CSE 473/573- Computer Vision and Image Processing

Project 3

Name: Vivek Singh

UBID: vsingh28

UB#: 50418786

Part A: face detection

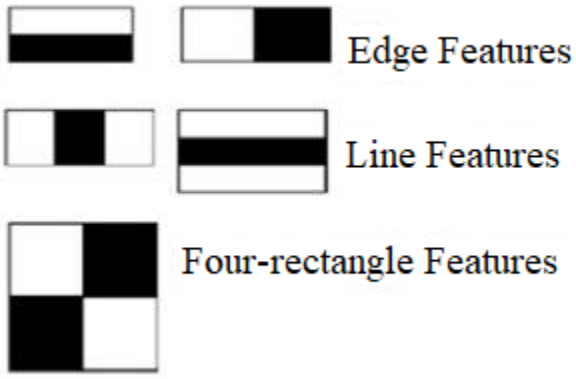
Problem Statement: Given and set of 100 images dataset on which we need to train the model to detect the faces.

Example:

Face detection is a typical problem where we need to identify the given image/object is contain any human face or not. This problem can be resolve using computer vision technology which can come handy in different ways, when we are training robots or in car automation.

Implementation:

1. Reading images from the given folder using cv2 library.
2. Using Viola-Jones algorithm in which HaarCascade function is used to detect the features in the bounding box.
   Viola-jones algorithm detect the front face in the grayscale image and then finds the location on the color image.
   "Haar" is an object detection algorithm which is used to detect features in each object(image/video). There are 3 main Haar like features that viola- jones used to identify
   - Edge features
   - Line features
   - Four-sides features



3. In my model I have tried to implement three main algorithms to solve the problem, these are:
   - Haarcascade_frontalface_default.xml
   - Haarcascade_frontalface_alt.xml
   - Haarcascade_profileface.xml
4. The best model which has given the highest **F1 score of 81.69 %** is "**Haarcascade_frontalface_default**", where the **scaleFactor** is set to : 1.23 which specify that image size is reduce at each image scale, and **minNeighbour** is set to 5 which signifies that how many neighbors each candidate rectangle should have to retain it
5. Once the function detects the faces in the images, I have stored the bounding boxes of the detected faces in a list.

```
[{"iname":  "img.jpg", "bbox":  [x, y, width, height]}, ...]
```

Results:

Evaluated F1 score:

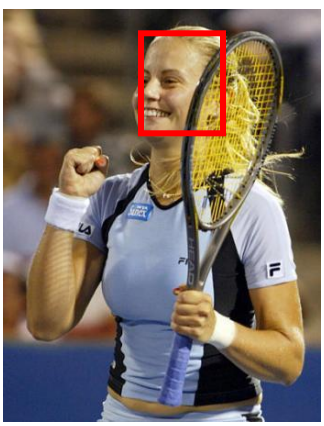
```
(project3) C:\Users\singh\Desktop\Project3_data>python ComputeFBeta
0.8169934640522876
```

Bounding Box list:


```
[{"iname": "img_104.jpg", "bbox": [73, 69, 181, 181]}, {"iname":
"img_106.jpg", "bbox": [81, 85, 231, 231]}, {"iname": "img_
107.jpg", "bbox": [228, 83, 90, 90]}, {"iname": "img_107.jpg",
"bbox": [58, 47, 102, 102]}, {"iname": "img_107.jpg", "bbox":
[331, 91, 94, 94]}, {"iname": "img_107.jpg", "bbox": [161, 82,
55, 55]}, {"iname": "img_108.jpg", "bbox": [67, 83, 145, 145]},
{"iname": "img_109.jpg", "bbox": [259, 121, 156, 156]},
{"iname": "img_110.jpg", "bbox": [97, 97, 126, 126]}, {"iname":
"img_111.jpg", "bbox": [68, 53, 102, 102]}, {"iname": "img_
112.jpg", "bbox": [100, 93, 215, 215]}, {"iname": "img_113.jpg",
"bbox": [308, 55, 94, 94]}, {"iname": "img_113.jpg", "bbox":
[149, 99, 152, 152]}, {"iname": "img_113.jpg", "bbox": [35, 135,
107, 107]}, {"iname": "img_114.jpg", "bbox": [12, 26, 114,
114]}, {"iname": "img_114.jpg", "bbox": [126, 99, 150, 150]},
```


Challenges:

As stated above using ”Haarcascade_frontalface_default.xml” will able to detect the faces which are facing in front, there are some image where it could not find the front face image, that leads to the low accuracy of my model.



