Facial Recognition and Image Compression using Singular Value Decomposition and Linear Discriminant Analysis

Philippe Joly

Vanier Cégep/College, Montréal, Quebec, H4L 3X9, Canada
2151940@edu.vaniercollege.qc.ca

Submitted: 26th May 2023

Abstract

Whether it is on the latest smart phone or passing through airport security, we interact with facial recognition everyday. This feat of computer vision may seem magical; however, it relies heavily on fundamental concepts of mathematics. This paper explores various linear algebra concepts such as singular value decomposition and linear discriminant analysis and their application in fast and efficient facial recognition. Starting with a collection of 64 168x192 images of 38 different subjects from the *Extended Yale Face Database B*, a software capable of differentiating each individual was designed by applying mathematical approximation, reduction, clustering, and classification techniques. We were able to compress and decompress images from 32 256 down to 800 values retaining the predominant facial features, using a truncated version of the left singular matrix. Furthermore, using linear discriminant analysis, looking at the 300 leading left singular vectors, we were able to associate new images back to each subject in the database.

1 Introduction

The aim of this research project is to investigate the properties and capabilities of principal component analysis combined with discriminant analysis in a facial recognition context. We first explore how singular value decomposition can be used as a tool to compress images, fishing out the most prominent features across a data set.

This has been done before and the process is described in Steven L. Brunton and J. Nathan Kutz's *Data Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* [2]. Similarly to this paper, Brunton and Kutz used the *Yale Extended Face Database B* [3] to examine the capabilities of singular value decomposition in image compression [2]. They concluded that this technique is reasonably effective, compressing testing images from 32 256 dimensions down to 400 values still getting an somewhat accurate reconstruction of the original image [2].

This paper attempts to extend the findings of Brunton and Kutz, investigating if the compressed data can be used for facial recognition. In essence, we explore the feasibility of creating a working facial recognition software relying on the principle components of a large face data set. In other words, we attempt to create a software capable of classifying and recognizing people in a database using linear discriminant analysis. The discriminant analyses will be performed on the compressed images projected onto the principal components of the larger face database.

To investigate the properties of singular value decomposition and linear discriminant analyses in a facial recognition setting, we must first understand the underlying mathematical concepts behind them.

The aim of singular value decomposition (SVD) is to factor any rectangular matrix in two matrices with orthonormal column vectors and a diagonal matrix.

$$A = U\Sigma V^T$$

Where the columns of U and V called the left and right singular vectors respectively form orthonormal sets.

To show that any rectangular matrix can undergo SVD we must pass by spectral decomposition. Spectral decomposition aims to diagonalize any symmetric matrix into $M = Q\Lambda Q^T$ where the columns of Q are orthogonal eigenvectors of M and Λ is a diagonal matrix with the corresponding eigenvalues.

$$M = Q\Lambda Q^T = \begin{bmatrix} | & & | \\ \vec{u}_1 & \dots & \vec{u}_n \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} — & \vec{u}_1^T & — \\ & \vdots & \\ — & \vec{u}_n^T & — \end{bmatrix} \text{ where } \langle \vec{u}_i,\ \vec{u}_j \rangle = 0 \text{ for } i \neq j$$

1.1 Spectral Decomposition

The spectral theorem for symmetric matrices guarantees that any symmetric matrix can undergo spectral decomposition. The spectral theorem states that, for any real symmetric matrix, all of its eigenvalues are real, its eigenvectors corresponding to distinct eigenvalues are orthogonal and it is diagonalizable. First we can prove these propositions one by one.

Spectral Theorem Proof [11]

Let M be a real symmetric $m \times n$ matrix, then

1. All eigenvalues of M are real

$$M\vec{v} = \lambda\vec{v} \rightarrow \overline{M\vec{v}} = \overline{\lambda\vec{v}} \rightarrow M\bar{\vec{v}} = \bar{\lambda}\bar{\vec{v}}$$

$$(M\bar{\vec{v}})^T = (\bar{\lambda}\bar{\vec{v}})^T = \bar{\vec{v}}^T M = \bar{\lambda}\bar{\vec{v}}^T$$

$$\rightarrow \bar{\vec{v}}^T M \vec{v} = \begin{cases} \bar{\vec{v}}^T (M\vec{v}) = \bar{\vec{v}}^T (\lambda \vec{v}) = \lambda \bar{\vec{v}}^T \vec{v} \\ (\bar{\vec{v}}^T M)\vec{v} = (\bar{\vec{v}}^T \bar{\lambda})\vec{v} = \bar{\lambda}\bar{\vec{v}}^T \vec{v} \end{cases}$$

$$\rightarrow (\lambda - \bar{\lambda})\bar{\vec{v}}^T \vec{v} = 0$$

$$\text{For nonzero vectors } \bar{\vec{v}}^T \vec{v} > 0 \rightarrow \lambda = \bar{\lambda} \rightarrow \lambda \in \mathbb{R}$$

