Project 2

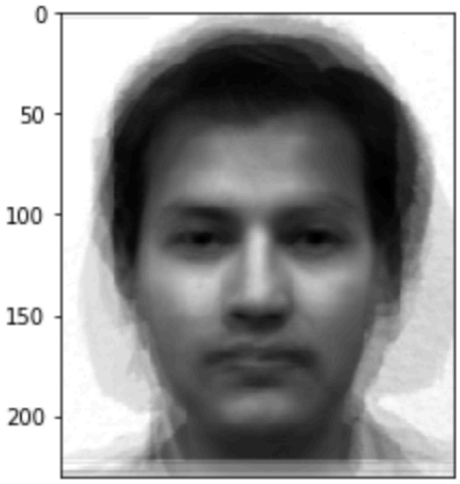
Members:

- Qi Yin (qy652)
- Mengxi Wu (mw4355)

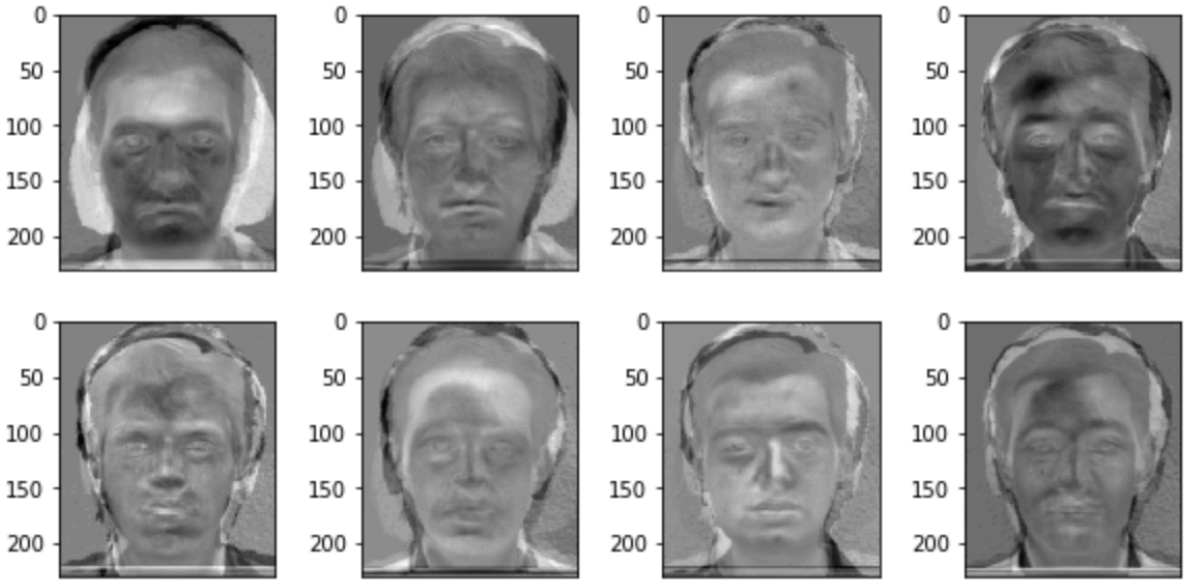
Running Instruction

- Install opencv-python, numpy, matplotlib, Pillow
- Running the program with Jupyter Notebook

Mean Face

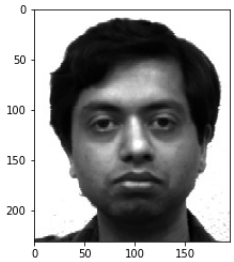


Eigenfaces



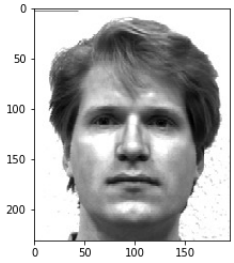
Eigenface coefficients for Train

- 1st Trainning Image: subject10.normal.jpg



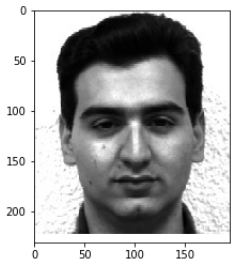
```
[-2.57010807e+07  5.31036358e+07 -2.29862430e-09 -7.67395071e+07
  1.13657321e+07 -2.13665227e+07 -4.79155988e+07  8.39149947e+06]
```

- 2nd Trainning Image: subject01.normal.jpg



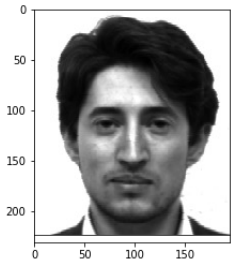
```
[-7.58393182e+07  9.86591897e+07  1.15711781e-09  7.35308451e+07
  2.03390545e+07 -2.09059360e+07  1.34689494e+07 -4.95663886e+06]
```

- 3rd Tainning Image: subject15.normal.jpg



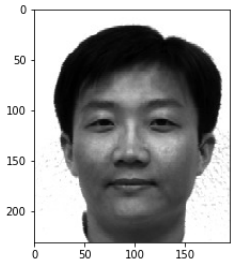

```
[ 1.53443135e+08 -4.12463935e+07  1.93141931e-09  5.17636090e+07
 -1.14888738e+07 -1.73505791e+05 -4.28493981e+07 -2.98320729e+07]
```