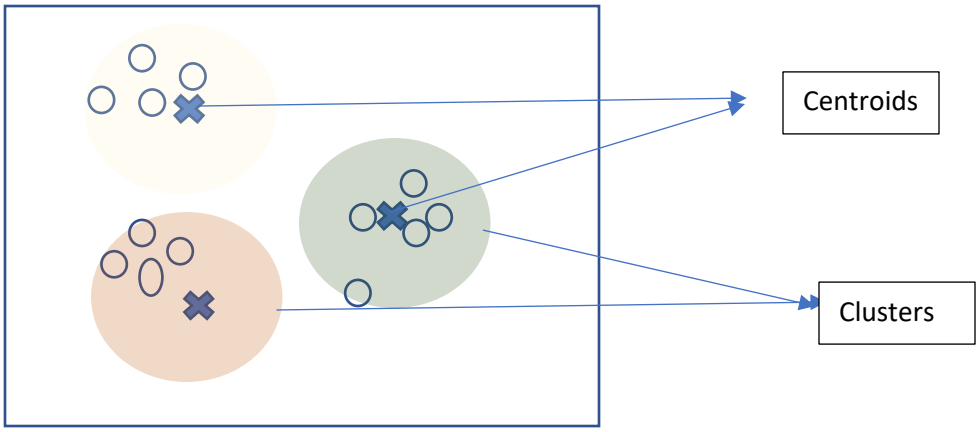
Secondly, in choosing the “scaleFactor” and “minNeighbour” value to get high f1 score. As if I choose scaleFactor value 1.3 then f1 score decreases to 70% while at 1.1 F1 is 81.599%

Part B: Face Clustering

Problem Statement: Cluster all the face which has been detected by above application and make the different clusters by choosing the K value.

Implementation:

- I have passed down the above function in clustering function, in which we have the bounding box values: image name and x, y, width and height of the detected face.
- By using face recognition library provided by open.cv2 I have first converted the BGR to RGB then passes the bounding box parameter of the detected face image to face recognition which essentially detect the bounding box and do the encoding calculation for each bounding box.
- Each face-location is a tuple of (top, right, bottom, left) where top = y, left = x, bottom = y + height, right = x + width.
- From " face recognition.face encodings(img, boxes)" I got the list of NumPy array of dimension 128
- Using the K means clustering algorithm I have found the clusters for our face images which represent the same identity.
- Idea behind choosing the K means algorithm is that: it is an unsupervised learning technique which puts the random K centroids to extract the similar features in data using the Euclidean distance formula between the data points.



Results: Accuracy: 100% as we can see from the clustered image below, after the evaluation of model there is no incorrect face is cluster together.

Cluster0

cluster 1

Cluster 2

Cluster 3

Cluster 4

Cluster json:


```
[{"cluster_no": 0, "elements": ["14.jpg", "15.jpg", "16.jpg",
"17.jpg", "18.jpg", "19.jpg", "20.jpg", "21.jpg"]},
{"cluster_no": 1, "elements": ["10.jpg", "11.jpg", "12.jpg",
"13.jpg", "9.jpg"]}, {"cluster_no": 2, "elements": ["1.jpg",
"2.jpg", "3.jpg", "4.jpg", "5.jpg", "6.jpg", "7.jpg", "8.jpg"]},
{"cluster_no": 3, "elements": ["22.jpg", "23.jpg", "24.jpg",
"25.jpg", "26.jpg", "27.jpg", "28.jpg", "29.jpg", "30.jpg"]},
{"cluster_no": 4, "elements": ["31.jpg", "32.jpg", "33.jpg",
"34.jpg", "35.jpg", "36.jpg", "36.jpg"]}]
```


Reference:

https://stackoverflow.com/questions/38316294/cannot-locate-haarcascade-default-frontal-face-xml-file-on-windows

https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php

https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

https://nrsyed.com/2018/03/29/image-segmentation-via-k-means-clustering-with-opencv-python/

https://www.geeksforgeeks.org/concatenate-images-using-opencv-in-python/