**Name: Brijesh Rohit**
**Admission no: U19CS009**

# CV-ASSIGNMENT-03

Apply PCA on the dataset of face images given. There are 40 images in the dataset. Experiment by selecting a different number of principle components while performing dimensionality reduction. Reconstruct few of the original images using the dimensionality reduced data and observe the reconstruction capability. Show original image and the reconstructed image side by side in the submission pdf. Also mention the number of principle components used.

**CODE :**

```
# U19CS009
# BRIJESH ROHIT

import numpy as np
from PIL import Image
import matplotlib.image as mplib
import matplotlib.pyplot as plt
import random
plt.rcParams.update({'figure.max_open_warning': 0})

def Extract(Img_mean,L,e,N_comp):
    # Sorting according to eigen values
    Indices = np.argsort(L)[::-1]
    L_sorted = L[Indices]
    e_sorted = e[:,Indices]

    # Selecting 1st N_comp eigen vectors
    E = e_sorted[:,0:N_comp]

    # Calulating Reduced Image Matrix
    Img_Red = np.dot(E.transpose() , Img_mean.transpose()
```

```
).transpose()
    return Img_Red,E

def eigen_Cal(Img_mean ):
    # Calculating Covariance
    cov_mat = np.cov(Img_mean , rowvar = False)

    # Calculating Eigen Values and Eigen Vectors
    L , e = np.linalg.eigh(cov_mat)
    return L,e

img=[]
for x in range(1,41):
    X=Image.open("Images for Assignment/f"+str(x)+".pgm")
    X=np.asarray(X)
    X=X.flatten()
    img.append(X)
img_arr=np.array(img)
Img_mean = img_arr - np.mean(img_arr , axis = 0)

# GETTING EIGEN VALUES AND VECTORS
L,e=eigen_Cal(Img_mean)

# GETTING REDUCED IMAGE FOR COMPONENTS COUNT = 100
P1,E1=Extract(Img_mean,L,e,100)
final1=np.dot(P1,E1.transpose())

# GETTING REDUCED IMAGE FOR COMPONENTS COUNT = 200
P2,E2=Extract(Img_mean,L,e,200)
final2=np.dot(P2,E2.transpose())

print("Matrix Size of 40 images of dimension [112,92] before
PCA : ",img_arr.shape,"\n")
print("Matrix Size of 40 images of dimension [112,92] after PCA
```

```
: ",P1.shape,", ",P2.shape,"\n")

# PLOTTING IMAGE
plt.figure(figsize=(20,20))
for x in range(0,40):
    Img_org=img_arr[x].reshape([112,92])
    Img_red1=final1[x].reshape([112,92])
    Img_red2=final2[x].reshape([112,92])
    f, axarr = plt.subplots(1,3)
    axarr[0].set_title("Original Image")
    axarr[1].set_title("(100)")
    axarr[2].set_title("(200)")
    axarr[0].imshow(Img_org)
    axarr[1].imshow(Img_red1)
    axarr[2].imshow(Img_red2)

# GETTING REDUCED IMAGE FOR COMPONENTS COUNT = 1000
P3,E3=Extract(Img_mean,L,e,1000)

final3=np.dot(P3,E3.transpose())
# GETTING REDUCED IMAGE FOR COMPONENTS COUNT = 5000
P4,E4=Extract(Img_mean,L,e,5000)
final4=np.dot(P4,E4.transpose())

print("Matrix Size of 40 images of dimension [112,92] before
PCA : ",img_arr.shape,"\n")
print("Matrix Size of 40 images of dimension [112,92] after PCA
: ",P3.shape,", ",P4.shape,"\n")

# PLOTTING IMAGE
plt.figure(figsize=(20,20))
for x in range(0,40):
    Img_org=img_arr[x].reshape([112,92])
    Img_red1=final3[x].reshape([112,92])
```

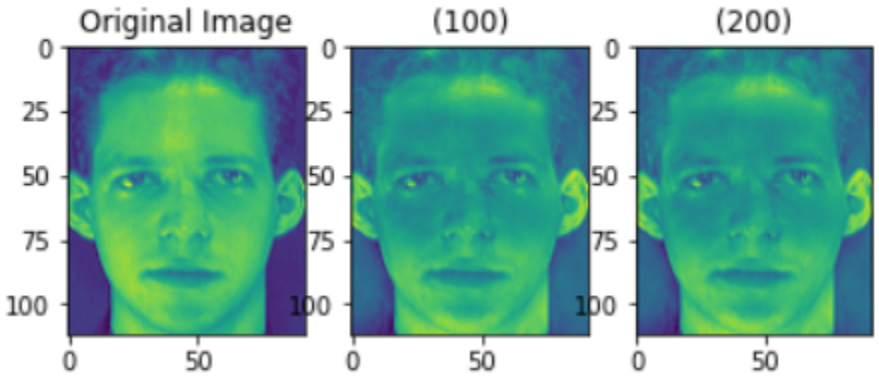
```
    Img_red2=final4[x].reshape([112,92])
    f, axarr = plt.subplots(1,3)
    axarr[0].set_title("Original Image")
    axarr[1].set_title("(1000)")
    axarr[2].set_title("(5000)")
    axarr[0].imshow(Img_org)
    axarr[1].imshow(Img_red1)
    axarr[2].imshow(Img_red2)
```

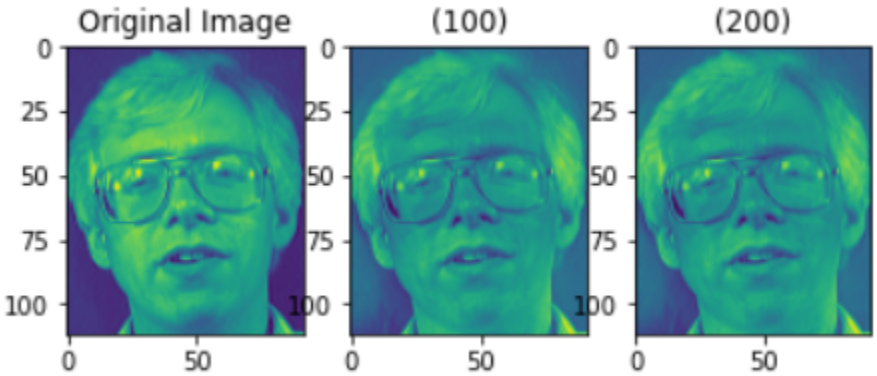
**OUTPUT : For number of principle component = 100 and 200**

```
Matrix Size of 40 images of dimension [112,92] before PCA :  (40, 10304)

Matrix Size of 40 images of dimension [112,92] after PCA :  (40, 100) ,  (40, 200)
```

<Figure size 1440x1440 with 0 Axes>



**OUTPUT : For number of principle component = 1000 and 5000**

```
Matrix Size of 40 images of dimension [112,92] before PCA :  (40, 10304)

Matrix Size of 40 images of dimension [112,92] after PCA :  (40, 1000) ,  (40, 5000)
```

<Figure size 1440x1440 with 0 Axes>