Eigenfaces for Recognition

Debangshu Bhattacharya (MDS201910) Swaraj Bose (MDS201936)

Avirup Chakraborty (MDS201908) Ipsita Ghosh (MDS201913)

May 1, 2020

Abstract

This project aims at emulating the works of M.Turk et. al. in "Eigenfaces for Recognition" [1]. The authors had treated the face recognition problem as an intrinsically two-dimensional recognition problem rather than requiring the recovery of three dimensional geometry, (taking advantage of the fact that faces are normally upright) . So a face may be described by a small set of 2D characteristic views. The system functions by projecting face images onto a feature space that spans the variations among known face images. The significant features are termed as "eigenfaces" as they are the eigenvectors of the training set of faces. These features do not necessarily correspond to features like nose, eyes, mouth, etc. However, the projection operation characterizes any face image as a weighted sum of the eigenface features and hence for recognition purposes we only need to compare these weights with the weights of known people.

1 Literature Review

The authors(M.Turk et. al) in the paper have drawn inspiration from Information theory to come up with this algorithm [2]. They wanted to extract the relevant information of a face image, encode it as efficiently as possible and compare with a database of such encodings for recognition. For extracting the relevant information of a face, they wanted to capture the variation in a collection of face images. In mathematical terms, they were concerned with the principal component of the distribution of faces, or the eigen vectors of the covariance matrix of the set of face images. These ”eigenfaces” efficiently encodes the variation between face images and each individual face image can thus be represented as a linear combination of the eigenfaces. Each face can also be approximated using the “best” eigen faces i.e eigenvectors corresponding to those that have the largest eigen values of the covariance matrix and which therefore account for the most variance within the set of face images. The best M' eigen faces span an M'-dimensional subspace- “face space “ – of all possible images. This approach of face recognition can be summarized as -

- Acquire an initial set of face images (training set).
- Calculate the eigen faces from the training set , keeping only the M' images that correspond to the highest eigenvalues.These M' images define the facespace.
- For any test image, project the image onto each eigenfaces of the face space and calculate the corresponding M' weights.
- Compute the distance of the image from the face space and check if it is a face at all.
- If the image is that of a face image, classify the weight patterns as a known person from the training dataset or an unknown person.

2 Data Set Used

The dataset that we used in the project is the **Yale Face Database**[1] which contains 165 grayscale images in GIF format of 15 individuals..There are 11 images per subject, one per different facial expression or configuration: center-light, with glasses, happy, left-light, with no glasses, normal, right-light, sad, sleepy, surprised, and wink. The images were of size 320 $\times$ 243 (in pixels).

[1]`http://vision.ucsd.edu/content/yale-face-database`

3 Calculating eigen faces

A image I(x,y) can be represented as a 2-D $N_1 \times N_2$ array of 8-bit intensity values. So, an image may be considered as vector of dimension $N_1 N_2$. Let the training set of face images be $\tau_1, \tau_2 ....\tau_M$, where $M = n \times p$, n : number of images per person, p : number of persons/subjects in the dataset. The average face of the set is defined by

$$\psi = \frac{1}{M} \sum_{i=1}^{M} \tau_i \tag{1}$$

The average face for our Yale Dataset is shown in Figure 1.

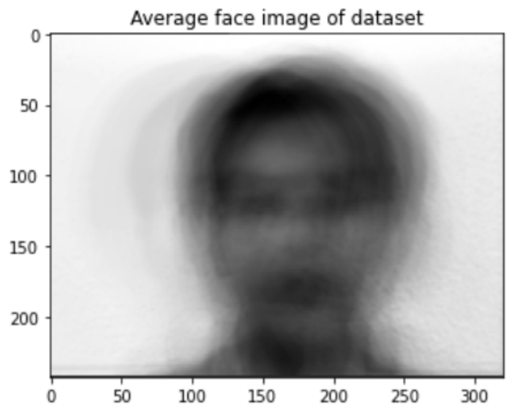


Figure 1: Average face image of Yale dataset

Since, we want to encode the variation between face images of the training dataset, we consider the deviation of each image from the average image. So each deviation image can be represented as

$$\Phi_i = \tau_i - \psi \tag{2}$$

. We can see from Figure 2 some original images and their deviation images. So our training data is now represented as $A = [\Phi_1, \Phi_2, ...\Phi_M]$ which is of dimension $N_1 N_2 \times M$

