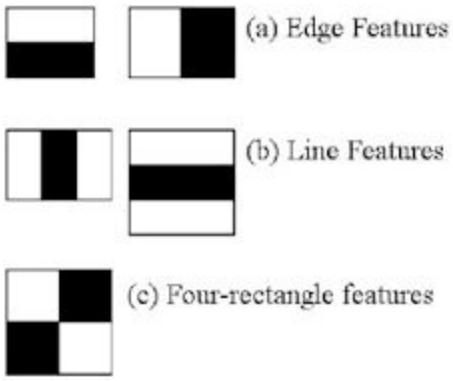
Assignment #2: Face Recognition
Probability and Random Processes (ES 331)
Dhyey Jani (18110068)

Introduction:

In this assignment, we try to use the Viola-Jones algorithm to *detect* the presence of a face, and we try to *recognize* the face using the Eigenface algorithm among the faces present in the dataset. For the sake of this assignment, we are using the *Yale Face Dataset* to accomplish the given tasks.

Using the Viola-Jones algorithm:

For this part of the assignment, we use the OpenCV module of Python. We need to train its cascade classifier using our own dataset. The classifier classifies an image based on the Haar features that are mentioned in the Viola-Jones paper. The Haar features look something like the image shown below.



Haar features
(ref.: https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html)

For training the cascade classifier, we need negative images as well apart from the positive images. The negative images would contain no face image for our case. For my dataset, I trained it using 99 images from the dataset as training images, and about 200 negative images that are only *background images.*

After training the cascade classifier, an XML file is generated which is used in the Python code to classify the images. The cascade classifier of OpenCV is implemented using the Viola-Jones algorithm, however, we are using our own dataset to *train* for the algorithm.

The relevant section of the code is commented to indicate the purpose of it. In a nutshell, the code creates a bounding box around the part of the image that it *detects* as a face and then it is cropped and saved in another directory. Since we also need these images to train the Eigenface algorithm, we run the Viola-Jones algorithm on the entire dataset to get the images cropped that contains the *detected* face. Below is an image that is detected and cropped.

Some anomalies for the Viola-Jones algorithm

On testing the images, some of the faces are not completely detected e.g. the images of *subject11.* Apart from these, there are images that are detected, however, the bounding box is surrounding only a part of the face, i.e., only half or quarter of the face. Such anomalies are not used further for the Eigenface algorithm. This type of image is shown below.

Results from Viols-Jones algorithm:

After running the Viola-Jones algorithm on the training set of 99 images and testing it over the rest of 55 images, we observe that the accuracy of detection of this algorithm is 88.667%

Resizing the images:

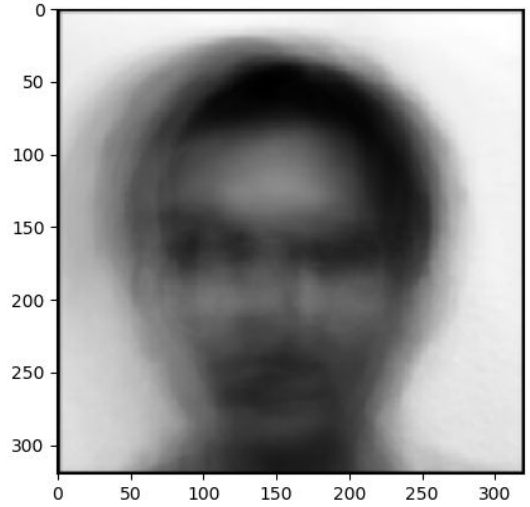
After detecting the images and cropping them, we need to scale up the images to the same size so that they are usable for the Eigenface algorithm. We need to do so since some of the steps require the flattened images to be of the same size so that operations like average can be calculated. Thus, we resize all the cropped images to the size of 320 x 320 using OpenCV.

