University at Buffalo

CSE 473/573 - Computer Vision and Image Processing

Pratik Malani – 50416266

Project 3

Face Detection in the Wild

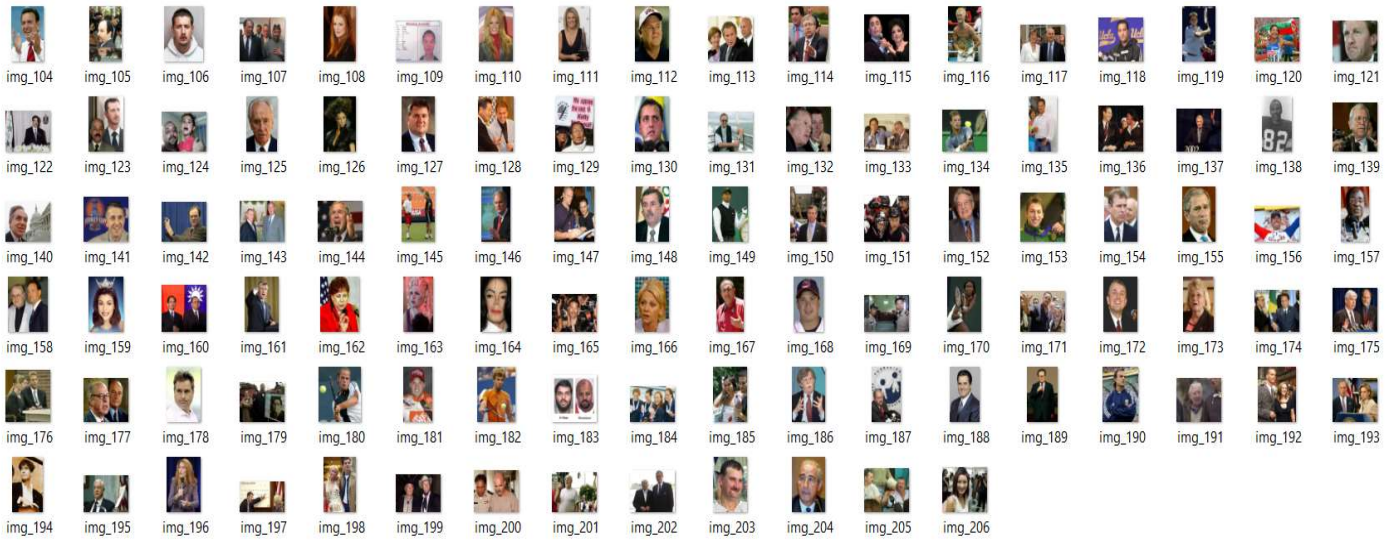
Part – A : Face Detection.

Algorithms Tried:

- **haarcascade_frontalface_default.xml**
- **haarcascade_profileface.xml**

The best accuracy achieved by the xml file in the code was haarcascade_frontalface_default.xml.

Images Given :



Face detection is the technology which uses computer vision libraries and algorithms to identify faces of the human in images and videos.

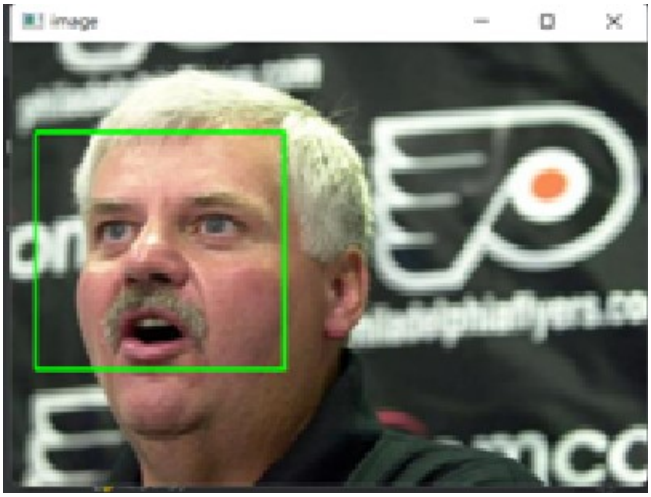
HAAR Cascade:

HAAR cascade is a feature-based algorithm for object detection. It was proposed by Paul Viola and Michael Jones and thus it is also called, Voila-Jones Face Detection.

Steps:

- To begin with the images were imported with respect to the location/directory where they were stored. Then they were read in a variable “img” and converted into grayscale for easier computation and detection.
- Next, pre trained HAAR cascade model is loaded using cv2.CascadeClassifier and the XML file “**haarcascade_frontalface_default.xml” is loaded.**
- Next the detection is done using “haar_cascade.detectMultiScale” method, which helps in finding the features of the images.
- After this cv2.rectangle returns the boundary rectangles to the detected images. It also gives [x,y,w,h] which are the bbox elements and those are appended back in the result list.