Next, we compute the covariance matrix, C, defined by:

$$C = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T \tag{3}$$

$$= AA^T$$

So, we want to find the top M' eigen vectors of C corresponding to the highest M' eigen values of C. This constitutes the face space. However, the dimension of the covariance matrix is $N_1 N_2 \times N_1 N_2$ and computing the eigen values and eigen vectors of such a huge matrix is computationally very expensive. However, we can get the eigen vectors of AA^T by taking a linear

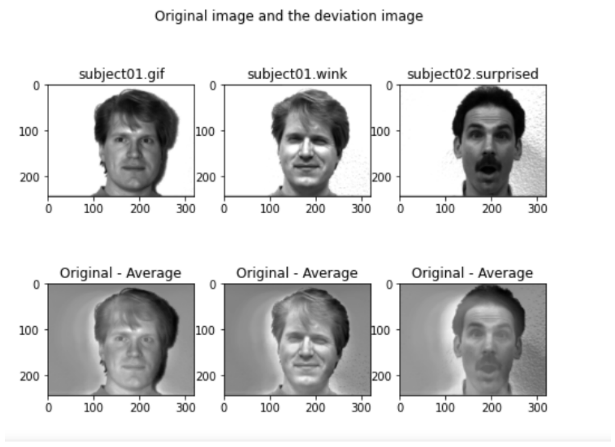


Figure 2: Original and their deviation from mean image

combination of the eigen vectors of $\boldsymbol{A^T A}$ which greatly reduces the computation. We only need to compute the eigen vectors of a $\boldsymbol{M \times M}$ matrix instead. Consider the eigenvectors $\boldsymbol{v_i}$ of $\boldsymbol{A^T A}$ such that-

$$\boldsymbol{A^T A v_i = \mu_i v_i} \tag{4}$$

Pre-multiplying both sides by A we get,

$$\boldsymbol{A A^T A v_i = \mu_i A v_i} \tag{5}$$

from which we can see that $\boldsymbol{A v_i}$ are the eigen vectors of C = $\boldsymbol{A A^T}$. Thus each eigenface $\boldsymbol{u_i}$ can be computed as

$$\boldsymbol{u_i = \sum_{k=1}^{M} v_{ik} \Phi_k, \qquad i = 1, 2, ...M} \tag{6}$$

where $\boldsymbol{v_{ik}}$ is the $\boldsymbol{k^{th}}$ component of the $\boldsymbol{i^{th}}$ eigen vector of $\boldsymbol{A^T A}$. We normalize each eigenface such that the $\boldsymbol{\|u_i\| = 1, \;\; i = 1, 2, ..., M}$. We consider the top 50 ($\boldsymbol{M'}$) eigen faces for our face space.

The top 6 computed eigenfaces of Yale dataset is shown in figure 3

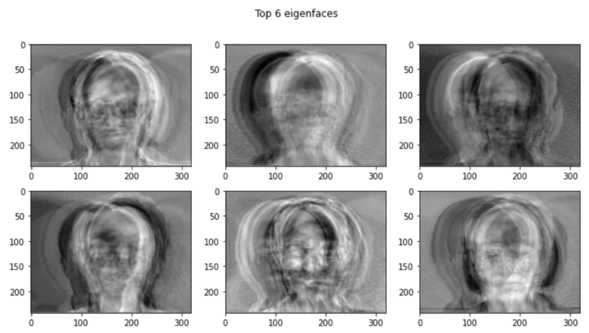


Figure 3: Top 6 eigenfaces of our dataset

4 Classifying A Face Image

For any face image τ we compute $\Phi = \tau - \psi$ where ψ is the average face image of the dataset. We project Φ on each of the eigenfaces where the projected weights are computed as

