INTRODUCTION

Face is a complex multidimensional structure and needs good computing techniques for recognition. The face is our primary and first focus of attention in social life playing an important role in identity of individual. We can recognize numerous faces learned throughout our lifespan and identify that faces at a glance even after years. There may be variations in faces due to aging and distractions like beard, glasses or change of hairstyles.

Face recognition is an integral part of biometrics. In biometrics, basic traits of human are matched to the existing data and depending on the results, identification of a human being is done. Systems that can detect and recognize faces could be applied to a wide variety of practical applications including criminal identification, security systems, identity verification etc. Efficient facial recognition is one of the important problem being solved now days using different techniques.

OBJECTIVE

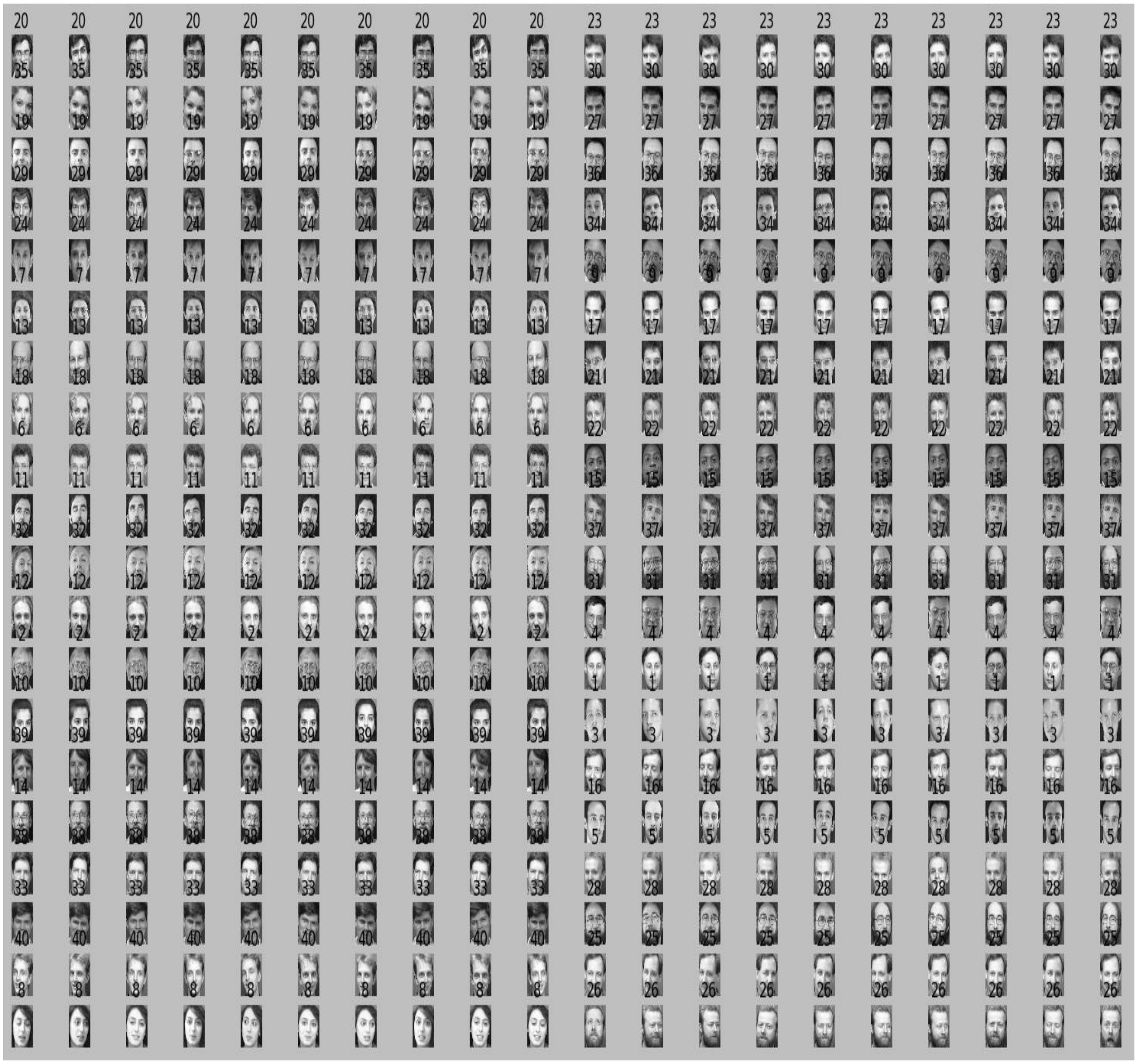
Although with the advent of ML/AI in recent years, deep neural networks and other similar approaches have already showcased accuracy up to 99% in this field but these methods primarily require huge amount of data to train to provide such accuracy. Another limitation is, they are computationally expensive solution for Face Recognition and can't be deployed on low end devices. Principal Component Analysis (PCA) based Face Recognition can be efficiently used in such cases.

