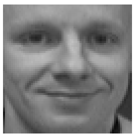
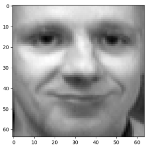
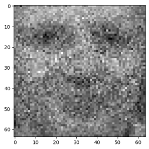
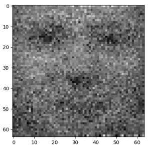
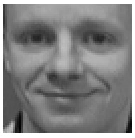
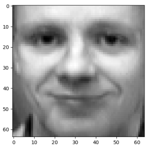
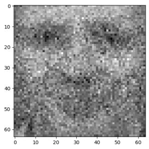
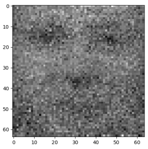
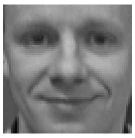
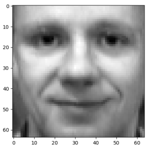
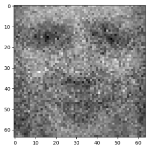
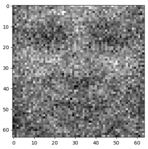
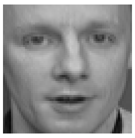
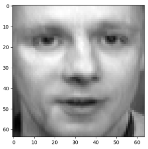
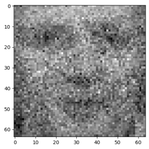
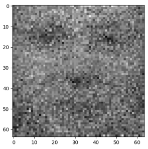
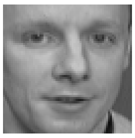
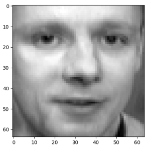
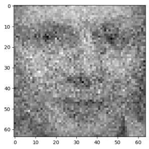
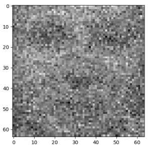
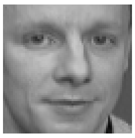
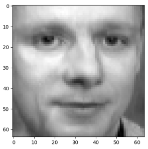
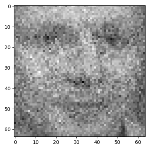
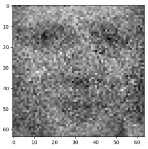
Submission for Programming Assignment 1

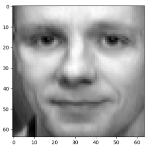
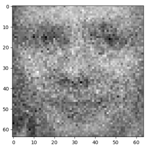
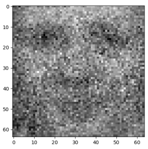
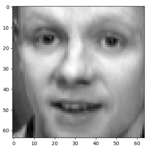
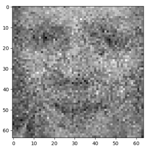
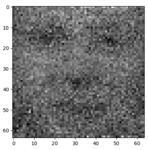
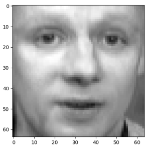
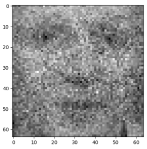
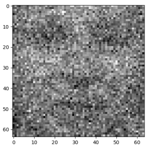
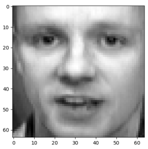
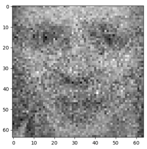
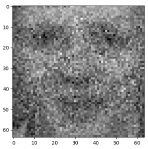
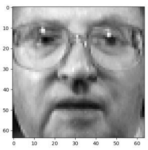
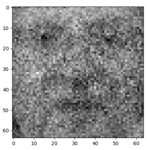
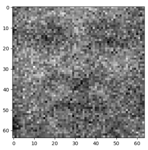
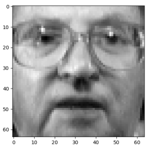
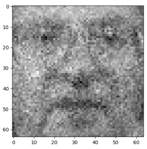
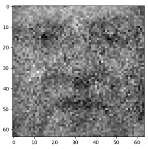
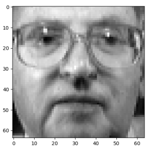
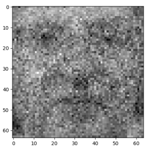
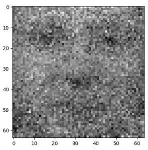
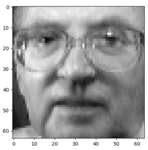
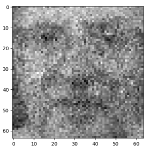
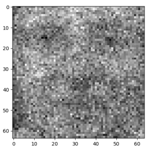
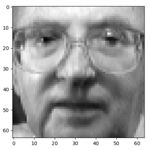
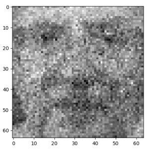
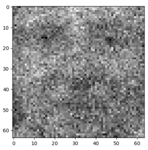
CS 427: Mathematics for Data Science, Autumn 2020-21