$$\omega_k = u_k^T \Phi \tag{7}$$

for k= 1, 2, ..., M'. The weights form a vector $\Omega^T = [\omega_1, \omega_2, ...\omega_{M'}]$ which describes the contribution of each eigenface in representing the face image treating the eigenfaces as basis set vectors. The vector may then be used in any standard pattern recognition approach to see which out of a predescribed set of face classes, if any best describes the face. In our approach, we classify a face image as the face class k which minimizes the euclidean distance

$$\epsilon_i = ||\Omega - \Omega_i||^2 \tag{8}$$

where Ω_i is the weights of the projection of the average face image of i^{th} class on the face space. To compute the average face image of the i^{th} class, if necessary, we may even consider a few number of examples as available (as few as 1). The distance of an image from the face space can be calculated as

$$\epsilon^2 = ||\Phi - \Phi_f||^2 \tag{9}$$

where Φ_f is the projection of the image on the face space and can be represented as: $\Phi_f = \sum_{i=1}^{M'} \omega_i u_i$.

We have the distance of the image from face space and the distance of the image from each class computed as shown above. To classify the image, we need to come up with a face threshold value(θ_e) and class threshold values vector(θ) which is a vector of same shape as the number of classes(K). The computation of these thresholds were not given in the paper and we came up with our own implementation of it and is given in detail in Experiments and Results section.

To sum up the face classification algorithm using eigenfaces, for a test face image we first compute the distance of the image from the face space. If the distance of the image is less than threshold θ_e, then the image is that of a face as it is close to the face space. Then we compute the distance of the image from each known class label and find the possible classes where $\epsilon_k < \theta_k$ and we classify the image as the one with minimum ϵ_k. If no possible candidates exist, then we classify it as a face image of an unknown person. If the image has a higher distance from the face space than the threshold θ_e, we confirm that the image is not that of a face.

5 Experiments and Results

In the first part of our experiment, we have considered the entire dataset for computing the eigenfaces, i.e, 11 images of each person totalling 165 images. Subsequently, we tested on ”normal” images of each person (15 of them in total) for testing the accuracy of classification. We found that the classifying mechanism was able to correctly classify all the 15 images belonging to the test set. Some results are shown in details in **Appendix Section 8.1**.

In the second part of the experiment, we had a mutually exclusive train set for computing eigenfaces and a separate unseen test set for predicting the class of the faces. As before, we took the ”normal” images of each person (15 of them in total) as our test set. However, this time for computing the eigen faces, our train set consisted of the entire dataset images **minus** the test set images. So our training set consisted of 10 images of each class totalling 150 images. We found that the classifying mechanism was able to correctly classify 14 out of the 15 test set images.

For both these experiments, we tried out different values of the number of top eigenfaces ($\boldsymbol{M'}$) to consider as the face space. Considering more number of top eigenfaces gives a better representation of a face image but there is the cost of added computational complexity. So, there is a tradeoff here between the computational cost and classification accuracy. We found out that considering the **top 50** eigenfaces gave sufficiently good results with respect to accuracy. It is also computationally effective as we are only considering about $\boldsymbol{1/3^{rd}}$ of the total eigenfaces. So for our experiments, we considered the top 50 eigenfaces as our facespace.

The projections of the first three images belonging to the test dataset (corresponding to the first three subjects) on to the face space are shown in Figure 4.

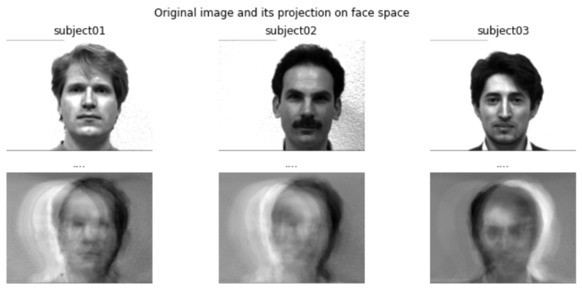


Figure 4: Projection on to the face space

For calculating the face threshold value ($\boldsymbol{\theta_e}$) we planned to use the maximum distance of an image from the face space out of all the images in the training set. For calculating the class threshold value vector($\boldsymbol{\theta = [\theta_1, \theta_2, ..., \theta_K]}$), for each class label k, we planned to take the maximum distance of the image from that class label for all images of that class label in the training set.

