# import libraries

From imutils import paths

Import numpy as np

Import argparse

Import imutils

Import pickle

Import cv2

Import os

# load serialized face detector

Print(“Loading Face Detector…”)

protoPath = “face\_detection\_model/deploy.prototxt”

modelPath = “face\_detection\_model/res10\_300x300\_ssd\_iter\_140000.caffemodel”

detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)

# load serialized face embedding model

Print(“Loading Face Recognizer…”)

Embedder = cv2.dnn.readNetFromTorch(“openface\_nn4.small2.v1.t7”)

# grab the paths to the input images in our dataset

Print(“Quantifying Faces…”)

imagePaths = list(paths.list\_images(“dataset”))

# initialize our lists of extracted facial embeddings and corresponding people names

knownEmbeddings = []

knownNames = []

# initialize the total number of faces processed

Total = 0

# loop over the image paths

For (I, imagePath) in enumerate(imagePaths):

# extract the person name from the image path

If (i%50 == 0):

Print(“Processing image {}/{}”.format(I, len(imagePaths)))

Name = imagePath.split(os.path.sep)[-2]

# load the image, resize it to have a width of 600 pixels (while maintaining the aspect ratio), and then grab the image dimensions

Image = cv2.imread(imagePath)

Image = imutils.resize(image, width=600)

(h, w) = image.shape[:2]

# construct a blob from the image

imageBlob = cv2.dnn.blobFromImage(

cv2.resize(image, (300, 300)), 1.0, (300, 300),

(104.0, 177.0, 123.0), swapRB=False, crop=False)

# apply OpenCV’s deep learning-based face detector to localize faces in the input image

Detector.setInput(imageBlob)

Detections = detector.forward()

# ensure at least one face was found

If len(detections) > 0:

# we’re making the assumption that each image has only ONE face, so find the bounding box with the largest probability

I = np.argmax(detections[0, 0, :, 2])

Confidence = detections[0, 0, I, 2]

# ensure that the detection with the largest probability also means our minimum probability test (thus helping filter out weak detections)

If confidence > 0.5:

# compute the (x, y)-coordinates of the bounding box for the face

Box = detections[0, 0, I, 3:7] \* np.array([w, h, w, h])

(startX, startY, endX, endY) = box.astype(“int”)

# extract the face ROI and grab the ROI dimensions

Face = image[startY:endY, startX:endX]

(fH, fW) = face.shape[:2]

# ensure the face width and height are sufficiently large

If fW < 20 or fH < 20:

Continue

# construct a blob for the face ROI, then pass the blob through our face embedding model to obtain the 128-d quantification of the face

faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,

(96, 96), (0, 0, 0), swapRB=True, crop=False)

Embedder.setInput(faceBlob)

Vec = embedder.forward()

# add the name of the person + corresponding face embedding to their respective lists

knownNames.append(name)

knownEmbeddings.append(vec.flatten())

total += 1

# dump the facial embeddings + names to disk

Print(“[INFO] serializing {} encodings…”.format(total))

Data = {“embeddings”: knownEmbeddings, “names”: knownNames}

F = open(“output/embeddings.pickle”, “wb”)

f.write(pickle.dumps(data))

f.close()