Eigenfaces for Face Recognition

Pushpendra Pratap Singh - 20110151

1 Understanding of Algorithm

1.1 The Eigenface Approach

Eigenvectors of a linear transformation is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it, i.e., they do not change any direction when the transformation is applied. To get an idea of how this could solve facial recognition, we can consider the following: The solution for the problem of facial recognition should be an algorithm that detects faces irrespective of all the different features among the different faces.

To do this, the algorithm should try to identify the details which remain constant irrespective of the change in overall facial features. Since eigenvectors are such vectors that remain almost constant (only scaled even if varies) while transformation, we can see analogically that the combination of eigenvectors could help to identify the constant element, which allows us to classify shapes as faces.

Mathematically, it was given in the paper that we have "find the principal components of the distribution of faces or the eigenvectors of the covariance matrix of the set of face images."

The most straightforward approach will be to use standard pattern recognition to find which of the predefined face classes best describes the new image. This is possible by comparing the Euclidian distance between the given face image $\tilde{t}$, and each image $\mathbf{t}^{(i)}$, from the dataset, i.e., the predefined face classes.

$$E_i = \left\| \tilde{\boldsymbol{t}} - \mathbf{t}^{(i)} \right\|$$

Whenever the value of a E_i is smaller than a chosen threshold α, the given face image can be identified as the person that is associated with the i-th face image.

However, the vectors that represent the face images often have large dimensionality (n^2-dimensional vectors). Even if the face images are relatively small in size such as 256 by 256 pixels, the vectors would have a dimension of 65,536 or equivalently, a point in 65,536 -dimensional space. Performing computations

on this huge dimensional vectors would be time consuming and computationally expensive.

Hence, we have to perform the facial recognition by utilising the concept of eigenvectors and eigenfaces.

1.2 Face Dataset

I have used AT&T "The Database of Faces" for this problem. The number of images in that dataset was 400 (out of which I have used 360 for training and 40 for testing), neither too high nor too low to form computation and gain accuracy. For a general case, suppose there is m number of nxn face images.

- Each of the images can be represented as an nxn matrix

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & \cdot & \cdot & \cdot & t_{1n} \\ \cdot & \cdot & \cdot & & \cdot & \\ \cdot & \cdot & & \cdot & & \cdot \\ t_{n1} & t_{n2} & \cdot & \cdot & \cdot & t_{nn} \end{bmatrix}$$

or, equivalently each of them can be represented as an n^2-vector where each of the columns are stacked below the previous columns as such :

$$\mathbf{t} = \begin{bmatrix} t_{11} \\ \cdot \\ t_{n1} \\ t_{12} \\ \cdot \\ t_{nn} \end{bmatrix}$$

- Now, all the m images can be represented as an n^2 x m matrix $\mathbf{F}$ where the superscript (i) will be used to denote the i-th image. Hence, i $\in 1, \ldots, m$.

$$\mathbf{F} = \begin{bmatrix} t_{11}^{(1)} & t_{11}^{(2)} & \cdot & \cdot & \cdot & t_{11}^{(m)} \\ t_{21}^{(1)} & t_{21}^{(2)} & \cdot & \cdot & \cdot & t_{21}^{(m)} \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \\ t_{nn}^{(1)} & t_{nn}^{(2)} & & \cdot & \cdot & t_{nn}^{(m)} \end{bmatrix}$$

1.3 Calculating Eigenfaces

As discussed in the first section, we have to find the eigenvectors of the covariance matrix of the set of face images. Since each of the vectors that represent the face images contributes more or less to each eigenvector, the image formed by each eigenvector is called eigenface. Each individual face can be represented

exactly in terms of a linear combination of the eigenfaces.

Furthermore, each of the face images can be approximated using only the eigenfaces/eigenvectors that are associated with the largest eigenvalues. The best M eigenfaces that spans a M-dimensional subspace is known as face space.