However, after some experimentation, we observed that the threshold values do not vary much if we choose 2-3 images instead of all of them, per class. Thus to reduce computations and run time complexity we have calculated the class thresholds by taking only 3 (”no-glasses”, ”happy” and ”sad”) images for each class, i.e, a total of 45 images. So, instead of the whole training set we did our computations for the threshold values on these 45 images instead and got replicable results.

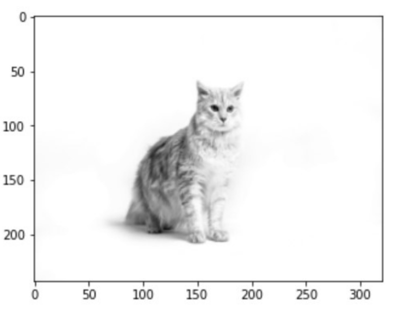
However, in the second part of the experiment where the test set is mutually exclusive, we took a different approach to calculate the class label threshold values. Instead of taking the maximum distance from each class label, we take the outlier detection approach and select

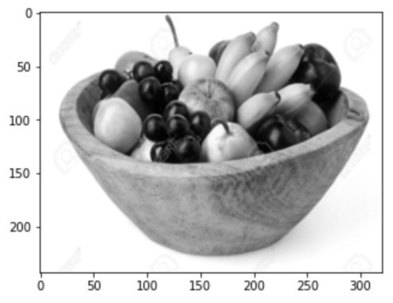
$$\theta_k = Q_3 + 4.5 * IQR$$

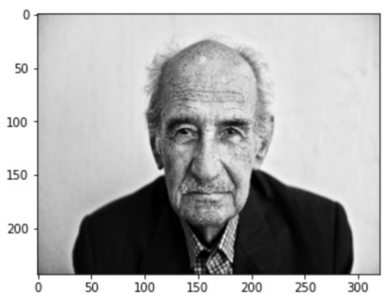
where Q_3 is the third quartile value and IQR is the inter quartile range value for distance values from that class label. The motivation behind taking this approach is considering that face image, whose distance from a particular class is above its threshold value, an outlier corresponding to that class. Here also, for computing these threshold values, we are not considering the whole dataset but only 3 images (’wink’, ’sad’, ’sleepy’) from each class label, i.e, a total of 45 images. We have seen that this approach gives better results than just considering the maximum out of possible class label distances.If we were considering the maximum possible distance from class label as the threshold value, then the best classification accuracy we achieved was classifying 13 test set images correctly out of 15. Also, in the misclassified images, there were a lot more number of possible candidates being considered. But using this approach, we are able to correctly classify 14 out of the 15 test set images. Even if there is a misclassification the model actually considers the true label in its set of possible candidates and the number of possible candidates are also fewer compared to the other method. So, even when the model predicts wrong we are not off by much. For example, in our test set the image with **true label 14** is being predicted as **predicted label 7**. However, the possible candidates being shown are **6,7,8 and 14** and so we are not significantly wrong.

5.1 Some other images we tested on

We also downloaded a few more images from Google, preprocessed them to grayscale and adjusted the dimensions to match those of the Yale dataset and observed that the algorithm was able to correctly classify the images. Some of those images are as shown in Figure 5. A detailed explanation for each of the images here with the computed distance values and the threshold values are given in **Appendix Section 8.1**.