2. Eigenvectors of distinct eigenvalues are orthogonal

Let λ_1 and λ_2 be eigenvalues of M with corresponding eigenvectors $\vec{v}_1$ and $\vec{v}_2$. Then following the same procedure outlined in the previous step replacing $\bar{\vec{v}}^T M \vec{v}$ by $\vec{v}_1^T M \vec{v}_2$, we get

$$\rightarrow \vec{v}_1^T M \vec{v}_2 = \begin{cases} \vec{v}_1^T (M\vec{v}_2) = \vec{v}_1^T (\lambda_2 \vec{v}_2) = \lambda_2 \vec{v}_1^T \vec{v}_2 \\ (\vec{v}_1^T M)\vec{v}_2 = (\vec{v}_1^T \lambda_1)\vec{v}_2 = \lambda_1 \vec{v}_1^T \vec{v}_2 \end{cases}$$

$$\rightarrow (\lambda_1 - \lambda_2)\vec{v}_1^T \vec{v}_2 = 0$$

$$\text{We see that } \lambda_1 \neq \lambda_2 \rightarrow \vec{v}_1^T \vec{v}_2 = \langle \vec{v}_1,\ \vec{v}_2 \rangle = 0 \rightarrow \vec{v}_1 \perp \vec{v}_2$$

3. M is diagonalizable [9]

For the base case of a 1×1 matrix, M is already in diagonal form. Therefore the statement stands for $n = 1$. Assume that all $k \times k$) real symmetric matrices are orthogonally diagonizable. Let A be a $(K + 1) \times (k + 1)$ real symmetric matrix. Let λ_1 be one of the eigenvalues of A with corresponding unit eigenvector $\vec{v}_1$. Using the Gram-Schmidt process, we can complement $\vec{v}_1$ to form an orthonormal basis $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n\}$ of $\mathbb{R}^n$. Let Q_1 be the orthogonal matrix with column vectors $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n\}$.

$$Q_1 = \begin{bmatrix} | & | & & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \\ | & | & & | \end{bmatrix}$$

Making use of the orthogonality of Q,

$$Q_1^T A Q_1 = \begin{bmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_n^T & — \end{bmatrix} A \begin{bmatrix} | & | & & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_n^T & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ A\vec{v}_1 & A\vec{v}_2 & \dots & A\vec{v}_n \\ | & | & & | \end{bmatrix}$$

$$= \begin{bmatrix} — & \vec{v}_1^T & — \\ — & \vec{v}_2^T & — \\ & \vdots & \\ — & \vec{v}_n^T & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \lambda_1\vec{v}_1 & A\vec{v}_2 & \dots & A\vec{v}_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \lambda_1 & * \\ 0 & A_1 \end{bmatrix} = B$$

Since $\vec{v}_1^T(\lambda_1\vec{v}_1) = \lambda_1\langle\vec{v}_1,\ \vec{v}_1\rangle = \lambda_1$ and $\vec{v}_i^T(\lambda_1\vec{v}_1)\langle\vec{v}_i,\ \vec{v}_1\rangle = 0$ for $i \neq 1$, using the fact that $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n\}$ is an orthonormal set.

Notice that B is symmetric,

$$B^T = (Q_1^T A Q_1)^T = Q_1^T A^T Q_1 = B$$

Therefore, B has to have the form:

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & A_1 \end{bmatrix}$$

Where A_1 has to be symmetric. Moreover, B is similar to A. It can be shown that similar matrices like A and B share the same characteristic polynomial [Appendix A]. Note that the characteristic polynomial of A_1 divides the characteristic polynomial of A and B. Therefore, A_1 is a $k \times k$ symmetric real matrix with real eigenvalues. Hence, the induction applies to it. That is to say that there is an orthogonal matrix P_2 such that $P_2^T A_1 P_2$ is a diagonal matrix D_1. Let Q_2 be an orthogonal $(k+1) \times (k+1)$ matrix.

$$Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix}$$

It follows that $Q = Q_1Q_2$ is also an orthogonal $(k+1) \times (k+1)$ matrix. Then,

$$Q^T A Q = (Q_1Q_2)^T A (Q_1Q_2) = Q_2^T(Q_1^T A Q_1)Q_2 = Q_2^T B Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & P_2^T \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & A_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & P_2^T A_1 P_2 \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_1 & 0 \\ 0 & D_1 \end{bmatrix}$$

Which is a diagonal matrix. Thissatifies the induction step, leading to the conclusion that for all $n \geq 1$, an $n \times n$ real symmetric matrix is orthogonally diagonizable.

1.2 Singular Value Decomposition [11]

This concept of spectral decomposition extends to non-symmetric and non-square matrices in singular value decomposition.

Similarly to the spectral decomposition, for $m \times n$ matrix A, SVD's goal is to find sets of orthonormal vectors $\{\vec{v}_1 \dots \vec{v}_r\} \subset \mathbb{R}^n$ and $\{\vec{u}_1 \dots \vec{u}_r\} \subset \mathbb{R}^m$ as well as a corresponding set of scalars $\{\sigma_1 \dots \sigma_r\} \subset \mathbb{R}$ such that $A\vec{v}_i = \sigma_i\vec{u}_i$ where $r = rank(A)$. Note that beyond the rank of A, $\vec{v}_{r+k} \in Nul(A) \rightarrow A\vec{v}_{r+k} = \vec{0}\ \forall k > 0$ and, therefore, can be discarded.

