Report

Assignment 2: Eigenfaces for Face Recognition

Objective:

Design a real-time face recognition system using the eigenfaces

Understandings and Implementation Details:

- I have used *Caltech Faces* dataset of 450 images of different human beings faces and a few number of cartoon images.
- We have 450 images and each of dimension 542* 600. We resize it into 400*400.
- We convert it into a vector of size N^2.
- Then we calculate the mean of the face vectors and subtract it from each vectors.
- This will do with all the images and this will give a matrix of the size of $N^2 \times M$.
- Then find the Covariance matrix by the multiplication of A and A^T. Dimension of A is $N^2 \times M$.

  $Cov = A^T \times A$

  The dimension of covariance matrix is $M \times M$. Here M is number of images and it is computationally efficient.
- After computing the covariance matrix, we calculate the eigenvalues and eigenvectors using the formula:

  $A^T A v_i = \lambda_i v_i$

  $AA^T A v_i = \lambda_i A v_i$

  $C' u_i = \lambda_i u_i \quad where\ C' = AA^T\ and\ u_i = Av_i$
- Here $C'\ and\ C\ have\ same\ eigenvalues\ ad\ eigen\ vector.\ C = A^T A$
- First we calculate eigenvectors and eigenvalues of C and map them to C' using the formula $u_i = Av_i$.
- Corresponding to the K largest eigenvalues of C' we select the K eigenvectors and K should be less than M.

- Thereafter take the normalized training faces.
- Then represent each face vector in the linear combination of K eigenvectors

  Formula: $x_i - \psi = \sum_{j=1}^{k} w_j u_j$ where u_j $are\ the\ eigenFaces$
- Take the coefficient of eigenfaces and represent the training faces in form of a vector.

Testing

- In the testing implementation, I separated 45 images from the given dataset.
- The first process is preprocessing the data.
- Each face vector is subtracted from the test face from mean values.
- We normalize the vector and project into eigenspace to get eigenfaces.

  $\phi = \sum_{i=1}^{k} w_i u_i$
- We take the vector of coefficients and subtract it from training image to get the least distance between train image and test image.
- Using this distance we plot the %error vs k graph.

Results:

Mean Face:

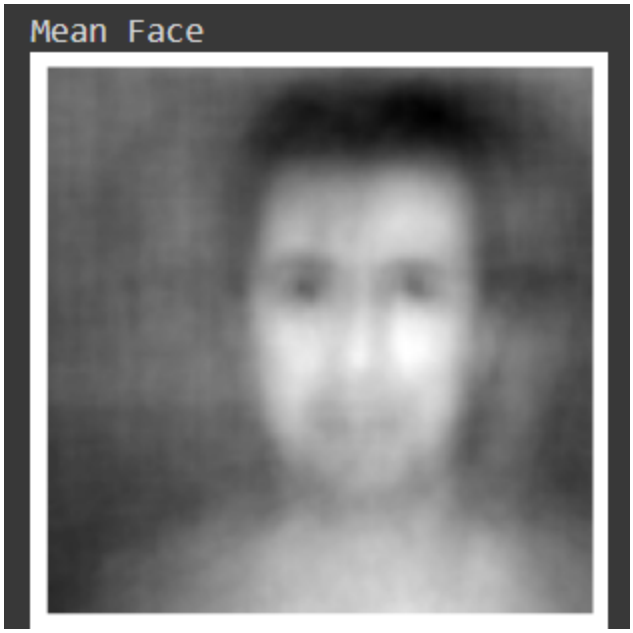


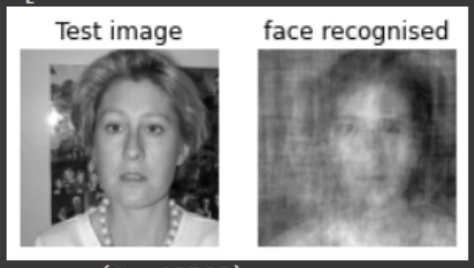
Fig1: Mean face

```
[[0.46668603]
 [0.47240862]
 [0.47271847]
 ...
 [0.49461147]
 [0.49421448]
 [0.49722585]] (62500, 1)
```

Fig2: Mean values and mean vector size

Test image and Face recognised

```
  [0.47587419 0.45337961 0.46170792 ... 0.64702599 0.63785716 0.61222393]]
```