(a) True : Not a face
Predicted : Not a face


(b) True : Not a face
Predicted : Not a face


(c) True : Unknown face
Predicted : Unknown face

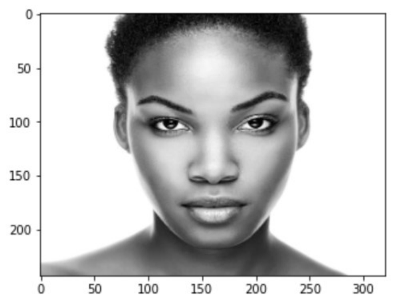
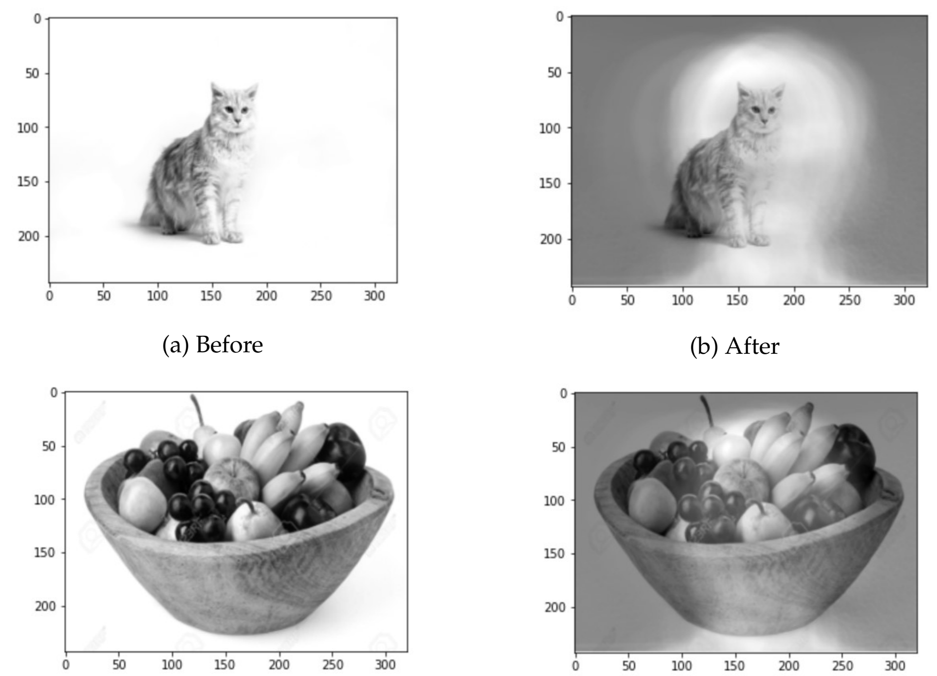

(d) True : Unknown face
Predicted : Unknown face

Figure 5: True and predicted labels of unknown and non face images

6 Some other Findings

- We found that for non face images, subtracting the average face image (computed from the training images), results in a ″halo″ on the deviation image as shown in Figure 6. This resulted in the images to be classified as face images as the ″halo″ structure resembles that of a face and hence reduces the distance from the face space. To counter this issue, for a non face image we did not subtract the average image before checking its distance from the face space, while for a face image, we subtracted the average image and then computed its distance from the face space.
- In the paper, the dataset they used primarily consisted of all images taken in same resolution and the subjects had the same upright posture. However, in our dataset there is a wide variation of emotions for each subject. So, we have seen that while considering the projection of average face image of a class (Ω_i) in equation 8, it is better to consider lower number of images with less emotions (like ″centerlight″ or ″noglasses″) for the average image computation, to have a better classification accuracy.



(a) Before (b) After

(c) Before (d) After

Figure 6: Images before and after subtracting the average face image

7 Conclusion and Future Works

Thus, from the above results and findings we can conclude that we have successfully managed to emulate the works of M.Turk et. al. in ”Eigenfaces for Recognition” [1]. The process of using eigenfaces to check the presence of a face in an image and classify it as known or unknown (w.r.t some images used in training) has been illustrated in the project.

For future work, one may further go on to detect the face from an image by drawing a bounding box around the face. This can be done by dividing the face image into smaller sub-images and running the same algorithm on each sub-image to see if there is a face in the grid. Then the sub-image with the minimum distance from the face space can be used as the bounding box for the face. Another future work that can be done is updating the eigenfaces when an unknown image occurs multiple times and incorporating that unknown class into a known label for future recognition.