The mean face image $\boldsymbol{\mu}$ of the set is defined by

$$\boldsymbol{\mu} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{t}^{(i)}$$

Each of the face images differ from the mean by the vector $\widetilde{\boldsymbol{t}}$ defined by

$$\tilde{\boldsymbol{t}}^{(i)} = \mathbf{t}^{(i)} - \boldsymbol{\mu}$$

All of the $\tilde{\boldsymbol{t}}^{(i)}$ can be represented in a n^2xm matrix $\widetilde{\boldsymbol{F}}$ as

$$\widetilde{\boldsymbol{F}} = \begin{bmatrix} \tilde{\boldsymbol{t}}^{(1)} & \tilde{\boldsymbol{t}}^{(2)} & . & . & . \\ \boldsymbol{t}^{(m)} & & & & \end{bmatrix}$$

Here, each of the $\widetilde{\boldsymbol{t}}^{(i)}$ are vectors.

The $n^2 \times n^2$ covariance matrix of $\tilde{F}, \mathbf{C}$ is defined by

$$\mathbf{C} = \frac{1}{m}\sum_{i=1}^{m}\left(\widetilde{\boldsymbol{x}}^{(i)}\widetilde{\boldsymbol{x}}^{(i)^{\mathbf{T}}}\right) \quad \text{or} \quad \mathbf{C} = \frac{1}{m}\widetilde{A}\widetilde{A}^{\mathrm{T}}$$

Calculating the eigenvectors of $\mathbf{C}$ is a computationally expensive task since C is a relatively huge matrix. However, it can be shown (explained in detail in the reference paper) that the mxm symmetric matrix $\frac{1}{m}\widetilde{A}^{\mathrm{T}}\widetilde{A}$ has m eigenvalues that are also eigenvalues of $\mathbf{C}$.

$$\frac{1}{m}\widetilde{F}^{\mathrm{T}}\widetilde{F}\mathbf{v}_i = \lambda_i\mathbf{v}_i \Longrightarrow \widetilde{F}\left(\frac{1}{m}\widetilde{F}^{\mathrm{T}}\widetilde{F}\mathbf{v}_i\right) = \widetilde{F}\left(\lambda_i\mathbf{v}_i\right) \Longrightarrow \frac{1}{m}\widetilde{F}\widetilde{F}^{\mathrm{T}}\widetilde{F}\mathbf{v}_i = \lambda_i\widetilde{F}\mathbf{v}_i$$
$$\Longrightarrow \mathbf{C}\tilde{F}\mathbf{v}_i = \lambda_i\widetilde{F}\mathbf{v}_i$$

From above, it can be concluded that both $\frac{1}{m}\widetilde{F}\widetilde{F}^{\mathbf{T}}$ and $\frac{1}{m}\widetilde{F}^{\mathbf{T}}\widetilde{F}$ have the same eigenvalues. This implies that, if $m < n^2$, then there will only be m, rather than n^2 meaningful eigenvectors. All the remaining eigenvectors will have associated eigenvalues of zero.

Therefore, we construct the $m \times m$ symmetric matrix $\mathbf{L} = \frac{1}{m}\widetilde{F}^{\mathrm{T}}\tilde{F}$ and find its eigenvectors $\mathbf{v}_i$. Using the eigenvectors of $\mathbf{L}$, the eigenvectors of $\mathbf{C}$ can be found by $\widetilde{F}\mathbf{v}_i$ (whose eigenvectors are n^2-dimensional vectors). With this method, the computational power needed to find the eigenvectors of $\mathbf{C}$ is greatly reduced. The associated eigenvalues can be used to rank the eigenvectors in a descending order.

1.4 Using Eigenfaces to Classify a Face Image

The m eigenvectors calculated from $\mathbf{L}$ spans a basis set that describes the face images. However, in practice, a smaller number of eigenvectors is sufficient for identification. Therefore, we will select the best M eigenvectors such that $M < m$ based on our ranking. These M eigenvectors are our eigenfaces and the space spanned by them is our face space.