$$AV = U\Sigma$$

$$A \begin{bmatrix} | & & | \\ \vec{v}_1 & \dots & \vec{v}_r \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \vec{u}_1 & \dots & \vec{u}_r \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

$$\text{since the columns of } V \text{ are orthonormal, } AV = U\Sigma \to A = U\Sigma V^T$$

$$A = \begin{bmatrix} | & & | \\ \vec{u}_1 & \dots & \vec{u}_r \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \begin{bmatrix} — & \vec{v}_1^T & — \\ & \vdots & \\ — & \vec{v}_r^T & — \end{bmatrix}$$

We can show that, if the columns of V are orthogonal, then the columns of U are orthogonal from the initial stipulation of SVD.

$$A\vec{v}_i = \sigma_i \vec{u}_i \to \vec{u}_i = \frac{A\vec{v}_i}{\sigma_i}$$

$$\to \langle \vec{u}_i,\ \vec{u}_j \rangle = \vec{u}_i^T \vec{u}_j = (\frac{A\vec{v}_i}{\sigma_i})^T \frac{A\vec{v}_j}{\sigma_j} = \frac{\vec{v}_i^T A^T A \vec{v}_j}{\sigma_i \sigma_j} = \frac{\vec{v}_i^T \lambda_j \vec{v}_j}{\sigma_i \sigma_j} = \frac{\sigma_j}{\sigma_i} \langle \vec{v}_i,\ \vec{v}_j \rangle = 0 \text{ for } i \neq j$$

We can explore the decomposition of any rectangular matrix A by looking at A^TA and AA^T. Let A be a $m \times n$ matrix, then A^TA is $n \times n$ symmetric matrix with positive eigenvalues as shown below.

$$(A^TA)^T = A^T(A^T)^T = AA^T \to A^TA \text{ is symmetric}$$

$$A^TA\vec{u} = \lambda \vec{u} \to \vec{u}^T A^T A \vec{u} = \lambda \vec{u}^T \vec{u}$$

$$\to \lambda = \frac{\vec{u}^T A^T A \vec{u}}{\vec{u}^T \vec{u}} = \frac{(A\vec{u})^T A\vec{u}}{\langle \vec{u},\ \vec{u} \rangle} = \frac{\langle A\vec{u},\ A\vec{u} \rangle}{\langle \vec{u},\ \vec{u} \rangle} \geq 0$$

Note that the same properties follow for AA^T and that the non-zero eigenvalues of AA^T and A^TA will be identical. Since A^TA and AA^T are symmetric, they can undergo spectral decomposition.

$$A^TA = V\Lambda V^T$$

$$AA^T = U\Lambda U^T$$

Therefore, by looking at A^TA, we can show that the existence of a spectral decomposition of any symmetric matrix implies the existence of a singular value decomposition for any rectangular matrix.

$$A^TA = V\Lambda V^T = V \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_r \end{bmatrix} V^T = V \begin{bmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_r} \end{bmatrix}^2 V^T = V \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}^2 V^T$$

$$= V\Sigma I \Sigma V^T = V\Sigma U^T U \Sigma V^T = (U\Sigma V^T)^T (U\Sigma V^T) \to A = U\Sigma V^T$$

With the columns of V (from the properties of spectral decomposition) and of U respectively being orthonormal and $\sigma_i^2 = \lambda_i$ of A^TA (which are always positive for as proved above).

By convention, we write the σ's, $\vec{v}$'s, and $\vec{u}$'s of matrices Σ, V, and U in an order such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. Because of this, we notice that values of $\vec{u}_i \sigma_i \vec{v}_i^T$ of i's closer to one will be much greater and significant when it comes to the reconstruction of A than for i's closer to r. This motivates the Eckart-Young Minkinski theorem which states that the closest rank-k approximation to A is given by the rank-k SVD of $\tilde{A}$:

$$\underset{\tilde{A},\, s.t.\, rank(\tilde{A})=k}{\operatorname{argmin}} \|A - \tilde{A}\| = \tilde{U}\tilde{\Sigma}\tilde{V}^T$$

Where $\tilde{\Sigma}$ is the diagonal matrix with σ_1 to σ_k, $\tilde{U}$ and $\tilde{V}$ are the first k leading columns of U and V [7].

$$A \approx \tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^T = \begin{bmatrix} | & & | \\ \vec{u}_1 & \dots & \vec{u}_k \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} — & \vec{v}_1^T & — \\ & \vdots & \\ — & \vec{v}_k^T & — \end{bmatrix}$$

Note that $\|M\|$ refers to the general Euclidean norm of a matrix defined as $\|M\| = \max_{\vec{x}\neq\vec{0}} \frac{\|M\vec{x}\|}{\|\vec{x}\|}$. This means that the truncated SVD will always be the best approximation of an original matrix for its rank.

To prove that the best rank-k matrix approximation can be found using the first k left and right singular vectors as well as their corresponding singular values, it must be shown that a matrix's Euclidean norm is always equal to its dominant singular value [7]. To do this, it can be shown that Euclidean matrix norm is simultaneously greater and smaller or equal to the largest singular value.