References

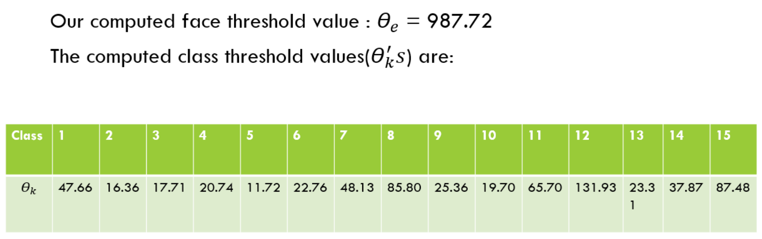
[1] M. Turk and A. Pentland, “Eigenfaces for recognition,” *Journal of cognitive neuroscience*, 1991.

[2] M. Turk and A. Pentland, “Face recognition using eigenfaces,” in *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, 1991, pp. 586–587.

8 Appendix

8.1 Some results in detail

We have shown the results that we observed by running our algorithm and some other observations we made.



Our computed face threshold value : $\theta_e = 987.72$

The computed class threshold values($\theta_k' s$) are:

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
θ_k	47.66	16.36	17.71	20.74	11.72	22.76	48.13	85.80	25.36	19.70	65.70	131.93	23.31	37.87	87.48

Figure 7: The threshold for face space and different classes

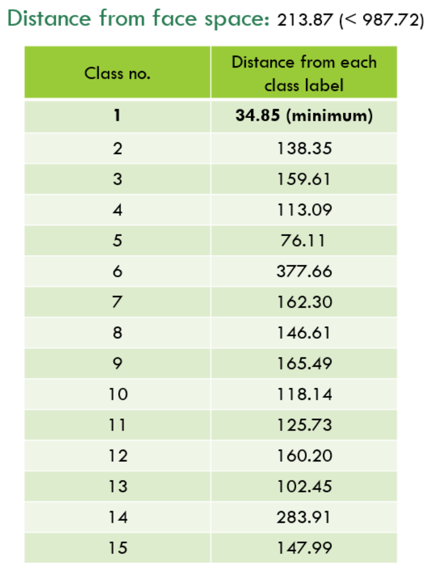
In our algorithm, for a particular image, we have first calculated the distance from face space. If the distance is greater than the threshold of the face space ($\boldsymbol{\theta_e}$), we classify it as a non-face image. If it is less than the threshold of the face space ($\boldsymbol{\theta_e}$), then the image is recognised as a face image. Further, by calculating the distance from each class, if the distance exceeds the class thresholds of all classes then it is classified as an unknown face. Otherwise we denote the classes as candidate classes for which the distance is less than the class threshold. We then classify the image as a face of that class from which it has the minimum distance among the candidate classes.
Now we will elaborately explain some of the outputs from our experiment in the figures below :

Normal Face 1

True: Face of Class 1

$34.85 < \theta_1 = 47.66$

Prediction: Face of Class 1



Distance from face space: 213.87 (< 987.72)

Class no.	Distance from each class label
1	**34.85 (minimum)**
2	138.35
3	159.61
4	113.09
5	76.11
6	377.66
7	162.30
8	146.61
9	165.49
10	118.14
11	125.73
12	160.20
13	102.45
14	283.91
15	147.99

Figure 8

Normal Face 8

True: Face of Class 8

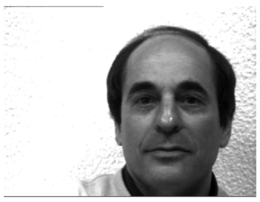
$24.64 < \theta_8 = 85.80$

Prediction: Face of Class 8

Distance from face space: 302.82 (< 987.72)

Class no.	Distance from each class label
1	180.15
2	268.82
3	42.20
4	232.70
5	186.18
6	311.65
7	51.90
8	**24.64 (minimum)**
9	66.43
10	105.21
11	223.71
12	274.47
13	173.55
14	197.55
15	43.04