Using Eigenface algorithm:

Now, we use the Eigenface algorithm to identify the image by mapping it to the label that is predicted upon the trained data. As input images, we feed in the cropped and resized images that are output from the Viola-Jones algorithm. We use 8 images per person from 14 persons from the dataset (total 112 train images), and use 19 images for testing.

Procedure:

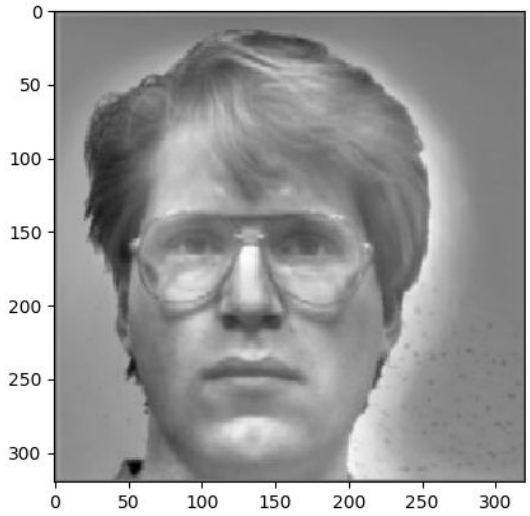
First, we flatten the images into a 1D array. Then, we take the average value of the flattened image matrices and subtract them from the original image flattened matrices. Every row of a matrix A is calculated as defined below.



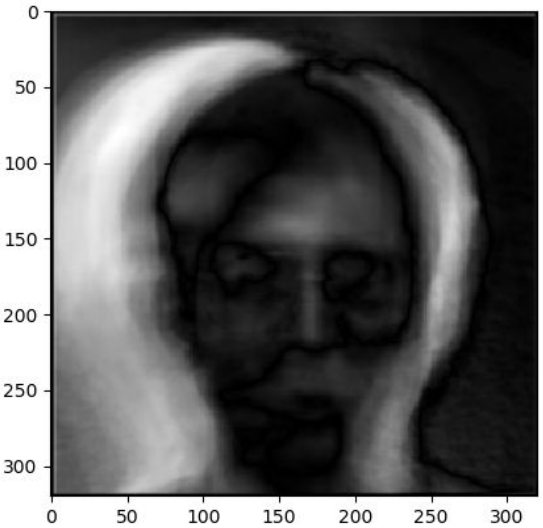
(*Average* face produced from all flattened image vectors)

$$A[i] = Img[i] - (1/n)\sum_{0}^{n} Img[i]$$

After doing this, we calculate the *reduced* covariance matrix, that is, $A^T A$. We calculate the eigenvalues and eigenvectors of this *reduced* covariance matrix from NumPy, and now proceed to calculate the *eigenfaces* using the reverse sorted eigenvectors matrix, reverse sorted according to the eigenvalues. For this, we multiply the reverse sorted eigenvector matrix (X) with A.



(A[0] - avg calculated from the training set)



(1st eigenface produced)

After calculating the Eigenfaces, we calculate the weight matrix from all these images. For doing so, we take the first 27 most important features from X (PCA of sorts). The number of features that we are using is not determined mathematically, but it can be analysed by varying it across the test images and checking for the best fit. In our case, we are taking it to be 27.

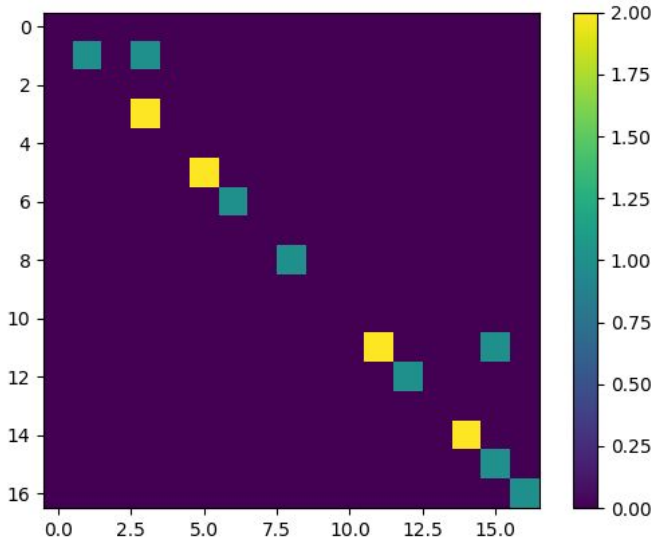
Testing Eigenfaces:

After training the Eigenfaces from the training images, we test it over 19 images. For testing each image, we proceed by calculating the weight matrix for each image by the method described above and then calculate the Euclidean distance between the test image weights and the weights matrix obtained from the training images. After calculating the distance, we calculate if any anomaly is present (that is, some image of a person which is entirely absent from the dataset), and discard it from the test set, and check if for some image, the matching person that exists from the index corresponding to the minimum weight.

For detecting the anomaly, we find the maximum possible distance from all the possible distances (minimum of all distance arrays) that might yield to a match and keep its magnitude as the threshold. If we encounter an image that has the magnitude of the minimum greater than the threshold, we flag it as an anomaly.

Results from Eigenfaces:

From the Eigenfaces algorithm, we obtain an accuracy of 82.36%. Also, the confusion matrix is as follows.



The output visible for the eigenfaces function is as shown below.

```
PS C:\Users\hp\Desktop\sem5\Assignment_2> & C:/Users/hp/AppData/Local/Programs/Python/Python37/python.exe c:/Users/hp/Desktop/sem5/Assignment_2/final_eigen.py
Image of subject01 detected as subject03
Image of subject01 detected as subject01
Image of subject03 detected as subject03
Image of subject03 detected as subject03
Image of subject05 detected as subject05
Image of subject05 detected as subject05
Image of subject06 detected as subject06
Image of subject08 detected as subject04
Image of subject08 detected as subject08
Image of subject09 detected as subject09
Image of subject09 detected as subject14
Image of subject09 detected as subject09
Image of subject10 detected as subject10
Image of subject13 detected as subject13
Image of subject13 detected as subject13
Image of subject14 detected as subject14
Image of subject15 detected as subject15
The accuracy is: 82.3529411764706
Images detected outside the dataset:
['anomaly_res', 'anomaly_res2']
The confusion matrix is:
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]]
PS C:\Users\hp\Desktop\sem5\Assignment_2>
```

(‘anomaly_res’ and ‘anomaly_res2’ are images that are not in the dataset, flagged as an anomaly)

The “cascading” between Eigenface and Viola-Jones:

First, we try to find if the input image contains a face or not using the Viola-Jones algorithm, and flag the face if it is a face or not. After this, we crop and resize the images, and feed it to the Eigenface. While training the images, we need to clean the training set to remove the images that are detected as having a face but are cropped with the bounding box covering only a small portion of the face in order to improve for the accuracy of Eigenface.

Observations:

From the Viola-Jones algorithm, we observe that some images show better performance in detection than the other, this happens when there is a considerable contrast between images and their background (For subject 13 we are getting a very high count of detected images as compared to the rest). Also, increasing the size of the dataset yields better results.

From the Eigenfaces algorithm, we observe that we can not only identify the presence of an image but also detect some kind anomaly and add it to the dataset for further better training (since from Viola-Jones we can already flag it off if it is not a face).

References:

- https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html
- M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Maui, HI, USA, 1991, pp. 586-591, doi: 10.1109/CVPR.1991.139758.
- P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.
- https://github.com/JoakimSoderberg/haarcascade-negatives/tree/master/images for getting the necessary negative images.
- OpenCV Python documentation
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
- http://vision.ucsd.edu/content/extended-yale-face-database-b-b

*** In the analysis, I have discarded the subject 11 images since it has the least detection (about 2 or 3) in Viola-Jones and thus it might cause complication while cascading the Viola-Jones output with Eigenfaces.