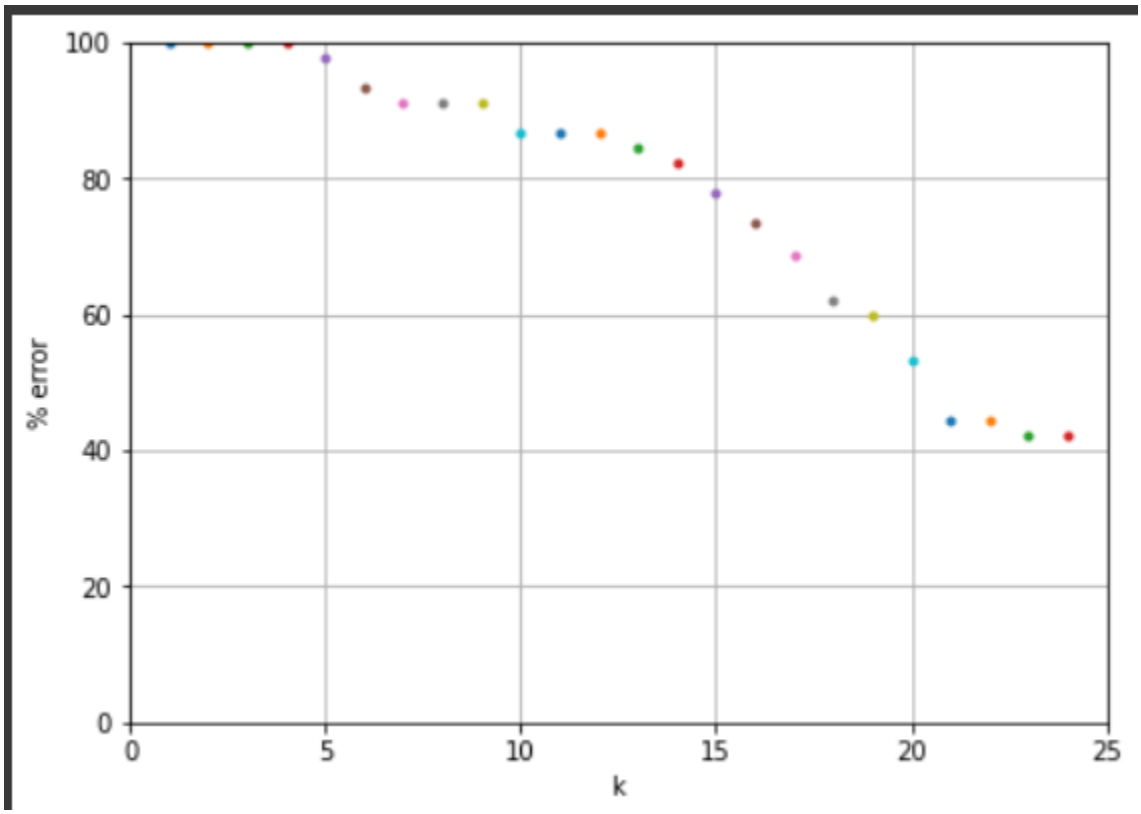



```
zmean (1, 62500)
uvx (62500, 15)
zmean (1, 62500)
(405, 15)
(15, 1)
Zmean shape (1, 62500)
fc_old (62500,)
fc_new (1, 62500)
[[-0.04965919 -0.01045981 -0.0101431  ... -0.00793617 -0.05493875
  -0.03687919]
 [-0.01601329  0.00459134  0.01188229 ...  0.10565403  0.0067655
   0.02378138]
 [-0.02319293  0.01869977  0.03623992 ...  0.04314162 -0.01217895
```

Accuracy:

The graph will show the accuracy of the test model under different K values with a certain threshold.

Percentage error vs K (K number of eigenvectors) graph:



We set the threshold of 20 that if the distance is greater than 20 then it is not correctly detected. From this, we get the result of the above graph.

Inferences from the Result:

1. We can further decrease the result through increase the dataset
2. We can use the Convolutional Neural Network model to build a better Face Recognition System.
3. From the above accuracy graph, we get the insight that on increasing the K values the % error is decreasing but after 23 it seems to saturate to 40% with a threshold 20.

References:

1. https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/
2. https://www.face-rec.org/algorithms/PCA/jcn.pdf
3. https://ieeexplore.ieee.org/document/139758
4. pythonprogramming.net/loading-images-python-opencv-tutorial/
5. https://docs.opencv.org/4.x/index.html