From the definition of the generalized Euclidean matrix norm,

$$\|A\| = \max_{\vec{x}\neq\vec{0}} \frac{\|A\vec{x}\|}{\|\vec{x}\|}$$

It follows that,

$$\|A\| = \max_{\vec{x}\neq\vec{0}} \frac{\|A\vec{x}\|}{\|\vec{x}\|} = \max_{\vec{x}\neq\vec{0}} \frac{1}{\|\vec{x}\|}\|A\vec{x}\| = \max_{\vec{x}\neq\vec{0}} \|\frac{1}{\|\vec{x}\|}A\vec{x}\| = \max_{\vec{x}\neq\vec{0}} \|A(\frac{\vec{x}}{\|\vec{x}\|})\| = \max_{\|\vec{q}\|=1} \|A\vec{q}\|$$

It can then be shown that the generalized Euclidean matrix norm is smaller or equal to the first singular value.

Let $\vec{u}_1, \ldots, \vec{u}_n$ be an orthonormal basis of $\mathbb{R}^n$ consisting of singular vectors of A: $\vec{u}_1, \ldots, \vec{u}_k$ and $\vec{u}_{r+k}, \ldots, \vec{u}_n \in \ker A$, where A is a $m \times n$ matrix of rank k. Also, let $\vec{v}_1, \ldots, \vec{v}_k$ be an orthonormal basis of the image of A.

$$A\vec{u}_i = \begin{cases} \sigma_i \vec{v}_i, \text{ for } i \in [1, k] \\ 0, \text{ for } i \in (k, n] \end{cases}$$

Let $\vec{q}$ be any unit vector in $\mathbb{R}^n$

$$\vec{q} = c_1\vec{u}_1 + \ldots + c_n\vec{u}_n, \text{ where } \|\vec{q}\| = \sqrt{c_1^2 + \ldots + c_n^2} = 1$$

$$\rightarrow A\vec{q} = c_1\sigma_1\vec{v}_1 + \ldots + c_k\sigma_k\vec{v}_k \rightarrow \|A\vec{q}\| = \sqrt{c_1^2\sigma_1^2 + \ldots + c_k^2\sigma_k^2} \leq \sqrt{c_1^2\sigma_1^2 + \ldots + c_k^2\sigma_1^2}$$
$$= \sigma_1\sqrt{c_1^2 + \ldots + c_k^2} \leq \sigma_1\sqrt{c_1^2 + \ldots + c_n^2} = \sigma_1$$
$$\rightarrow \|A\| = \|A\vec{q}\| \leq \sigma_1$$

To show that the matrix norm has to be larger or equal to the dominant singular value, we must simply find a vector $\vec{q}$ such that $\|A\vec{q}\| = \sigma_1$, as the matrix norm is defined as the maximum value of $\|A\vec{q}\|$. If we let $\vec{q} = \vec{u}_1$,

$$\rightarrow \|A\| = \|A\vec{q}\| = \|A\vec{u}_1\| = \|\sigma_1\vec{v}_1\| = \sigma_1$$
$$\rightarrow \|A\| = \|A\vec{q}\| \geq \sigma_1$$
$$\left.\begin{matrix} \|A\| \geq \sigma_1 \\ \|A\| \leq \sigma_1 \end{matrix}\right\} \rightarrow \|A\| = \sigma_1$$

Using this fact we can show that the SVD gives the best low rank approximation to nay matrix.

Let $A_k = U\tilde{\Sigma}_k V^T$ where U and V are the right and left singular matrix of A and $\tilde{\Sigma}_k$ is diagonal matrix with the first k values: $\sigma_1, \ldots, \sigma_k$ and the rest equal to 0.

$$\tilde{\Sigma}_k = \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & \end{bmatrix}$$

$$A - A_k = U\Sigma_k V^T - U\tilde{\Sigma}_k V^T = U(\Sigma - \tilde{\Sigma})_k V^T = U \begin{bmatrix} \ddots & & & & \\ & 0 & & & \\ & & \sigma_{k+1} & & \\ & & & \ddots & \\ & & & & \sigma_r \end{bmatrix} V^T$$

Now we use the fact the Euclidean norm of a matrix is its largest singular value.

$$\|A - A_K\| = \sigma_{k+1}$$

We can show that this is the smallest difference. Let B be an $m \times n$ matrix of rank k. By the rank-nullity theorem, dim ker $B = n - k$. If we let $W_{k+1} \subset \mathbb{R}^n$ be that $(k+1)$-dimensional subspace spanned by the $(k+1)$ first singular vectors of A, we can look at the nontrivial intersection subspace between W_{k+1} and ker B, as their dimensions sum up to $k + 1 + n - k = n + 1 > n$.

$$\therefore\ \exists \vec{q}_* = c_1\vec{u}_1 + \ldots + c_{k+1}\vec{u}_k + 1 \in W_{k+1} \cap \text{ker } B$$

$$\text{such that } \|\vec{q}_*\| = 1 \text{ and } B\vec{q}_* = \vec{0}$$

$$\rightarrow (A - B)\vec{q}_* = A\vec{q}_* = c_1 A\vec{u}_1 + \ldots + c_{k+1}A\vec{u}_{k+1} = c_1\sigma_1\vec{v}_1 + \ldots + c_{k+1}\sigma_{k+1}\vec{v}_{k+1}$$

$$\rightarrow \|(A - B)\vec{q}_*\| = \sqrt{c_1^2\sigma_1^2 + \ldots + c_{k+1}^2\sigma_{k+1}^2} \geq \sqrt{c_1^2\sigma_{k+1}^2 + \ldots + c_{k+1}^2\sigma_{k+1}^2}$$

$$= \sigma_{k+1}\sqrt{c_1 + \ldots + c_{k+1}} = \sigma_{k+1}\|\vec{q}_*\| = \sigma_{k+1}$$

The smallest difference between a matrix A and a rank-k approximation B is σ_{k+1}. This difference is achieved with $B = A_k = U\tilde{\Sigma}_k V^T$. Therefore, the truncated SVD is the best low-rank approximation minimizing the general Euclidean matrix norm [7].

