Computer Vision and Image Processing

Project 3

Name – Sannihith Kilaru

UBID – 50428194

Goal

To detect faces in the provided dataset and get an f1 score greater than 0.8.

Using this detection function we must recognize faces in the faceCluster_5 dataset. The algorithm to be used for this step is K-Means.

Techniques tried for face detection(Part A)

1. **Haar feature based cascade classifiers**

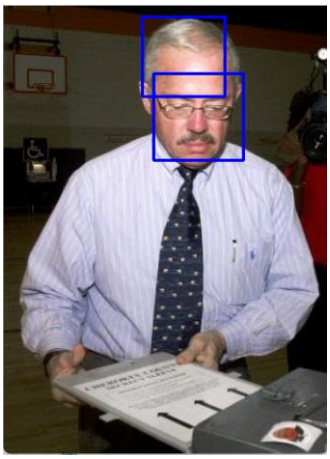
For detecting faces I used the Haar cascade classifier first because it is computationally much cheaper than Deep Neural Networks based techniques though it is less accurate. Haar uses edge detection just like the Viola Jones algorithm.

The f1 score of this detector on the validation dataset was < 0.8

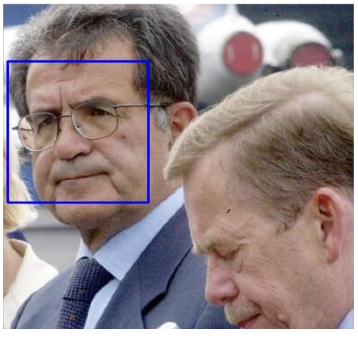
```
(automate) C:\Users\Sannihith\Documents\UB MS in Robotics\Computer Vision and Image Processing\Project3_data>python ComputeFBeta/ComputeFBeta.py --preds results.json --groundtruth validation_folder
/ground-truth.json
0.789156626506024
```

Taking a closer look at the detections, I found out:

- There are many false positives. Example:



- Faces at a side pose are not detected by the frontal face haar cascade classifier. Example:



- Faces that are blurred don’t get detected easily. Example:

- Faces are not detected most of the time if scale of face is too big/ too small. Example:

To improve the detection accuracy the next model I tried was the OpenCV DNN module.

2. **OpenCV DNN module**

The OpenCV’s DNN module takes two files:

- The model’s architecture file(opencv_face_detector.pbtxt)

  Reference:

  https://github.com/opencv/opencv/tree/4.x/samples/dnn/face_detector

- The model’s weight file(opencv_face_detector_uint8.pb)

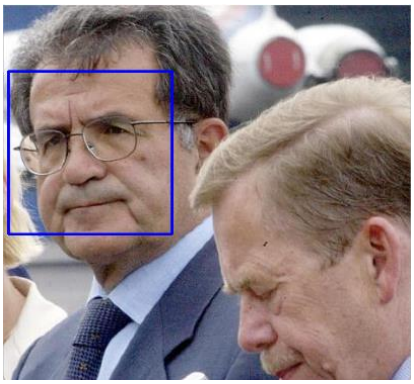
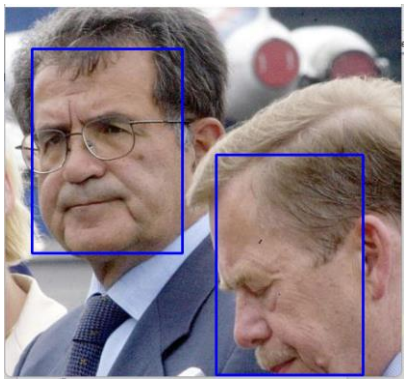
Reference:
https://raw.githubusercontent.com/opencv/opencv_3rdparty/dnn_samples_face_detector_20180220_uint8/opencv_face_detector_uint8.pb

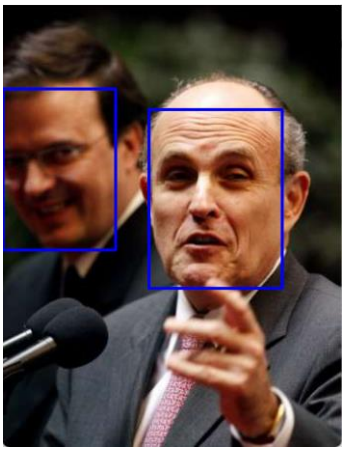
The detector is a Single-shot-multibox detector which uses a ResNet-10 Architecture.

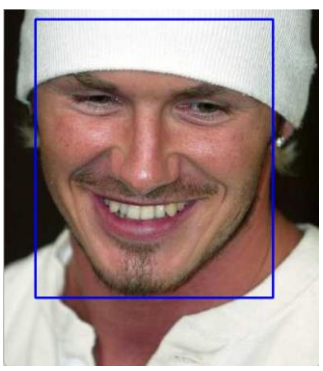
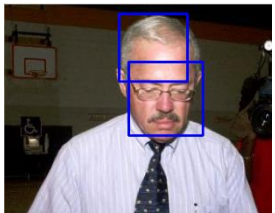
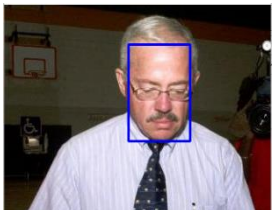
The F1 score obtained is 0.9272 which is much better than Haar cascade's score

```
(automate) C:\Users\Sannihith\Documents\UB MS in Robotics\Computer Vision and Image Processing\Project3_data>python Comp
uteFBeta/ComputeFBeta.py --preds results_val.json --groundtruth validation_folder/ground-truth.json
0.9272727272727272
```

Let's see the detection results side by side for the same examples shown above. The first column is the output using OpenCV's Haar cascade and the second is using OpenCV's DNN module.



- Face at a side pose is also being detected. (Row 1)
- Blurred faces are also being detected. (Row 2)
- OpenCV's DNN detector handles scale variance also. (Row 3)
- As we can see from the above comparison that false positives from are removed. (Row 4)
- Comparing the bounding box result for the same faces, the DNN module is more accurate as compared to the Haar. The IOU of the DNN module will be more with the groundtruth. (Row 1,2,4)

Recognition (Part B)

For recognition of faces KMeans clustering of the face encodings was done using the sklearn library. We were given the number of clusters to be taken. The face encodings for each face in the dataset were obtained from the face_recognition package.

The clustered images are shown below. As seen below it got all the images belonging to one person correctly clustered.

Conclusion

Haar has many drawbacks such as not performing when face's image is from the front, blurred images, image not the usual scale. Also, it has many false positives, and the bounding box is not accurate. These shortcomings are overcome by using OpenCV's DNN module. Clustering using KMeans successfully gave the result. All the faces were clustered correctly.