Result:

- Once we get the “results.json” file after running the code and the terminal commands, The F1 score achieved was **“0.8102893890675241”.**
- The initial accuracy achieved was 0.7(something) so to increase the accuracy the parameters that were changed were scaleFactor and minNeighbours. When the value for those were kept as 1.2 and 5 respectively I got the accuracy over 0.8.
- 

- The bounding box created on 69th image in the test_folder/images file.

Challenges:

- Increasing the scaling made it difficult for the code to find faces.
- The parameter tuning was another challenge as for multiple iterations I was stuck at 0.78-9 for the accuracy score.
- The code was initially done on Google Colab and then implemented in PyCharm to go step by step. The implementation of that code on PyCharm was a huge challenge as the code was giving a few errors while running it.

Part B : Face Clustering using K-Means.

Images Given:

K-Means Clustering for Image Segmentation is a method in python that uses openCV libraries to divide images to segment based on their characteristic of pixels. It helps to analyze and understand images more meaningfully. All the similar data will be grouped together.

Algorithm Tried:

K-Means Algorithm.

K-Means clustering algorithm is an unsupervised learning method having an iterative process in which the dataset are grouped into k number of predefined non-overlapping clusters or subgroups, making the inner points of the cluster as similar as possible.

Steps:

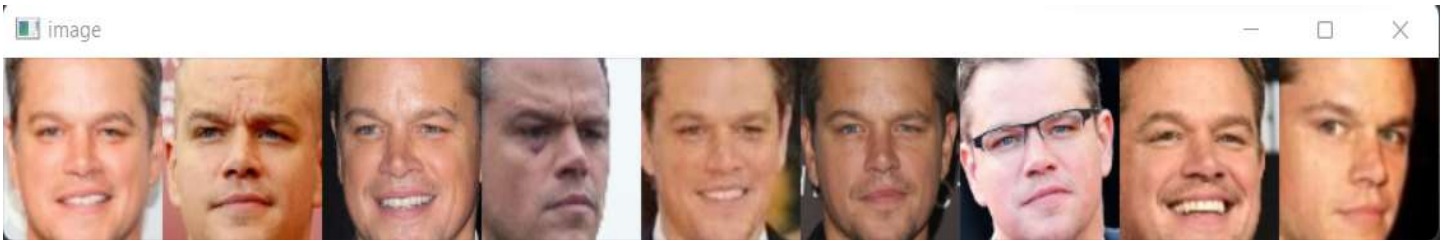
- The first step is to get the bounding boxes of the for the images specified in the path by using the face detection code form the first path.
- The next step is to apply encoding on the images and then crop them and also resize all of them to the same size.
- The function “face_recognition.face_encodings” represents the face using a set of 128 computer-generated measurements. Two different pictures of the same person would have similar encoding and two different people would have totally different encoding.
- Next the K-means function is implemented for clustering the faces and the cluster labels for each of the cropped faces were obtained and attached with their respective groups.
- Finally, the clustered images of each of the actors were concatenated respectively and visualized to check the accuracy of the code.

Results:

- After implementing clustering, the clusters of the respective actors are:

Cluster 0:

Cluster 1:



Cluster 2:

Cluster 3:

Cluster 4:

- Hence, the algorithm was able to correctly identify each image and put them in the right cluster.
- The final clusters were dumped in the json file “clusters.json”.

Challenges:

- The challenge was that initially when I was trying to visualize the clustered images I was trying to do it which matplotlib and showing them with plt.show(). But, the images that were been visualized were only the enoded images and I was trying to plot them using the subplot method by assigning rows and columns. Later I tried to show the images after using “hconcat” to concatenate the images.

References:

- https://www.geeksforgeeks.org/opencv-python-program-face-detection/

- https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d
- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- https://www.geeksforgeeks.org/python-multiple-face-recognition-using-dlib/#:~:text=The%20face_recognition%20API%20generates%20face%20encodings%20for%20the,two%20different%20people%20would%20have%20totally%20different%20encoding.
- https://medium.com/towardssingularity/k-means-clustering-for-image-segmentation-using-opencv-in-python-17178ce3d6f3#:~:text=K-Means%20Clustering%20for%20Image%20Segmentation%20using%20OpenCV%20in,us%20to%20analyze%20and%20understand%20images%20more%20meaningfully.
- https://www.gatevidyalay.com/k-means-clustering-algorithm-example/