Image Classifier

1 Collect the Image

2 Import main library like

Numpy

Matplotlib

Cv2

3 Load the image with cv2

Image will be in cv2 color format

Img = cv2.imread(image)

4 convert image to gray

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

5 check the image with plot imshow and the array in gray

6 load the cascade for face and eye

face\_cascade = cv2.CascadeClassifier('../../../opencv/haarcascades/haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('../../../opencv/haarcascades/haarcascade\_eye.xml')

7 define face in image by the help of face cascade

faces = face\_cascade.detectMultiScale(gray,1.3,5)

8. define variable of face as x,y,w,h

(x,y,w,h) = faces[0]

Where x = x axix

y= y axix

w= width

h = height

9 draw the shape in image so that to mark the face as per value given by cascade

face\_img = cv2.rectangle(img,(x,y),(x+w,y+h),(230,44,100),2)

10 now lets define the Eye with the help of eye cascade

The return value for I will also in list as there are 2 eyes.

cv2.destroyAllWindows() for clearing windows

for (x,y,w,h) in faces: for return values of faces cascades

face\_img = cv2.rectangle(img,(x,y),(x+w,y+h),(230,44,100),2) draw the red line in faces

roi\_gray = gray[y:y+h, x:x+w] mark area in gray image

roi\_color = face\_img[y:y+h, x:x+w] mark the area in color image

eyes = eye\_cascade.detectMultiScale(roi\_gray) eye cascade

for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2) draw rectangle for eyes

plt.figure()

plt.imshow(face\_img, cmap='gray')

plt.show()

[ roi = region of interested ] >>>>to identify the faces

Now lets create a function which will satisfy all the above steps and gives us the complete solution

def cropped\_face\_2\_eyes(impath):

my\_image = cv2.imread(impath)

gray = cv2.cvtColor(my\_image,cv2.COLOR\_BGR2GRAY)

face\_cascade = cv2.CascadeClassifier('../../../opencv/haarcascades/haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('../../../opencv/haarcascades/haarcascade\_eye.xml')

faces = face\_cascade.detectMultiScale(gray,1.3,5)

for (x,y,w,h) in faces:

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = my\_image[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

if len(eyes) >= 2:

plt.imshow(roi\_color)

11 Data Cleaning

Now lets clean our data. Lets make crop image in a cropped folder for the each use and keep record of user and their files in dictionary form.

File = xyz/cropped/user/image…png

Create cropped folder in each file

Person\_name\_file\_list = {

‘person’:[path\_personA\_pic1,’path\_personA\_pic2’]

‘person’:[path\_personB\_pic2,path\_personB\_pic2]

}

Code Snipped:

cropped\_image\_dir = []

person\_name\_file\_list = {}

dataset = './dataset/'

mylist = os.listdir(dataset)

mylist

for i in mylist:

person\_name\_file\_list[i]=[]

if os.path.isdir(dataset+i):

person = dataset+i

if os.scandir(person):

for j in os.listdir(person):

pic = person+'/'+j

if os.path.isdir(pic+'/'):

pass

else:

cropped\_face = cropped\_face\_2\_eyes(pic)

if cropped\_face is not None:

mypath = dataset+'cropped/'+i

if not os.path.exists(mypath):

os.makedirs(mypath)

cropped\_file\_name = i+'\_'+j+'.png'

cropped\_file\_name\_path = mypath + '/' + cropped\_file\_name

cv2.imwrite(cropped\_file\_name\_path, cropped\_face)

print(i,cropped)

person\_name\_file\_list[i].append(cropped\_file\_name\_path)

Person\_name\_file\_list

{

'picasso': ['./dataset/cropped/picasso/picasso\_picasso9.jpg.png',

'./dataset/cropped/picasso/picasso\_picasso0.jpg.png',

'./dataset/cropped/picasso/picasso\_picasso5.jpg.png'],

'henry': ['./dataset/cropped/henry/henry\_henry5.jpg.png',

'./dataset/cropped/henry/henry\_henry7.jpg.png',

'./dataset/cropped/henry/henry\_henry1.jpg.png',

'./dataset/cropped/henry/henry\_henry8.jpg.png',

'./dataset/cropped/henry/henry\_henry14.jpg.png']

}