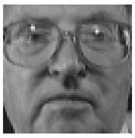
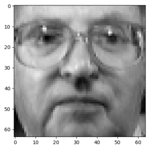
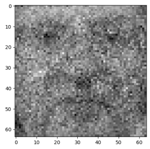
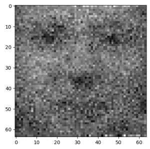
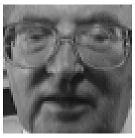
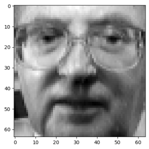
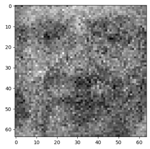
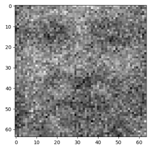
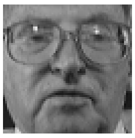
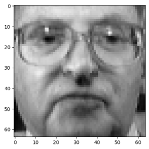
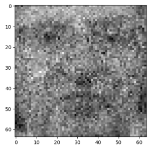
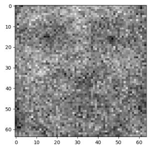
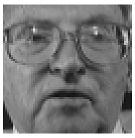
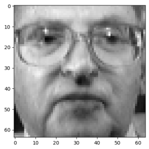
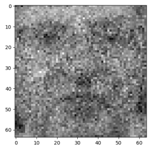
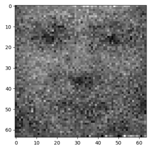
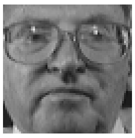
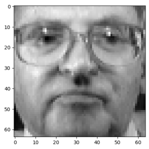
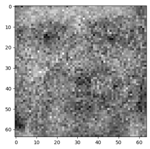
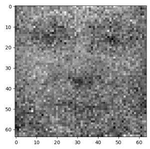
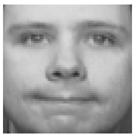
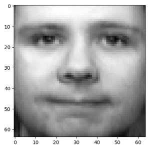
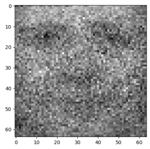
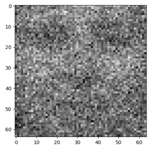
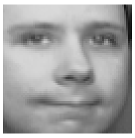
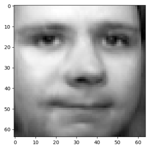
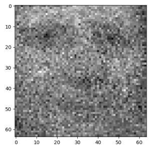
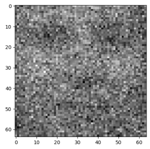
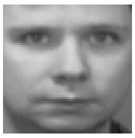
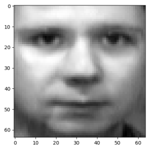
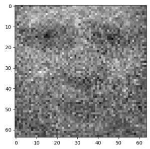
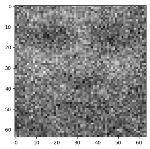
Group 9: K. Sai Anuroop (170030035), Rohan Savakar (170030033)