The purpose of this project is to analyze how a PCA based facial recognition system can be built from scratch and identify what parameters of the system are optimal. The goal is to analyze the trade-off between number of principal components used for data representation versus the precision achieved using them. The dimensionality reduction principle of PCA accounts for the smaller face space while still preserving the integrity of the data.

DATASET DESCRIPTION

AT&T "The Database of Faces" (formerly "The ORL Database of Faces")

It contains 10 different facial images of 40 different people. The images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement)



METHODOLOGY

Finding Eigenfaces:

1. Load images and convert each of them into a NumPy matrix *(X_i)*. Flatten each image *(X_i)* into a vector and make a matrix *(A)* of all images by placing each X_i as a column of matrix *A*.
2. Compute the mean of all the images.
3. Compute the normalized images *(Φ_i)* by subtracting the mean from all the images.
4. Compute A^TA, which is different from the covariance matrix *(AA^T)*, in order to avoid huge matrix for eigen decomposition problem.
5. Compute the eigenvalue and eigenvector *(v_i)* of A^TA using SVD.
6. Compute the eigen vectors *(u_i)* of AA^T as follows: $u_i = A.v_i$
7. Select the best *K* eigen vectors, the selection of these eigen vectors is done heuristically.

Finding Weights:

1. Eigen vectors *(u_i)* found in previous section, when converted to a matrix have a face like appearance. Since these are eigen vectors and have a face like appearance, they are called Eigenfaces.
2. Each face in the training set (minus the mean), Φ_i can be represented as a linear combination of these eigen vectors.

$$\Phi_{\boldsymbol{i}} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{k}} \boldsymbol{w_j u_j} \text{ , where } u_j \text{ are Eigen faces}$$

3. These weights can be calculated as follows: $\boldsymbol{w_j} = \boldsymbol{u_j^T} \Phi_{\boldsymbol{i}}$
4. Each normalized training image is represented in this basis as a vector.

$$\boldsymbol{\Omega_i} = \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \\ . \\ . \\ . \\ \boldsymbol{w_k} \end{bmatrix}$$

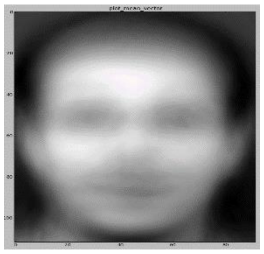
5. Calculate such a vector corresponding to every image in the training set and store them.

Recognition Task:

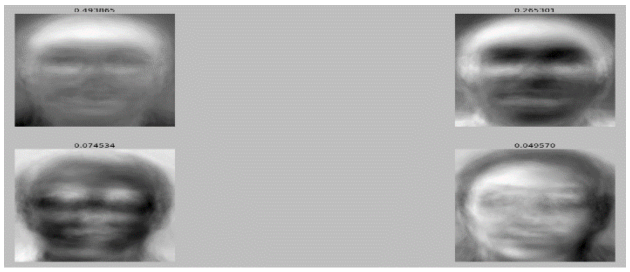
1. For an input test image $\boldsymbol{\Gamma}$, normalize it by subtracting the mean image from it.
2. Find the weight vector corresponding to it using $\boldsymbol{w_i} = \boldsymbol{u_i^T \Gamma}$, where i = 1 to K
3. Use the k-nearest neighbor (KNN) algorithm to find the k nearest neighbors of weight vector found in previous step.
4. Predict the image to be of same label as the maximum occurring neighbor in previous step.
5. In case of a tie, use the label with the least average distance.