- 4th Tainning Image: subject03.normal.jpg




```
[ 6.79094869e+07  2.86096157e+07  2.17088613e-08  2.11976451e+07
 -9.00930120e+06  6.14469992e+07 -3.38368115e+06  4.32998677e+07]
```

- 5th Tainning Image: subject14.normal.jpg

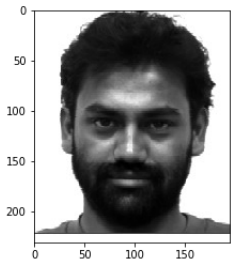



```
[-5.48035678e+07  7.56177219e+07  2.01051866e-08 -4.07897743e+07
 -2.52009615e+07  9.97235825e+06  2.99136079e+07 -2.68289508e+07]
```

- 6th Tainning Image: subject11.normal.jpg

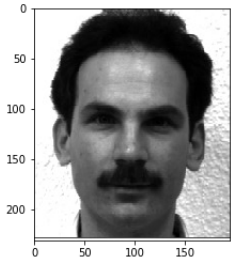

```
[-3.09965508e+08 -1.01569225e+08 -3.13860135e-08  7.23282904e+06
  3.43527729e+05  1.38164759e+07 -4.01893283e+06 -6.69097274e+05]
```

- 7th Tainning Image: subject07.normal.jpg




```
[ 1.59365384e+08 -5.98976188e+07 -1.62704394e-08 -3.62392288e+07
  2.39421030e+07  2.99666082e+07  3.11659201e+07 -1.50158366e+07]
```

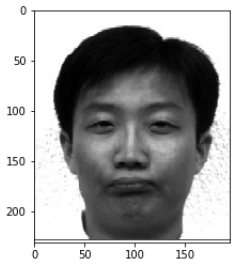
- 8th Tainning Image: subject02.normal.jpg




```
[ 8.55914694e+07 -5.32769258e+07  5.05249219e-09  4.35819945e+04
 -1.02912808e+07 -7.27564771e+07  2.36191334e+07  2.56112293e+07]
```

Results for Test

- 1st Testing Image: subject14.sad.jpg



```
[-3.09557722e+07  6.21419290e+07  1.73357689e-08 -4.11282105e+07
 -1.89530184e+07  9.12193903e+06  3.02355547e+07 -1.57609187e+07]
```

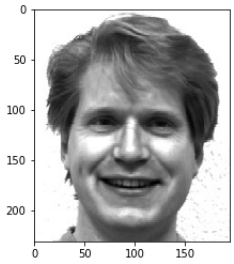
The recognition result is the subject14.normal.jpg

- 2nd Testing Image: subject11.happy.jpg

```
[-2.91280480e+08 -1.07981032e+08 -2.93551396e-08  8.07984970e+06
 -3.16856693e+06  8.84253231e+06  2.77433359e+06 -1.23549801e+06]
```

The recognition result is the subject11.normal.jpg

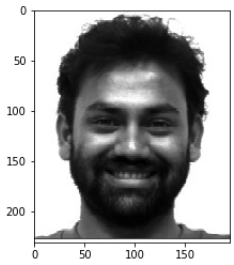
- 3rd Testing Image: subject01.happy.jpg



```
[-2.84226819e+07  6.21657618e+07  1.82785339e-09  5.58402336e+07
  1.39551196e+07 -2.39919521e+07  1.56038461e+07  4.88745009e+05]
```

The recognition result is the subject01.normal.jpg

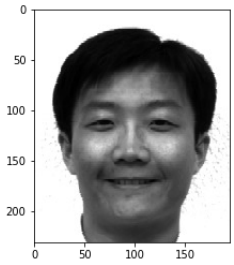
- 4th Testing Image: subject07.happy.jpg



```
[ 1.03893380e+08 -9.15597753e+06  1.96306999e-09 -2.31395182e+07
  9.78874352e+05  3.10262255e+06  4.00701970e+06 -6.01476791e+06]
```

The recognition result is the subject07.normal.jpg

- 5th Testing Image: subject14.happy.jpg



```
[-3.72003742e+07  7.23552596e+07  1.41319597e-08 -2.01072900e+07
 -1.10391535e+07  1.16749219e+07  3.12504845e+07 -1.93984877e+07]
```

The recognition result is the subject14.normal.jpg

Recoginition Accuracy

The accuracy is 100%.