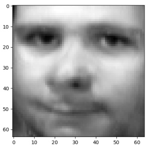
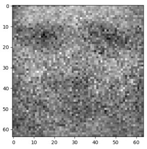
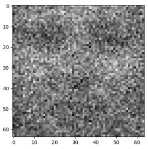
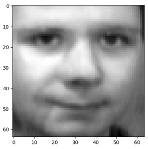
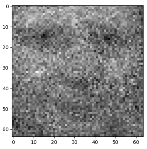
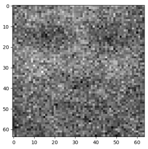
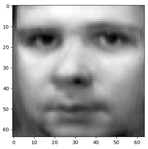
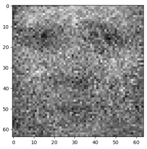
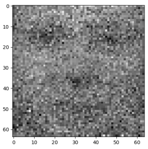
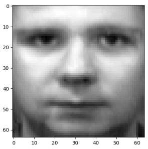
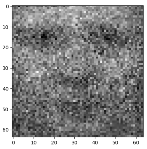
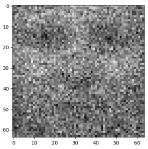
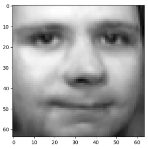
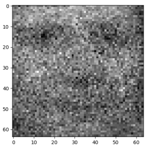
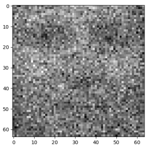
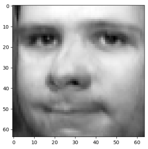
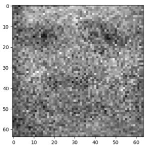
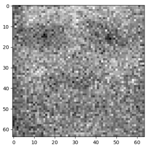
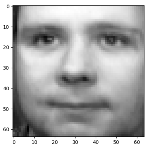
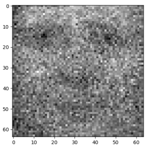
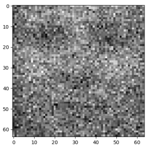
November 22, 2020

The code performs Principal Component Analysis on the set of images provided to us from Olivetti faces dataset. It then randomly chooses an image, reconstructs it using 4096 components, x components and $< x$ components, where x is provided as an input argument by the user. We interpret x as the minimum number of components required to reconstruct the image such that the face of the person is discernible.
Below is an exhaustive list of all the 30 images provided to us along with their reconstructed counterparts as per the scheme mentioned above.

Original	4096	x	$< x$
 0.png	 -	 3100	 353
 1.png	 -	 3250	 476
 2.png	 -	 3300	 1220
 3.png	 -	 3250	 476
 4.png	 -	 3500	 817
 5.png	 -	 3650	 2990

Original	4096	x	$< x$
 6.png	 -	 3650	 2990
 7.png	 -	 3250	 476
 8.png	 -	 3300	 1220
 9.png	 -	 3350	 3105
 10.png	 -	 3000	 1331
 11.png	 -	 3650	 2990
 12.png	 -	 3050	 541
 13.png	 -	 3200	 2098
 14.png	 -	 3200	 2098

Original	4096	x	$< x$
 15.png	 -	 3100	 353
 16.png	 -	 3000	 1331
 17.png	 -	 3200	 2098
 18.png	 -	 3100	 353
 19.png	 -	 3050	 541
 20.png	 -	 3000	 1331
 21.png	 -	 2700	 1587
 22.png	 -	 2700	 1587

Original	4096	x	$< x$
 23.png	 -	 3000	 1331
 24.png	 -	 2700	 1587
 25.png	 -	 2750	 523
 26.png	 -	 2900	 1744
 27.png	 -	 2900	 1744
 28.png	 -	 2850	 2450
 29.png	 -	 2700	 1587