CALCULATION

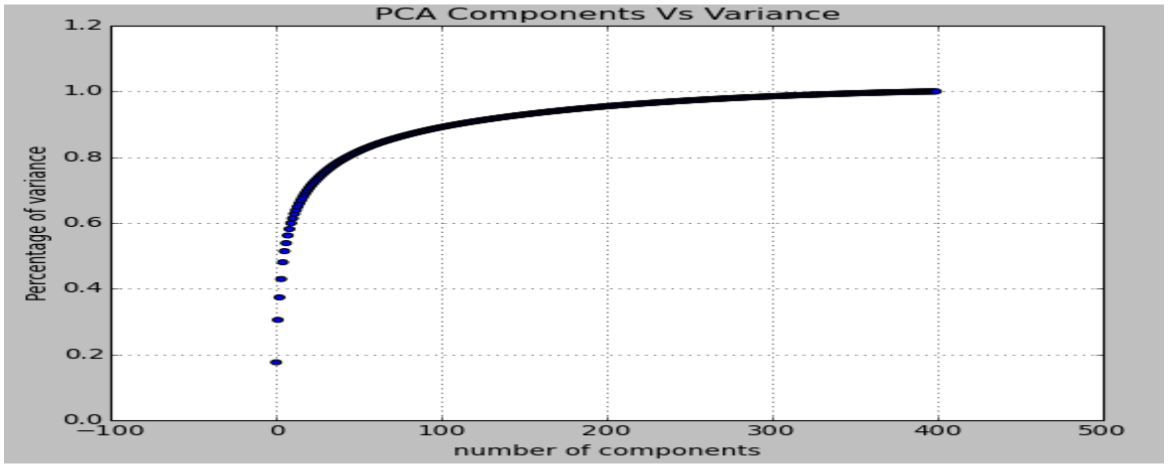
1. **Mean Image**



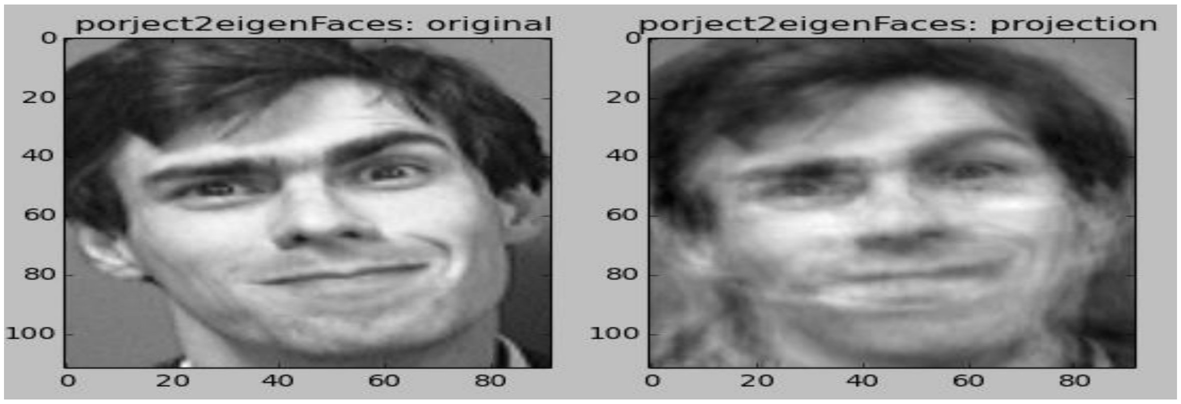
2. **Sample Eigen faces**



3. **Number of Principal Components vs Variance**



4. **Original image Vs Projected with K = 43 to preserve 80% of data (Total Components = 400)**