A new image $\hat{t}$ (n^2-vector) is first subtracted by the mean vector from the set of images $\widetilde{\boldsymbol{t}}$. The vector $\hat{t} - \boldsymbol{\mu}$ is then projected to the face space and represented by

$$\hat{\boldsymbol{t}}_{proj} = \sum_{i=1}^{q} \left(\mathbf{u}_i^{\mathbf{T}}(\hat{\boldsymbol{t}} - \boldsymbol{\mu})\right) \mathbf{u}_i$$

where

$$\mathbf{u}_i = \frac{\tilde{F}\mathbf{v}_i}{\left\|\tilde{F}\mathbf{v}_i\right\|}$$

The vector $\hat{\boldsymbol{t}}_{\text{proj}}$ is the new face image represented by the linear combination of the eigenfaces. The more the eigenfaces used for the reconstruction, the more the reconstructed face image looks like the original new face image. Notice that $\hat{\boldsymbol{t}}_{\text{proj}}$ and the eigenfaces are n^2-vectors. The vector $\hat{t} - \boldsymbol{\mu}$ can also be represented as a q-vector with respect to the eigenfaces, $\boldsymbol{\Omega}$

$$\boldsymbol{\Omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ . \\ . \\ . \\ \omega_q \end{bmatrix}$$

where

$$\omega_i = \mathbf{u}_i^{\mathbf{T}}(\hat{\boldsymbol{t}} - \boldsymbol{\mu}) \quad \text{for } i \in 1, 2, \ldots, q$$

1.5 Face Detection

To determine whteher the given image $\hat{t}$ is a face image, the distance of $\hat{t}$ from the face space can be used. The distance of $\hat{t}$ from the face space β is

$$\beta = \sqrt{\|\hat{t} - \hat{t}_{\text{proj}}\|}$$

If β is smaller than a chosen threshold α_1, then $\hat{t}$ can be categorized as a face image.

1.6 Face Recognition

To determine which face class provides the best description of the new image, we need to find the face class k that minimizes the Euclidean distance

$$E_k = \sqrt{\left\| \left(\mathbf{\Omega} - \mathbf{\Omega}^{(k)} \right) \right\|} \quad \text{for } k \in 1, 2, \ldots, m$$

where $\mathbf{\Omega}^{(k)}$ is the vector of $\widetilde{\boldsymbol{t}}^{(k)}$ represented with respect to the eigenfaces. Therefore,

$$\mathbf{\Omega}^{(k)} = \begin{bmatrix} \omega_1^{(k)} \\ \omega_2^{(k)} \\ \cdot \\ \cdot \\ \cdot \\ \omega_q^{(k)} \end{bmatrix}$$

where

$$\omega_i^{(k)} = \mathbf{u}_i^{\mathrm{T}} \widetilde{\boldsymbol{t}}^{(k)} \quad \text{for } i \in 1, 2, \ldots, q$$

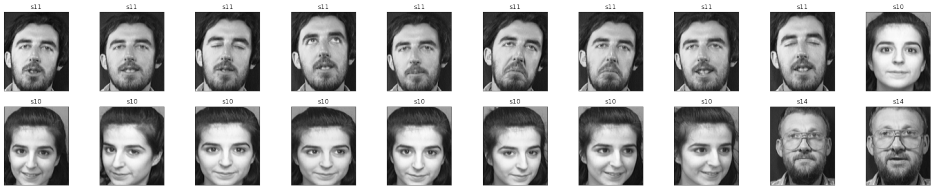
If the smallest E_k is lower than a chosen threshold α_2, the new face image $\hat{t}$ is classified as face k.

2 Results

To analyze the algorithm, I have divided the AT&T dataset into two sets. AT&T dataset contains 10 images each of 40 different people. I have separated 1 image each for every person into test set and the rest of the 360 images are the training set for the algorithm.

2.1 Training Set

The following are the first 20 images of the training set. Note that each person has 9 corresponding images, as 1 is in the test set.

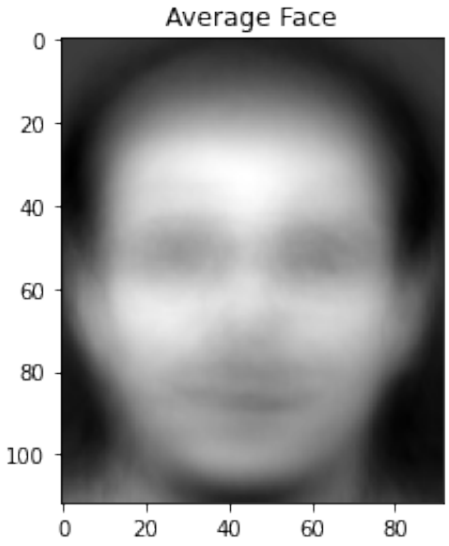


2.2 Test Set

The following are the 40 images used in the test set. In addition to this, Additionally, I have also conducted a test with a random set which contained some images from the training set and some random images which are not present in the training set and thus are unknown to the algorithm.