This motivates the compression of large data matrices into their smaller corresponding truncated SVDs to perform any operations on them, which would have been too computationally expensive.

In our application of the SVD, each row of A represents the pixel values of different faces. Therefore, A^TA can be interpreted as the covariance matrix between different faces with the entry $(A^TA)_{ij}$ being the inner product of the face vector of person i and j [2].

$$A^TA = \begin{bmatrix} — & \vec{a}_1^T & — \\ & \vdots & \\ — & \vec{a}_n^T & — \end{bmatrix} \begin{bmatrix} | & & | \\ \vec{a}_1 & \ldots & \vec{a}_n \\ | & & | \end{bmatrix} = \begin{bmatrix} \langle \vec{a}_1, \vec{a}_1 \rangle & \ldots & \langle \vec{a}_1, \vec{a}_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \vec{a}_n, \vec{a}_1 \rangle & \ldots & \langle \vec{a}_n, \vec{a}_n \rangle \end{bmatrix}$$

We can interpret the $(A^TA)_{ij}$ entries being a correlation score between two faces. A large entry would imply a large inner product pointing to similar features and facial structure. On the other hand a small inner product, implying that the face vectors are close to orthogonal, would suggest that the two faces are very different.

Notice that the columns of U derived from the SVD are eigenvectors of the rows of that correlation matrix.

$$(A^TA)^T = AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T = U\Lambda U^T$$

For faces, we can interpret the fact that the columns U are eigenvectors of the covariance matrix, the $\vec{u}$'s being representative of specific facial features with $\vec{u}_i$ being more significant than $\vec{u}_{i+1}$.

This kind of process to extract various important trends in large data sets is widely used in statistics and is referred to as principle component analysis (PCA) and is computed as follows for any $m \times n$ matrix M with m being the number of samples and n the number of variables [2].

$$\text{First subtract its mean } B = M - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} M_{ij} \begin{bmatrix} 1 & \ldots \\ \vdots & \ddots \end{bmatrix}$$

Create the covariance matrix of the rows of B and compute its eigenvalues and vector.

$$C = B^T B \text{ with } C \begin{bmatrix} | & & | \\ \vec{v}_1 & \dots & \vec{v}_k \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \vec{v}_1 & \dots & \vec{v}_k \\ | & & | \end{bmatrix} \Lambda$$

Find the principal components T and express it using SVD.

$$T = BV = U\Sigma V^T V = U\Sigma$$

This principal component matrix indicates what direction in the data set accounts for more variance, making it useful to single out the most important part of the data set and to differentiate samples. Again, through a statistical lens, the columns of U can be interpreted as the orthonormal eigenvectors of the principal component matrix characterizing the variance in the data set.

This makes the columns of U a good basis choice to project new sample data points onto. In fact, following from the Eckart-Young theorem, the best rank-k approximation of a vector $\vec{x}$ is given by its projection on the truncated version of U from the SVD (*i.e.*, $\tilde{U}_k$).

$$\underset{\tilde{x},\, s.t.\, \tilde{x} \in \mathbb{R}^k}{\text{argmin}} \|\vec{x} - \tilde{x}\| = \tilde{U}_k \tilde{U}_k^T \vec{x}$$

This implies that any vector $\vec{x} \in \mathbb{R}^n$ can be compressed stored and processed as $\tilde{x} = \tilde{U}_k^T \vec{x} \in \mathbb{R}^k$ and can be decompressed as $\vec{x} \approx \tilde{U}_k \tilde{x}$. This can be a major advantage for storage, processing, and clustering. For instance in this project, through PCA, images of 192x168 (treated as vectors of 32 356 dimensions) were compressed down to 300 dimensions.

1.3 Linear Discriminant Analysis [6] [1]

Discriminant analysis is a technique to cluster and differentiate between classes. Simply put, discriminant analysis is investigating the probability of a vector $\vec{x}$ of belonging to class. Linear and quadratic discriminant analysis is all about finding with the highest probability of $\vec{x}$ belonging to a specific class treating the data points in each class as normal distributions.

From Bayes's theorem, in a Gaussian distribution, the probability of a vector belonging to a class (*i.e.*, the posterior probability) is given by

$$P(C_i|\vec{x}) = \frac{P(\vec{x}|C_i)P(C_i)}{P(\vec{x})}$$

Where $P(\vec{x}|C_i)$ is the likelyhood of observing $\vec{x}$ in class C_i, $P(C_i)$ is the prior probability of class C_i, and $P(\vec{x})$ is the evidence. the evidence is simply a scaling factor such that $\sum_{j=1}^{K} P(C_j|\vec{x}) = 1$ across K classes.