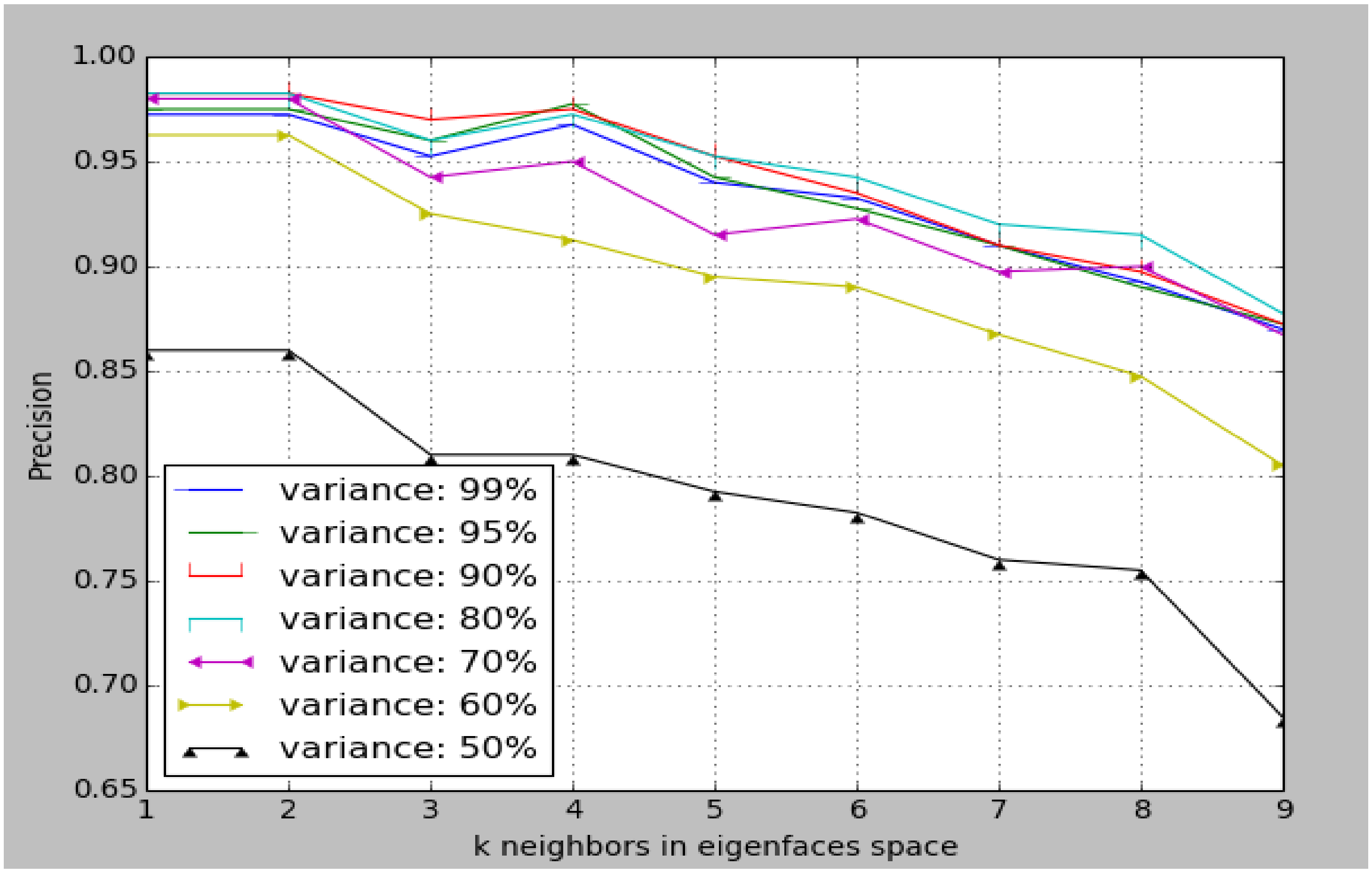
ANALYSIS AND CONCLUSION

Top K eigenfaces are used to represent an image, where K is determined by how much variance this sub-eigenspace can represent. Investigation is done on the following variance percentage and the derived values of K from each of them.

Variance %	99	95	90	80	70	60	50
Principal Components (K)	324	189	180	43	19	10	5

Each image in the dataset is considered as a query image and the other images as training data. If the predict label is the same as the original label, it's a true positive else a false positive. Precision percentage is considered as the performance metric for the recognition system.
Investigation is also done using different values of k in KNN algorithm (k ranging from 1 to 10).

Following is the data for precision achieved with different value of k (in KNN) using different variance percentages.



It's observed from the above data that the top K components eigen vectors (eigenfaces) represent as much variance as possible.

APPENDIX

Data set:

https://www.kaggle.com/kasikrit/att-database-of-faces or

http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/att_faces.zip