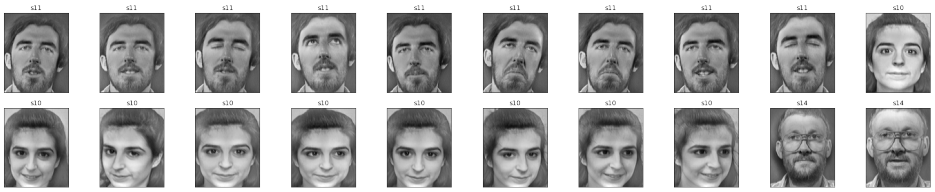
2.3 The Average Face

The average face of the training set is shown below. It is calculated by summing the elements of F and dividing by the total number of the images of the training set.



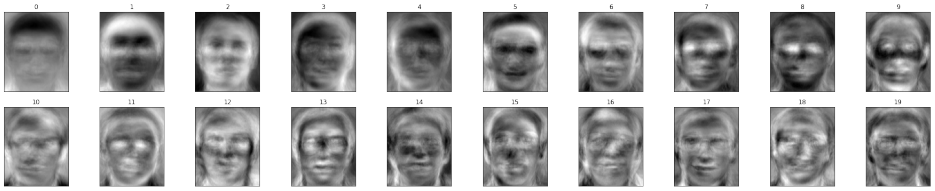
2.4 Mean Subtracted Images

The following are the first 20 mean subtracted images of the training set. Note that each person has 9 corresponding images, as 1 is in the test set.



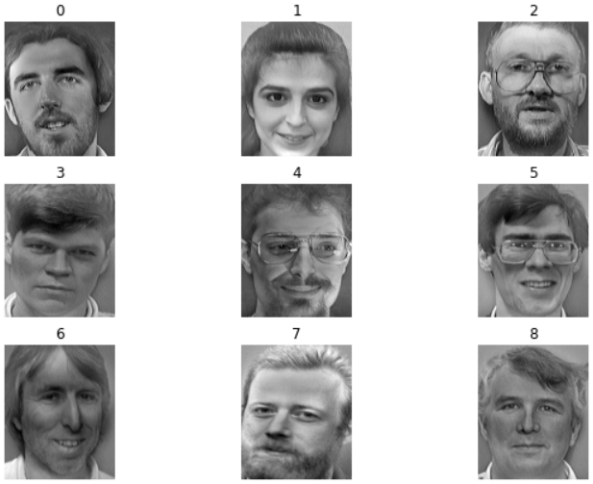
2.5 Eigenfaces

The following are the first 20 eigenfaces of the training set. Note that each person has 9 corresponding images, as 1 is in the test set.



2.6 Computations on Test-Set - Mean Subtracted Image

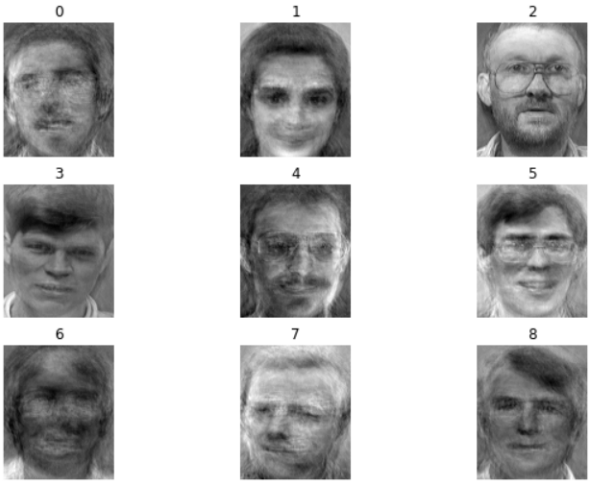
The following are the first 9 mean subtracted images of the test set.



