera is connected!")

break

# Detect and identify faces in the frame

faces = face\_detector.detect(frame)

for (index, face) in enumerate(faces):

(x, y, w, h) = face

# Detect age, gender, emotion

face\_image = frame[y:y+h, h:h+w]

age = face\_age\_estimator.estimate(frame, face\_image)

gender = face\_gender\_estimator.estimate(frame, face\_image)

emotion = face\_emotion\_estimator.estimate(frame, face\_image)

# Detect and draw face pose locations

shape = face\_pose\_estimator.detect(frame, face)

face\_pose\_estimator.add\_overlay(frame, shape)

# Display age, gender, emotion

cv2.putText(frame, "Age: {}".format(age),

(x, y-45), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE\_AA)

cv2.putText(frame, "Gender: {}".format(gender),

(x, y-30), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE\_AA)

cv2.putText(frame, "Emotion: {}".format(emotion),

(x, y-15), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE\_AA)

# Display updated frame to web app

yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' + cv2.imencode('.jpg', frame)[1].tobytes() + b'\r\n\r\n')

# Release the camera

camera.release()

cv2.destroyAllWindows()

# Initialize for web app

@app.route('/')

def index():

return render\_template('index.html')

# Entry point for web app

@app.route('/video\_viewer')

def video\_viewer():

return Response(process\_facedetection(), mimetype='multipart/x-mixed-replace; boundary=frame')

if \_\_name\_\_ == '\_\_main\_\_':

# Run flask for web app

app.run(threaded=True, debug=True)