Name: Tianxiu Ma

Student number: 230277302

Assignment number: Assignment 2

module code: ECS797U/ECS797P

Task 1: Construct the mean image and the covariance matrix of the training image set. Explain the method implemented and display the mean image. [4 marks]

Mean Face

mean_face = np.mean(X_train, axis=0): This line computes the mean image of the training set. The X_train matrix contains the training images, where each row represents a flattened image (images are originally 64x64 pixels, hence reshaping to (64, 64) later). The np.mean function with axis=0 calculates the mean of each column, which corresponds to averaging the pixel values across all training images. The result, mean_face, is a 1D array representing the average image in the flattened form.

cov_matrix = np.cov(X_train.T): This line calculates the covariance matrix of the dataset. The np.cov function requires that pixels are represented as rows, and observations (images) as columns, which is why X_train is transposed (X_train.T). The covariance matrix describes how each pair of pixels varies together across the set of images. It's a key component in understanding the data structure, especially for dimensionality reduction techniques like PCA, as it helps identify directions (principal components) with maximum variance.

mean_face_reshaped = mean_face.reshape((64, 64)): Since the mean image is calculated as a flattened array, to display it, you need to reshape it back to its original

2D shape (64x64 pixels in this case).

The mean face often looks like a blurred face because it averages the features (like eyes, nose, mouth) across all faces in the training set.

Task 2: Compute the eigenfaces of the training set, using pca_lowrank. Explain eigenfaces and display the 20 first eigenfaces [4 marks]

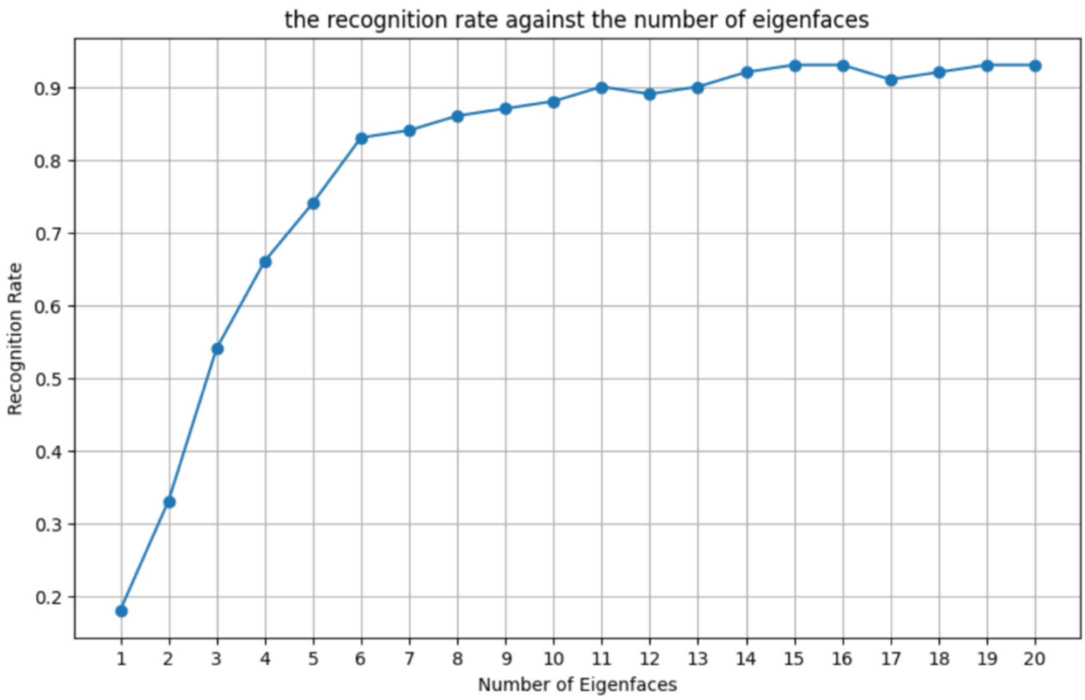
Eigenfaces are essentially principal components of a set of faces, which are used to efficiently represent faces in a lower-dimensional space. In other words, Eigenfaces provide a means to capture the essential variations in facial features. Each eigenface emphasizes certain features over others, such as edges, contours, or specific parts like the eyes, nose, or mouth. By projecting face images onto the space spanned by the eigenfaces, we significantly reduce dimensionality, enabling efficient face recognition and reconstruction tasks. This is possible because, typically, a small number of eigenfaces can capture most of the variability among face images.