Note that to find these mean subtracted images, I have taken q (number of chosen eigenfaces) = 350.

2.7 Computations on Test-Set - Face Reconstruction

The following are the first 9 reconstructed images of the test set.



2.8 Computations on Test-Set - Face Detection

The following are the results of face detection on the test set. Output is the same as obtained in colab file. The face column tells whether the face is detected in the image. Second tells us the smallest value and it will be less than threshold of 3000 if a face is detected.

```
Face detected!  1457.763885977506
Face detected!  1678.4948706753032
Face detected!  1429.1094824938086
Face detected!  1862.3354446145145
Face detected!  1033.9664578210006
Face detected!  1202.5658645147344
Face detected!  1688.5531461733149
Face detected!  1860.8035781090869
Face detected!  1649.8048597229517
Face detected!  1278.9262520749144
Face detected!  1192.7235378459918
Face detected!  1580.6163730141423
Face detected!  1810.7878275902185
```

```
Face detected!  1430.8077922815976
Face detected!  1618.8460830294227
Face detected!  1574.9330961603039
Face detected!  1628.1863816402415
Face detected!  1207.792372630867
Face detected!  1538.230157954824
Face detected!  1623.8836898367733
Face detected!  1634.8171754650623
Face detected!  1194.4019215883802
Face detected!  1386.3118486773426
Face detected!  1393.9320117542586
Face detected!  1601.202822047172
Face detected!  1326.6451817411407
Face detected!  1419.317124310311
Face detected!  1285.000665007869
Face detected!  1287.972564774455
Face detected!  1396.3693443174695
Face detected!  1405.8512791442529
Face detected!  1356.1610786348654
Face detected!  1138.4725815127642
Face detected!  1551.7760503751144
Face detected!  1307.742463509623
Face detected!  1192.4886897939448
Face detected!  1657.6206562081331
Face detected!  1436.4794299258513
Face detected!  1080.6798193810612
Face detected!  1516.8407546804208
```

It can be seen that the algorithm detects faces in each image, which is the correct result. The threshold for face detection, i.e. α is 3000. This was obtained from trial error on the test set. This particular value passed all the cases in this test set and the random set test, which I conducted afterwards. Hence, I chose this value for α.

2.9 Computations on Test-Set - Face Recognition

The following are the results of face recognition on the test set. Output is the same as obtained in the colab file. After these results, I have also added the table, which contains indexes of matching indexes with a given test image.

```
0. Actual-Id: s18   Result: s18   smallest_value: 1759.937891552405
1. Actual-Id: s1   Result: Unknown Face!   smallest_value: 3487.4149787107654
2. Actual-Id: s21   Result: s21   smallest_value: 1913.3822631669952
3. Actual-Id: s32   Result: s32   smallest_value: 2147.4213523689177
```

```
4. Actual-Id: s17   Result: s17   smallest_value: 801.8004298402174
5. Actual-Id: s9   Result: s9   smallest_value: 1790.743122938786
6. Actual-Id: s36   Result: s36   smallest_value: 2090.192229516275
7. Actual-Id: s11   Result: s11   smallest_value: 2372.120428656442
8. Actual-Id: s12   Result: s12   smallest_value: 2667.5570120684897
9. Actual-Id: DME   Result: DME   smallest_value: 1468.1019231815421
10. Actual-Id: s39   Result: s39   smallest_value: 1847.2883769082005
11. Actual-Id: s3   Result: s3   smallest_value: 1508.5776292057485
12. Actual-Id: s4   Result: s4   smallest_value: 2465.9440674742873
13. Actual-Id: s28   Result: s28   smallest_value: 1394.6719152733651
14. Actual-Id: s15   Result: s15   smallest_value: 2358.847743161786
15. Actual-Id: s20   Result: s20   smallest_value: 1610.7584587995839
16. Actual-Id: s2   Result: s2   smallest_value: 1945.5702255465105
17. Actual-Id: s7   Result: s7   smallest_value: 837.9867479540957
18. Actual-Id: s34   Result: s34   smallest_value: 1036.8182862708238
19. Actual-Id: s14   Result: s14   smallest_value: 3009.2692416591676
20. Actual-Id: s35   Result: s35   smallest_value: 1517.8444302507467
21. Actual-Id: s37   Result: s37   smallest_value: 1830.3760472442284
22. Actual-Id: s25   Result: s25   smallest_value: 1605.6080316365133
23. Actual-Id: s6   Result: s6   smallest_value: 2586.2926732112865
24. Actual-Id: s24   Result: s24   smallest_value: 1689.9626437883755
25. Actual-Id: s27   Result: s27   smallest_value: 2200.9771843894314
26. Actual-Id: s40   Result: s40   smallest_value: 1883.3938097189716
27. Actual-Id: s10   Result: s10   smallest_value: 1753.806799427839
28. Actual-Id: s29   Result: s29   smallest_value: 2096.4564316796145
29. Actual-Id: s38   Result: s38   smallest_value: 2153.46871068253
30. Actual-Id: s33   Result: s33   smallest_value: 2438.040072196313
31. Actual-Id: s26   Result: s26   smallest_value: 1986.168877240491
32. Actual-Id: s30   Result: s30   smallest_value: 1845.5701164469303
33. Actual-Id: s8   Result: s8   smallest_value: 1950.6607167884422
34. Actual-Id: s19   Result: s19   smallest_value: 1680.304168433191
35. Actual-Id: s13   Result: s13   smallest_value: 1780.362656211241
36. Actual-Id: s5   Result: s5   smallest_value: 2393.1159240032216
37. Actual-Id: s16   Result: s16   smallest_value: 1195.511458870295
38. Actual-Id: s31   Result: s31   smallest_value: 1541.049268567878
39. Actual-Id: s23   Result: s23   smallest_value: 1815.6069868578911
Correctly recognized: 39 images out of 40 images.
Accuracy: 97.5
```