$$P(\vec{x}) = \sum_{j=1}^{K} P(\vec{x}|C_j)P(C_j)$$

Note that the likelyhood of observing $\vec{x}$ in class C_i assuming a normal gaussian distribution in $\mathbb{R}^n$ is,

$$P(\vec{x}|C_i) = \frac{e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T \Sigma_i^{-1} (\vec{x}-\vec{\mu}_i)}}{\sqrt{(2\pi)^n |\Sigma_i|}}$$

$$\text{This results in, } P(C_i|\vec{x}) = \frac{e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T \Sigma_i^{-1} (\vec{x}-\vec{\mu}_i)} P(C_i)}{\sqrt{(2\pi)^n |\Sigma_i|} \sum_{j=1}^{K} P(\vec{x}|C_j)P(C_j)}$$

The discriminant function δ can be defined by taking the logarithm of the posterior probability. Note that we can disregard the $P(\vec{x}) = \sum_{j=1}^{K} P(\vec{x}|C_j)P(C_j)$ as it will be the same for all classes.

$$\ln P(C_i|\vec{x}) = \ln P(\vec{x}|C_i)P(C_i) = \ln P(\vec{x}|C_i) + \ln P(C_i)$$

$$= \ln \frac{e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)}}{\sqrt{(2\pi)^n|\Sigma_i|}} + \ln P(C_i)$$

$$= -\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i) - \frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_i| + \ln P(C_i)$$

Note that the $\frac{n}{2}\ln(2\pi)$ can be dropped as it is the same across all classes. Furthermore, for our purposes, we can assume the prior probability of each classes to be the same and, therefore, can also be discarded giving us the the discriminant function.

$$\delta_i(\vec{x}) = -\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i) - \frac{1}{2}\ln|\Sigma_i|$$

Notice that the the $\delta_i(\vec{x})$ function is a combination of a quadratic form, a linear term, and a constant term. It can be further reduced.

$$\delta_i(\vec{x}) = -\frac{1}{2}[(\vec{x}^T - \vec{\mu}_i^T)\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i) - \ln|\Sigma_i|]$$

$$= -\frac{1}{2}(\vec{x}^T\Sigma_i^{-1}\vec{x} - \vec{x}^T\Sigma_i^{-1}\vec{\mu}_i - \vec{\mu}_i^T\Sigma_i^{-1}\vec{x} + \vec{\mu}_i^T\Sigma_i^{-1}\vec{\mu}_i - \ln|\Sigma_i|)$$

Recalling that the covariance matrix is always symmetric (as shown in subsection 1.2) and noticing that $\vec{x}^T\Sigma_i^{-1}\vec{\mu}_i$ is a linear scalar function of the components of $\vec{x}$,

$$\vec{x}^T\Sigma_i^{-1}\vec{\mu}_i = (\vec{x}^T\Sigma_i^{-1}\vec{\mu}_i)^T = \vec{\mu}_i^T(\Sigma_i^{-1})^T(\vec{x}^T)^T = \vec{\mu}_i^T\Sigma_i^{-1}\vec{x}$$

$$\rightarrow \delta_i(\vec{x}) = \vec{\mu}_i^T\Sigma_i^{-1}\vec{x} - \frac{1}{2}(\vec{x}^T\Sigma_i^{-1}\vec{x} + \vec{\mu}_i^T\Sigma_i^{-1}\vec{\mu}_i - \ln|\Sigma_i|)$$

$$= \vec{\mu}_i^T\Sigma_i^{-1}\vec{x} - \frac{1}{2}\vec{x}^T\Sigma_i^{-1}\vec{x} + \frac{1}{2}\vec{\mu}_i^T\Sigma_i^{-1}\vec{\mu}_i - \frac{1}{2}\ln|\Sigma_i|$$

$$\rightarrow \delta_i(\vec{x}) = \vec{x}^TW_i\vec{x} + \vec{w}_i^T\vec{x} + \omega_i$$

Where $W_i = -\frac{1}{2}\Sigma_i^{-1}$, $\vec{w}_i = \Sigma_i^{-1}\vec{\mu}_i$, and $\omega_i = \frac{1}{2}\vec{\mu}_i^T\Sigma_i^{-1}\vec{\mu}_i - \frac{1}{2}\ln|\Sigma_i|$ are the parameters of the discriminant function for class C_i. Quadratic discriminant analysis will predict $\vec{x}$ to belong to the class that maximizes the discriminant function δ.

We can look at the decision boundary between two classes by setting their discriminant functions equal to each other.

$$\delta_i(\vec{x}) = \delta_j(\vec{x})$$

$$\rightarrow \vec{x}^TW_i\vec{x} + \vec{w}_i^T\vec{x} + \omega_i = \vec{x}^TW_j\vec{x} + \vec{w}_j^T\vec{x} + \omega_j$$

$$\rightarrow \vec{x}^T(W_i - W_j)\vec{x} + (\vec{w}_i - \vec{w}_j)^T\vec{x} + \omega_i - \omega_j = 0$$

$$\rightarrow \ell_{ij} : \vec{x}^TP_{ij}\vec{x} + \vec{p}_{ij}^T\vec{x} + \rho_{ij} = 0$$

We can interpret ℓ as a level hyper-surface of the sum of the quadratic form with $P_{ij} = \frac{1}{2}(\Sigma_j^{-1} - \Sigma_i^{-1})$, the linear function with $\vec{p}_{ij} = \Sigma_i^{-1}\vec{\mu}_i - \Sigma_j^{-1}\vec{\mu}_j$, and constant term $\rho_{ij} = \frac{1}{2}(\vec{\mu}_i^T \Sigma_i^{-1}\vec{\mu}_i - \vec{\mu}_j^T \Sigma_j^{-1}\vec{\mu}_j + \ln\frac{|\Sigma_j|}{|\Sigma_i|})$. Any point on that surface is considered to have an equal chance of belonging to class C_i and C_j.