Figure 9



Normal Face 12

True: Face of Class 12

$0.00 < \theta_{12} = 131.93$

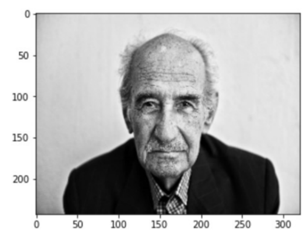
Prediction: Face of Class 12

Distance from face space: 483.23 (< 987.72)

Class no.	Distance from each class label
1	133.27
2	94.33
3	275.82
4	140.03
5	140.43
6	445.02
7	276.10
8	262.85
9	269.88
10	230.37
11	144.15
12	**0.00 (minimum)**
13	183.50
14	374.95
15	265.11

Figure 10

Similarly we were able to detect unknown faces. The following are a few unknown face images we downloaded from google and tested our algorithm on.



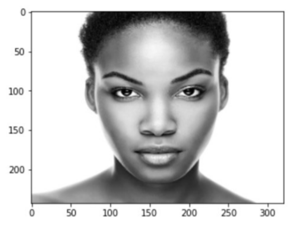
True: Unknown face

Computed threshold from the training set = 987.72
Distance from face space = 346.14 (<987.72)

Prediction: Unknown Face

Predicted Distance to classes	Epsilon k thresholds(θ_k)
209.07	47.66
255.22	16.36
223.20	17.71
235.40	20.74
208.91	11.72
229.82	22.76
212.67	48.13
198.50	85.80
191.56	25.36
197.84	19.70
201.54	65.70
239.04	131.93
228.91	23.31
216.99	37.87
196.25	87.47

Figure 11



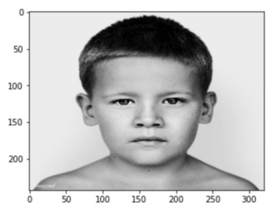
True: Unknown face

Computed threshold from the training set = 987.72
Distance from face space = 259.08 (<987.72)

Prediction: Unknown Face

Predicted Distance to classes	Epsilon k thresholds(θ_k)
155.01	47.66
244.69	16.36
130.94	17.71
214.22	20.74
166.05	11.72
248.51	22.76
132.77	48.13
107.74	85.80
96.82	25.36
135.21	19.70
200.50	65.70
247.02	131.93
177.03	23.31
167.89	37.87
100.50	87.47

Figure 12



True: Unknown face

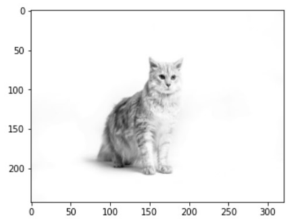
Computed threshold from the training set = 987.72
Distance from face space = 173.55 (<987.72)

Prediction: Unknown Face

Predicted Distance to classes	Epsilon k thresholds(θ_k)
142.81	47.66
215.22	16.36
156.41	17.71
188.48	20.74
165.75	11.72
264.23	22.76
154.54	48.13
130.13	85.80
138.59	25.36
129.98	19.70
162.34	65.70
213.92	131.93
176.50	23.31
199.19	37.87
122.35	87.48

Figure 13

We were also able to detect non-face images. The following are non-face images.



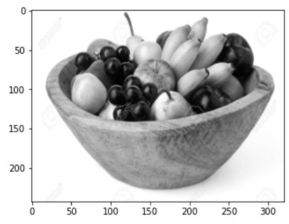
True: Not a face

Computed threshold from the training set = 987.72

Distance from face space = 1368.25 (> 987.72)

Prediction: Not a face

Figure 14



True: Not a face

Computed threshold from the training set = 987.72

Distance from face space = 1106.10 (>987.72)

Prediction: Not a face

Figure 15

8.2 Code Links

The google colab jupyter notebook click on links are given below:

- **Experiment 1**: Considering whole dataset for computing eigenfaces.
- **Experiment 2**: Considering a mutually exclusive train and test set and computing eigenfaces from training set images only.