The Actual-id is the id of the test set, which was labelled before splitting the complete dataset into training and test. The result is the label/id of the index which returns the smallest value for a given test image. The smallest value in the table above represents the smallest Euclidian distance which enabled the algorithm to identify the label and, thus, recognize the face. The threshold for facial recognition is $\alpha = 3200$. This value was obtained from trial and error. At this particular, most test cases seemed to pass without compromising the

accuracy of the algorithm when an unknown image is introduced.

Matching Indexes

```
0. Below Threshold   Smallest index = 3
Matching indexes = [1, 3]
1. Below Threshold   Smallest index = 15
Matching indexes = [1, 4, 5, 7, 15]
2. Below Threshold   Smallest index = 20
Matching indexes = [1, 18, 20]
3. Below Threshold   Smallest index = 29
Matching indexes = [1, 3, 27, 28, 29]
4. Below Threshold   Smallest index = 44
Matching indexes = [15, 19, 24, 37, 38, 39, 44]
5. Below Threshold   Smallest index = 51
Matching indexes = [1, 2, 18, 19, 24, 45, 46, 47, 51]
6. Below Threshold   Smallest index = 57
Matching indexes = [1, 4, 15, 18, 19, 24, 25, 39, 54, 57]
7. Below Threshold   Smallest index = 69
Matching indexes = [1, 3, 5, 15, 27, 31, 44, 63, 68, 69]
8. Below Threshold   Smallest index = 74
Matching indexes = [1, 4, 72, 73, 74]
9. Below Threshold   Smallest index = 89
Matching indexes = [1, 15, 18, 20, 81, 82, 86, 89]
10. Below Threshold   Smallest index = 92
Matching indexes = [2, 19, 23, 37, 49, 90, 92]
11. Below Threshold   Smallest index = 102
Matching indexes = [1, 3, 44, 47, 48, 50, 91, 98, 102]
12. Below Threshold   Smallest index = 113
Matching indexes = [1, 5, 7, 74, 79, 80, 91, 98, 112, 113]
13. Below Threshold   Smallest index = 123
Matching indexes = [1, 13, 15, 24, 27, 31, 42, 117, 120, 122, 123]
14. Below Threshold   Smallest index = 128
Matching indexes = [1, 8, 15, 18, 20, 24, 128]
15. Below Threshold   Smallest index = 138
Matching indexes = [1, 5, 13, 15, 120, 122, 135, 136, 137, 138]
16. Below Threshold   Smallest index = 151
Matching indexes = [15, 18, 19, 39, 57, 144, 150, 151]
17. Below Threshold   Smallest index = 155
Matching indexes = [1, 18, 20, 82, 83, 86, 87, 122, 153, 155]
18. Below Threshold   Smallest index = 164
Matching indexes = [1, 8, 12, 64, 109, 112, 114, 163, 164]
19. Below Threshold   Smallest index = 172
Matching indexes = [1, 3, 91, 117, 172]
20. Below Threshold   Smallest index = 186
Matching indexes = [1, 5, 7, 111, 113, 180, 186]
```

```
21. Below Threshold   Smallest index = 193
Matching indexes = [44, 45, 47, 48, 53, 106, 107, 189, 193]
22. Below Threshold   Smallest index = 204