3005 is the average of all the values of x listed in the above table, which we consider to be the value of x that is required to discern the faces, according to our discretion.

```
"""
This code performs PCA on the set of images provided in the format <number>.png,
reconstructs a randomly chosen image from the input set using [4096, k, <k] Principal Components,
and saves in ./output/img<number> directory.
Note that this code has to be placed in the same directory as the set of input images (or else change the path in the code accordingly)
"""
import os
import sys
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.decomposition import PCA
from glob import iglob
from numpy import mean
from numpy import cov
from numpy.linalg import eigh
from pathlib import Path

# checking if sufficient input arguments are provided and if provided, are correct or not
if(len(sys.argv)<2 or int(sys.argv[1])>4096 or int(sys.argv[1])<1):
    print("Usage: python3 pca3.py <num_principal_components_to_retain>")
    print("Note: 1 <= num_principal_components_to_retain <= 4096")
    exit(1)

# dataframes to store images of a person’s face
faces_0 = pd.DataFrame([])
faces_1 = pd.DataFrame([])
faces_2 = pd.DataFrame([])
all_faces = pd.DataFrame([])

# convert 64*64 images to a pandas dataframe of 4096 size each
for path in iglob("*.png"):
    img=mpimg.imread(path)
    # print(int(’’.join(path.split(’.’)[0])))
    face = pd.Series(img.flatten(),name=path)
    all_faces = all_faces.append(face)
    if int(int(path[0:path.find(’.’)])/10)==0:
        # print(int(path[0:path.find(’.’)]))
        faces_0 = faces_0.append(face)
    elif int(int(path[0:path.find(’.’)])/10)==1:
        # print(int(path[0:path.find(’.’)]))
        faces_1 = faces_1.append(face)
    else:
        # print(int(path[0:path.find(’.’)]))
        faces_2 = faces_2.append(face)

# uncomment to choose a random person to perform PCA on the set of images of "that" person
"""
# choose a random person to perform PCA
face_num = random.randint(0,((len(faces_0)+len(faces_1)+len(faces_2))/10)-1)
# choose a random image of the person to compare with
which_face = random.randint(0,10)
fit_which_face = faces_0
if face_num == 0:
    fit_which_face = faces_0
    my_face = faces_0[which_face]
elif face_num == 1:
    fit_which_face = faces_1
    my_face = faces_1[which_face]
else:
    fit_which_face = faces_2
    my_face = faces_2[which_face]
"""
# sort indices of images
all_faces = all_faces.assign(indexNumber=[int(’’.join(i.split(’.’)[0])) for i in all_faces.index])
all_faces.sort_values([’indexNumber’], ascending = [True], inplace = True)
all_faces.drop(’indexNumber’, 1, inplace = True)

temp = all_faces # temp: 30 x 4096
temp = temp.values
mean_matrix = mean(temp.T, axis=1)
```

```python
centred_matrix = temp - mean_matrix
cov_matrix = cov(centred_matrix.T) # finding covariance matrix
values, vectors = eigh(cov_matrix) # finding eigenvectors (Principal Components) of covariance matrix
vectors = np.real(vectors) # vectors: 4096 x 4096

# sort eigenvectors (Principal Components) based on their eigenvalues in descending order
idx = values.argsort()[::-1]
values = values[idx]
vectors = vectors[:,idx]

retention_vector = [4096, int(sys.argv[1]), random.randint(0,int(sys.argv[1]))] # array storing number of eigevectors (Principal Components) to be used while reconstructing image
# choose an image to reconstruct and create an output directory for the same
reconstruct_img_num = random.randint(0,29)
# reconstruct_img_num = 18
filename = "./output/img"+str(reconstruct_img_num)
Path(filename).mkdir(parents=True, exist_ok=True)

for k in retention_vector: # reconstruct the chosen image using [4096, k, <k] eigevectors (Principal Components)
    # retain only k eigevectors (Principal Components)
    retained_values = values[:k] # choose this number k
    retained_vectors = vectors[:k] # retained_vectors: k x 4096

    # Summary of matrix dimensions
    # temp: 30 x 4096
    # retained_vectors: k x 4096
    # projected_matrix: 30 x k
    # recon_matrix: 30 x 4096

    # reconstruct images
    projected_matrix = temp.dot(retained_vectors.T) # projected_matrix: 30 x k
    recon_matrix = projected_matrix.dot(retained_vectors)+mean_matrix # recon_matrix: 30 x 4096
    plt.imshow(recon_matrix[reconstruct_img_num].reshape(64,64),cmap="gray")
    plt.savefig(filename+"/"+str(k)+"pc_recon_"+str(reconstruct_img_num)+".png")

```

Python code to perform PCA on images

Scan this QR code to access the GitHub repository of my homweork solutions at
https://github.com/ksanu1998/MDS_HW_Solutions