CSE 573: Computer Vision and Image Processing
Project 3 Report

Name: Abhishek Girishchandra Patel
UBID: 50365571

Part A: Face Detection

- For detecting faces, I have used Haar Feature-based Cascade Classifier which uses Haar features for face detection
- I have loaded pretrained classifier using "cv2.cascadeclassifier()". I have used "detectMultiScale ()" to perform face detection which takes following arguments as inputs:
    1. Image Array
    2. Scale Factor: Parameter specifying how much the image size is reduced at each image scale [1].
    3. Min Neighbors: Parameter specifying how many neighbors each candidate rectangle should have to retain it [1].
- Using trial and error method, optimal values for scale factor and min neighbors are found to be 1.2 and 5 respectively which gives F1 score 0.81. The result analysis is shown in table1.

Scale Factor	Min Neighbors	F1 score
1.1	2	0.745
1.1	3	0.7765
1.1	4	0.7891
1.1	5	0.7975
1.1	6	0.7925
1.1	7	0.7962
1.1	8	0.7974
1.1	9	0.7945
1.1	20	0.7755
1.2	5	0.81
1.3	5	0.7748
1.4	5	0.66

Table 1: Result analysis, change in F1 score as a function of scale factor and Min. Neighbors

- The detectMultiScale() returns boundary rectangles for the detected faces which I stored in a dictionary. The list of dictionaries containing image name and bounding box is stored in results.json file.

Part B: Cluster Faces

- Using detected faces from part A, for each faces I created 128 dimensional face encoder using face_recognition.face encodings().
- For clustering similar face encodings, I used k-means clustering. Following inputs were provided to cv2.kmeans()
    1. Face encodings (np.array of type float32)
    2. K=5: number of clusters
    3. criteria: It is the iteration termination criteria. When this criterion is satisfied, algorithm iteration stops [2]. The required accuracy epsilon and maximum iteration were chosen to be 1 and 10.
    4. Maximum iteration =10
    5. Flags: This flag is used to specify how initial centers are taken [2]. The flag CV2.KMEANS_RANDOM_CENTERS was chosen.
- The clustering algorithm provided labels of clusters (0 to 4) as an output parameter. This labels along with images were stored in clusters.json file.

References:

[1]https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html#aaf8181cb63968136476ec4204ffca498

[2] https://docs.opencv.org/4.x/d1/d5c/tutorial_py_kmeans_opencv.html