Matching indexes = [1, 19, 45, 118, 119, 121, 198, 199, 200, 204]
23. Below Threshold   Smallest index = 207
Matching indexes = [4, 12, 15, 18, 19, 21, 25, 207]
24. Below Threshold   Smallest index = 224
Matching indexes = [1, 13, 15, 27, 30, 110, 122, 135, 217, 218, 224]
25. Below Threshold   Smallest index = 227
Matching indexes = [227]
26. Below Threshold   Smallest index = 242
Matching indexes = [4, 15, 18, 19, 24, 25, 45, 51, 235, 236, 242]
27. Below Threshold   Smallest index = 249
Matching indexes = [1, 18, 20, 85, 243, 245, 247, 249]
28. Below Threshold   Smallest index = 253
Matching indexes = [1, 3, 4, 18, 19, 20, 44, 48, 97, 252, 253]
29. Below Threshold   Smallest index = 265
Matching indexes = [1, 18, 20, 24, 81, 82, 88, 243, 245, 262, 263, 265]
30. Below Threshold   Smallest index = 276
Matching indexes = [1, 74, 80, 194, 270, 275, 276]
31. Below Threshold   Smallest index = 285
Matching indexes = [1, 4, 15, 18, 19, 24, 255, 281, 285]
32. Below Threshold   Smallest index = 293
Matching indexes = [4, 15, 106, 238, 288, 289, 291, 293]
33. Below Threshold   Smallest index = 299
Matching indexes = [1, 3, 16, 28, 29, 218, 222, 224, 298, 299]
34. Below Threshold   Smallest index = 314
Matching indexes = [1, 3, 307, 310, 314]
35. Below Threshold   Smallest index = 317
Matching indexes = [1, 228, 317]
36. Below Threshold   Smallest index = 328
Matching indexes = [1, 4, 13, 15, 326, 328]
37. Below Threshold   Smallest index = 341
Matching indexes = [1, 18, 20, 24, 35, 109, 115, 117, 118, 120, 153, 155, 156, 334, 341]
38. Below Threshold   Smallest index = 343
Matching indexes = [1, 18, 19, 24, 45, 343]
39. Below Threshold   Smallest index = 357
Matching indexes = [1, 15, 18, 20, 81, 82, 351, 354, 357]
```

2.10 Computations on Random Test Set - Results

The following are the results of face detection on the random test set. This random test set contains 10 different images. 3 of these images are of the persons from the training set. The rest of them are of various objects/people. All of these images will be available in the folder submitted with the assignment. You have to import these to google colab to run without an error.

Results of Face Detection:
The first column tells whether a face is detected, and the second tells the smallest value obtained by the algorithm in the process. It should be less than the threshold if a face is detected.

```
Face detected!  263.0872160381701
Face detected!  2506.838403478475
Face detected!  2332.887767835824
Face detected!  1927.2178517211769
Face detected!  235.32796757223264
Face detected!  241.68141138205536
No face detected!  3345.2287810194903
Face detected!  2319.7545283133004
No face detected!  5728.936150800011
Face detected!  2591.705249741133
```