Notice that if we assume that the covariance matrix is the same between classes the boundary lines reduce from a quadratic form to a linear form.

$$\Sigma_i = \Sigma_j = \Sigma \to \begin{cases} P_{ij} = \begin{bmatrix} 0 & \cdots \\ \vdots & \ddots \end{bmatrix} \\ \vec{p}_{ij} = \Sigma^{-1}(\vec{\mu}_i - \vec{\mu}_j) \\ \rho_{ij} = \frac{1}{2}(\vec{\mu}_i^T \Sigma^{-1}\vec{\mu}_i - \vec{\mu}_j^T \Sigma^{-1}\vec{\mu}_j) \end{cases}$$

$$\to \ell_{ij} : \vec{p}_{ij}^{\,T}\vec{x} + \rho_{ij} = 0$$

This is called linear discriminant analysis (LDA) - a simplification of quadratic discriminant analysis. LDA is less precise in predicting to which class a value belongs; however, it is much more efficient as it only need to calculate a single covariance matrix instead of one for each class which makes it much more efficient on large data sets with a great number of classes.

Another way to visualize LDA is to see it as a projection. In fact, discriminant analysis can be thought of as the projection of data vectors onto the normal of the decision hyper-surface. The projection on the hyper-surface normal can be interpreted as finding a vector's distance and relation to the hyperplane, giving a relative discriminant value between classes. Projecting onto the orthogonal complement of the hyper-surface happens to be the projection that maximized the difference between the means and minimized the spread within classes (in the image). This can be illustrated in $\mathbb{R}^2$ with two distributions projected onto a random subspace and onto the normal of the decision line found using LDA [Figure 1]. Figure 1 below taken from *Data Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* by Steven L. Brunton and J. Nathan Kutz exhibits the nature of discriminant analysis, showing the projection of data points onto the normal of a decision lines (one chosen randomly and the other found using LDA) as histograms. Figure 1 illustrates how the optimal projection found using LDA minimizes the spread within classes and maximizes the spread between classes. The same principle outlined in Figure 1 carries over to higher dimensions and for more classes.

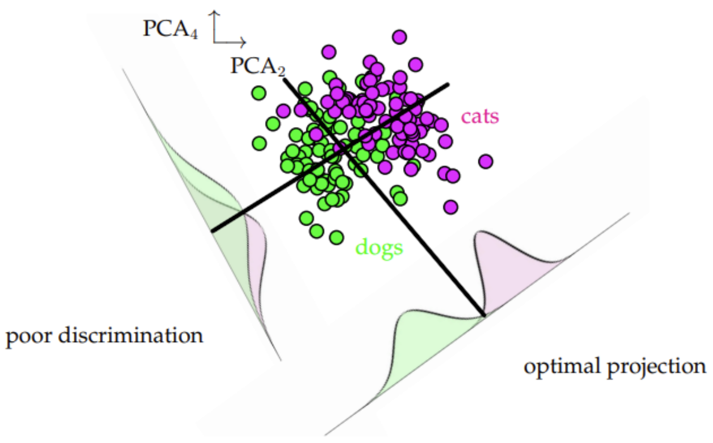


Figure 1: Data Points Projection onto the Normal Lines of an LDA Decision Line (Right) and an Non-Optimal Decision Boundary (Left). Reprinted from *Data Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* by Steven L. Brunton and J. Nathan Kutz

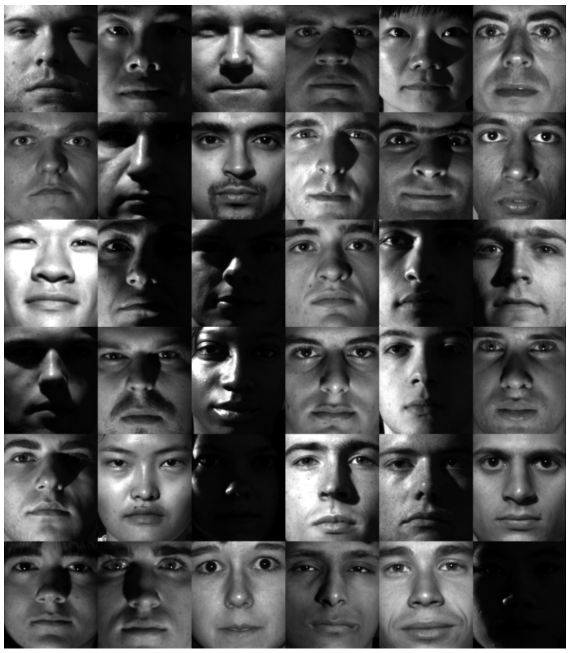
Whether thought of as a projection or a maximization, discriminant analysis gives a quantifiable and systematic way to predict to which class a new data vector belongs to. Note that both LDA and QDA have some limitations. For instance, it relies on the assumption that the data from each class is normally distributed (*i.e.*, a Gaussian distribution).