Task 3: Project both training images and testing images onto the first 20 eigenfaces. Compute the distance from the projected test images to the projected training images.

Display the top 6 best matched training images for the first 6 test images. Compute the recognition rate using 20 eigenfaces (predicted class is the class of the nearest training set image). Investigate the effect of using different number of eigenfaces for recognition (e.g. plot the recognition rate against the number of eigenfaces). [7 marks]

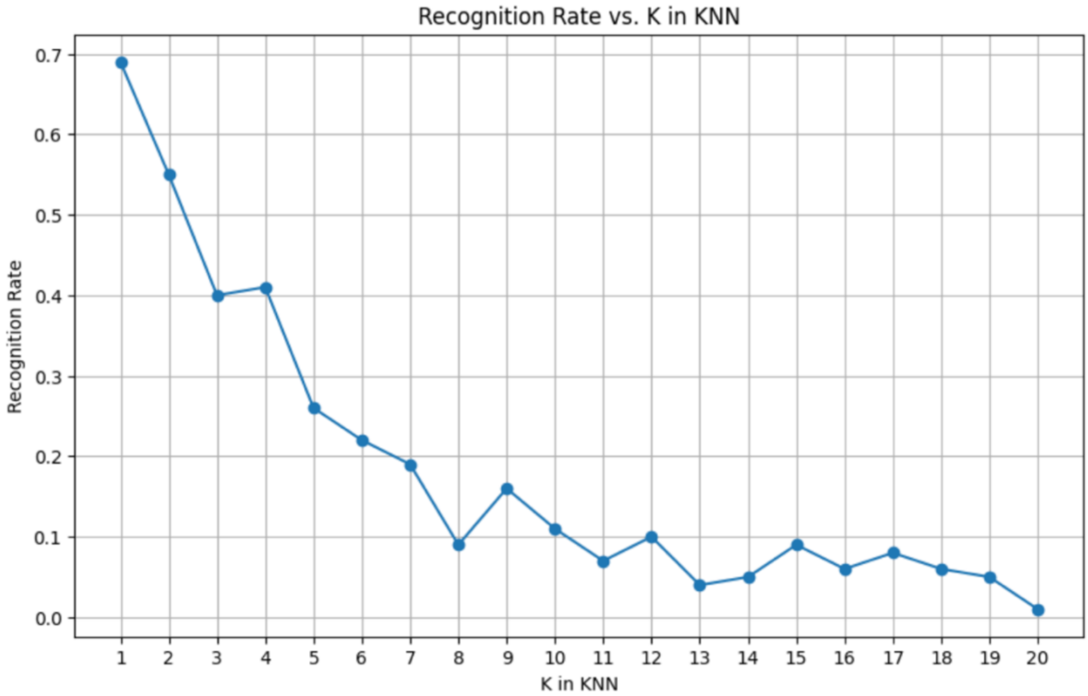
a. Display the top 6 best matched training images for the first 6 test images

b. The recognition rate against the number of eigenfaces



There is an increase in the recognition rate as more eigenfaces are used. After that optimal point, the recognition rate might plateau or even decrease due to overfitting or noise in the data being mistaken as signal.

Task 4: So far we have implemented a KNN classifier with K=1. Investigate the effect of K in K-Nearest-Neighbour (KNN) classifier. Plot the average recognition rate against K). [5 marks]



For tuning K in a KNN model, the goal is to find the value of K that achieves the highest recognition rate without overfitting. A high recognition rate for small values of K indicates that the nearest neighbors provide a good prediction for the class of a test sample. A decreasing recognition rate for larger values of K suggests that including more neighbors leads to noise and less accurate predictions, as the classifier starts considering points that are further away and less similar to the test sample.