Results of Face Recognition:
The first column tells about the smallest distance, second tells whether a face is Unknown or returns the id if it is known. The third tells us about the minimum index, and the last list returns all the matching index values who is less than the threshold.

```
26.637020330722425 s17 42 [12, 13, 37, 42]
5756.2522907220955 Unknown Face! 187 [1, 2, 8, 18, 186, 187]
4263.740507621488 Unknown Face! 3 [1, 3]
4322.295179450308 Unknown Face! 166 [1, 12, 18, 19, 20, 24, 83, 162, 166]
26.31400721224369 s40 240 [4, 15, 18, 19, 20, 234, 236, 240]
25.565353651116734 s18 6 [2, 6]
6022.930949748776 Unknown Face! 166 [12, 18, 20, 83, 162, 166]
4935.600173916786 Unknown Face! 223 [1, 4, 5, 15, 20, 31, 44, 223]
10189.30544443693 Unknown Face! 321 [1, 3, 177, 321]
8172.418104699671 Unknown Face! 11 [1, 3, 11]
Accuracy for Face Detection: 90
Accuracy for Face Recognition: 90
```

It is clear from the above output that the algorithm works correctly. If we consider this random test set, then by calculating the total of true positives and true negatives, we can get the total accuracy of the algorithm.

3 Accuracy and Inferences

To calculate the accuracy, I have used the following method:

Let us say that the total number of test set images is n. Now, for a particular image, if the face is correctly detected, then the variable *detect* is equal to 1, and it is equal to 0 if otherwise. Similarly, if the face is correctly recognized

from the training set, then the variable *recognize* is equal to 1, and it is equal to 0 if otherwise. Now, *net* variable is product of *detect* and *recognize*. We add this *net* for all the images and then divide it by the total number of test images to obtain the accuracy of the algorithm.

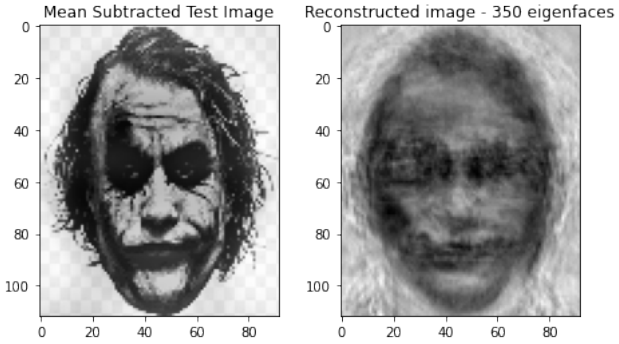
1. For the first test set, which was derived from the training set, the algorithms correctly detect the face images as faces. Additionally, it also recognizes all but one test image. Therefore, accuracy, in that case, came out to be 97.5 %.

2. For the random test set, which contained a few of persons from the training set and rest are different, it detected one face wrongly and consequently predicted it as unknown face. Other than this image, all the other images were correctly detected and recognized. Therefore, accuracy, in that case, came out to be 90%. However, the lower accuracy is caused by low number of test cases. If the test cases are increased drastically then the algorithm will perform better.

The accuracy of the algorithm is heavily dependent on the training set. The accuracy of the algorithm will improve if the training set contains a large number of objects/persons with a variety of expressions.

Suppose, for example, an algorithm is trained on a set which does not contain many expressions, then, if a new unknown image is shown to the algorithm, it might not be able to detect the face. In my case, the AT&T database contains all the different expressions, so even when introducing an unknown image with facial expressions shown, the algorithm is able to detect the face.

One hindrance to accuracy is the resolution, quality and colour of the image. The one case from the random test case where my algorithm failed had the following image:



In this scenario, the facial expressions were similar to those in the training set. However, the algorithm was trained on 8-bit grayscale images. All the images in the training set could be understood with the 8 bits. However, this particular image requires greater resolution and more number of colours to correctly detect the face and simultaneously recognize it. Thus, if a training set

has a better quality of images with better resolution and more colours and then it can detect faces even when a varied colour/expression-based new unknown image is shown.

4 References

- Turk, M. A., and Pentland, A. P. (1991, June). Face recognition using eigenfaces. In Proceedings 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 586-591). IEEE.
- Huang, K. (2012). Principal Component Analysis in the Eigenface Technique for Facial Recognition.
- Face-Recognition-Using-Eigenfaces
- Face-Recognition-using-eigen-faces
- ML | Face Recognition Using Eigenfaces (PCA Algorithm)