By Ojas Mithbavkar

Dataset:

For this task, the Yale Face Database has been used. It contains 11 images of 15 people each, in various modes e.g. with glasses, with sleepy face, without glasses, happy face, etc. The images in the database as given do not have the necessary image extensions (e.g. .png, .jpg). So, we have to preprocess them and append .png /.jpg to every file name. Each image is of size (243,320) in Black and White format. We split the dataset randomly into 70 – 30 % train test. So, we have 115 train images and 50 test images.

Algorithm:

The main essence of the algorithm is expressing the faces as a linear combination of basis Eigenfaces and using Principal Component Analysis to reduce the dimensions. The steps of the algorithm are as follows:

1. We have 115 training images each of shape (243,320). We flatten each image to size 77760 and concatenate them to form a training matrix A of shape (115,77760). We perform a similar procedure for creating the test matrix.
2. The next step is to subtract the mean from the matrix. We transpose the training matrix and then calculate the mean of every row to get a vector of size 77760. We subtract this vector from every column of the train matrix using broadcasting.
3. The covariance matrix $C = AA^T$ which will be of shape 77760 * 77760. To solve this matrix for the eigenvalues and eigenvectors will be computationally hefty. Hence, as proposed in [2], we compute the matrix $L = A^TA$ whose eigenvalues are the same as that of C. We then find the eigenvalues and eigenvectors E of this matrix.
4. As described in [2], the eigenvectors of C are given by U = AE. We then sort the 115 eigenvalues and pick k eigenvectors corresponding to the k largest eigenvalues. These are the eigenfaces that we shall use for the face recognition task.
5. We compute the weights corresponding to the training set by V^TA where V is the matrix containing k selected eigenvectors.

6. Then, we take the test matrix T and subtract the mean again. We project the test faces onto the eigenfaces by $V^T T$ to get weight matrix of shape (k,50).
7. Then for each test face, we take its weights, compute its distance from every train face. We pick the minimum distance and assign the label if the distance is below some threshold. Else we do not assign a label and consider the face unknown.
8. The accuracy obtained can vary, as we randomly select the training set. So, it may be possible that we include more images of a particular subject in the training set, which will skew the eigenfaces. In such condition, the model has seen less of other faces in the training set and more of them in the test set, thereby affecting the accuracy.

References:

1. Turk, Matthew A., and Alex P. Pentland. "Face recognition using eigenfaces." Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 1991.
2. Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), 71-86.
3. https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eigh.html
4. https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html
5. https://github.com/pushpendradahiya/FaceDetection/blob/master/eigenfaces.py
6. https://github.com/DavidTorpey/eigenfaces/blob/master/eigenfaces.pdf
7. https://github.com/khushnaseeb/Face-Detection-and-Recognition-Using-Eigen-Face-Method-/blob/master/FaceRecognition.py
8. Yale Face Database B.