Code

```
import matplotlib.image as mpimg
from PIL import Image
import numpy as np
import os
import cv2

# read images
faces_train = [] # store all train image data
faces_test = [] # store all test image data
```

```
train_dir ="Face dataset/Training"
test_dir ="Face dataset/Testing"

# this function is for read image,the input is directory name
def read_directory(directory_name):
    images = []
    # this loop is for read each image in this foder,directory_name is the foder
name with images.
    for filename in os.listdir(r"./"+directory_name):
        #print(filename) #just for test
        #img is used to store the image data
        img = cv2.imread(directory_name + "/" + filename)
        images.append(img)
    return images

faces_train = read_directory(train_dir)
faces_test = read_directory(test_dir)



## check the size of image
print(f"The size of Training dataset is: {len(faces_train)} ")
print(f"The size of Test dataset is: {len(faces_test)} ")
print(f"The shape of image is: {faces_train[0].shape} ")


# use matplotlib to display images
for faces in [faces_train, faces_test]:
    plt.figure(figsize=(20,10))
    for i in range(1,len(faces)+1):
        plt.subplot(2,8,i)
        plt.imshow(faces[i-1])
        plt.xticks([])
        plt.yticks([])
    plt.show()

# prepocess training dataset
faces_train_p = np.array(faces_train) # (8, 231, 195, 3)
n_samples, h, w, _ = faces_train_p.shape
npix = h*w
faces_train_p = faces_train_p[:,:,:,0] # (8,231,195)
faces_train_p = faces_train_p.reshape(n_samples, npix) # (8,45045)


def plt_face(x):
```

```
    global h,w
    plt.imshow(x.reshape((h, w)), cmap=plt.cm.gray)
    plt.xticks([])

plt.figure(figsize=(10,5))
for i in range(n_samples):
    plt.subplot(2,4,i+1)
    plt_face(faces_train_p[i])
plt.show()

# preprocess test data
faces_test_p = np.array(faces_test) # (4, 231, 195, 3)
n_samples, h, w, _ = faces_test_p.shape
npix = h*w
faces_test_p = faces_test_p[:,:,:,0]  # (5,231,195)
faces_test_p = faces_test_p.reshape(n_samples, npix) # (5,45045)

plt.figure(figsize=(10,5))
for i in range(n_samples):
    plt.subplot(1,n_samples,i+1)
    plt_face(faces_test_p[i])
plt.show()

# compute mean face
mean_face = faces_train_p.mean(axis=0)
#print(f"The size of mean face is {mean_face.shape}")
mean_face = mean_face.reshape(-1,1).transpose()
print(f"The shape of mean face is {mean_face.shape}")
print(mean_face)
# print mean face
plt_face(mean_face)
plt.savefig("Mean_face.jpg")
plt.show()

# subject mean face
faces_train_centered = faces_train_p - mean_face # equals A.T

# construct L matrix
L = np.dot(faces_train_centered, faces_train_centered.transpose())

# eigen decomposition get eigenvetor of L
eigenvalue, V= np.linalg.eig(L) # equals V
U = np.dot(faces_train_centered.transpose(), V)

print(f'The size of processed traning dataset A is
{faces_train_centered.transpose().shape}')
```

```
print(f'The size of L is {L.shape}')
print(f'The size of V is {V.shape}')
print(f'The size of eigenvector of C is {U.shape}')

# display 8 eigenvectors
def plt_face(x):
    global h,w
    plt.imshow(x.reshape((h, w)), cmap=plt.cm.gray)
    plt.xticks([])

plt.figure(figsize=(10,5))
nplt = 8
for i in range(nplt):
    plt.subplot(2,4,i+1)
    plt_face(U[:,i])
    if i == nplt-1:
      plt.savefig("eigenface.jpg")
plt.show()

# get eigenface coefficient
eigenface_coefficients = np.dot(U.transpose(), faces_train_centered.T)
print(f'The size of eigenface coefficients is {eigenface_coefficients.shape}')

# Compute the Eigenface coefficients of the training images
projected_train = np.dot(U.T, faces_train_centered.T)
print(projected_train.shape)

# Compute the Eigenface coefficients of the test images
faces_test_centered = faces_test_p - mean_face
projected_test = np.dot(U.T, faces_test_centered.T)
print(projected_test.shape)

print(f'The size of train eigenface coefficients is {projected_train.shape}')
temp_train = projected_train.T
for idx,face in enumerate(faces_train):
    plt.imshow(face)
    plt.savefig("train"+str(idx)+".png")
    print(temp_train[idx])
    print('++++++++++++++++++++++')
print("##############################")
print("##############################")
print(f'The size of test eigenface coefficients is {projected_test.shape}')
temp_test = projected_test.T
for idx,face in enumerate(faces_test):
    plt.imshow(face)
    plt.savefig("test"+str(idx)+".png")
```

```
    print(temp_test[idx])
    print('++++++++++++++++++++++')

def dist(x,y):
    dis = 0
    for xi, yi in zip(x,y):
        dis+= pow(xi-yi,2)
    return pow(dis, 0.5)

def predict(test_eigen, train_eigen):
    d = float("inf")
    num = -1
    for i in range(train_eigen.shape[1]):
        dis = dist(test_eigen, train_eigen[:,i])
        if dis < d:
            d = dis
            num = i
    return num

result = []
y = [4,5,1,6,4]
wrong = 0
correct = 0
for i in range(projected_test.shape[1]):
    pre = predict(projected_test[:,i], projected_train)
    if  pre != y[i]:
        wrong +=1
    else:
        correct +=1
    result.append(pre)
print(result)
print(f"The accuracy is {correct/(correct+wrong)}")
```