2 Methods

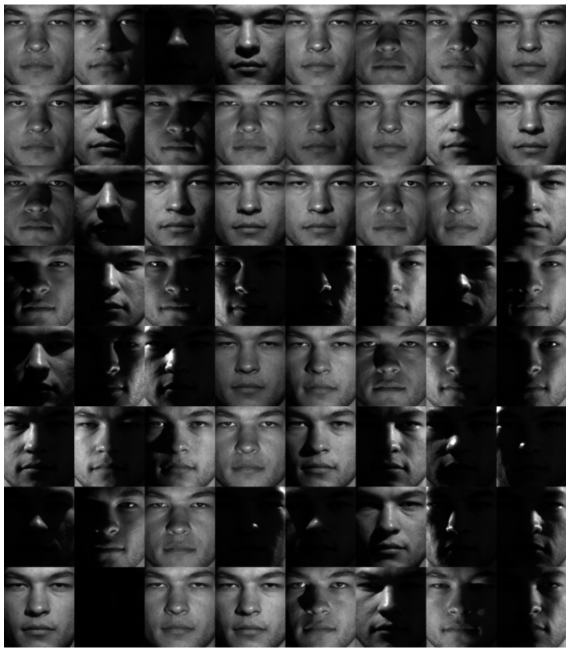
Singular value decomposition as well as linear discriminant analysis, implemented in a Python script, were used to create a working facial recognition software. The most important sections are detailed in the Appendix. The complete source code is available as a *Google Colab* file linked here. The model is built upon *Extended Yale Face Database B* [3], containing centered pictures of the faces of 38 people. The database can be accessed here [3]. Note that the website is often out of service. The pictures were formatted into a data matrix, which underwent singular value decomposition such that any image can be projected onto the basis of the left singular vectors. Then, the compressed images were sorted by person, making it possible - through linear discriminant analysis - to determine the identity of a new face image of a person who is already in the database. All numerical computations were performed using the Numpy Python module [4] [10]. Images were plotted using the Matplotplib module [5].

2.1 Data Retrieval and Processing

The *Extended Yale Face Database B* [3] is a database containing 64 192×168 centered images under various lighting conditions for 38 subjects - a total of 2432 images [Figure 2]. After downloading the data set, the images were retrieved and converted from Portable Grey Map (PGM) files to 192×168 arrays and then to 32 256 dimensional vectors. Those vectors were stored to a Python dictionary with the subjects' identification code as keys (from yaleB01 to yaleB39, excluding yaleB14) [Appendix B]. The images were then split into 3 distinct sets: a first testing set composed of the pictures of the last two individuals, a second testing set containing one image from each person, and the training set composed of the rest of the images [Appendix C]. The training group consists of 36 of the 38 individuals [Figure 2a] resulting in a data array of size $32\,256 \times 2\,260$. The first testing set will be used to test facial reconstruction and image compression capabilities using PCA. The second one will be used to test the facial recognition potential of the combination of PCA and LDA.



(a) 36 of the 38 Subjects of the *Extended Yale Face Database B*



(b) The 64 Pictures of Subject yaleB38 under different lighting conditions

Figure 2: Composition of the *Extended Yale Face Database B*

2.2 ***Eigen Faces*** [2]

The average face was calculated from the training set. It was subtracted from the training set – normalizing the data [Figure 3] [AppendixD].

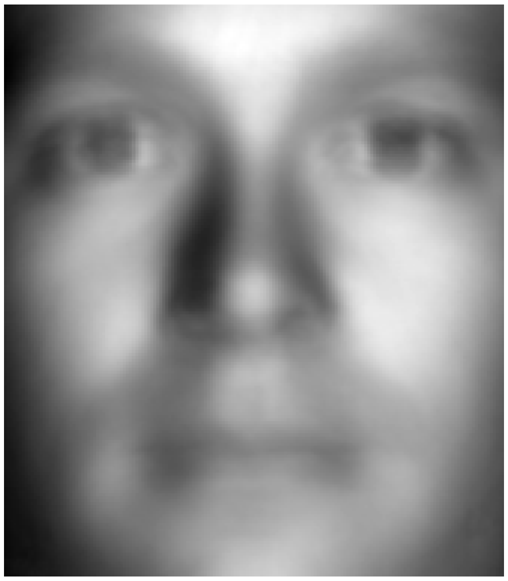


Figure 3: Average Face of the Training Data Set

Then, using SVD, this large training data array is factorized into left and right singular matrices as well as the corresponding diagonal matrix with respective sizes of $32\,256 \times 2\,260$, $2\,260 \times 2\,260$, and $2\,260 \times 2\,260$ [Appendix D]. We can then look at the left singular vectors, which can be interpreted as the *eigen faces* of the training data set. These left singular vectors denote distinctive patterns and characteristics across the data from the most common (the first *eigen faces*/left singular vector) to more specific features.

To test the potential of singular value decomposition in image compression, a few of the pictures of the two individuals in the testing set were normalized (subtracting the mean face from the training set) and projected onto a portion of the left singular vectors [Appendix D]. The images were projected onto 25, 100, 400, 800, and 1 600 first of the 2 260 left singular vectors [Appendix D]. The images were then transformed back summed with the training average face to observe the resemblance between the original and reconstructed images [Appendix D].

2.3 Clustering and Facial Recognition

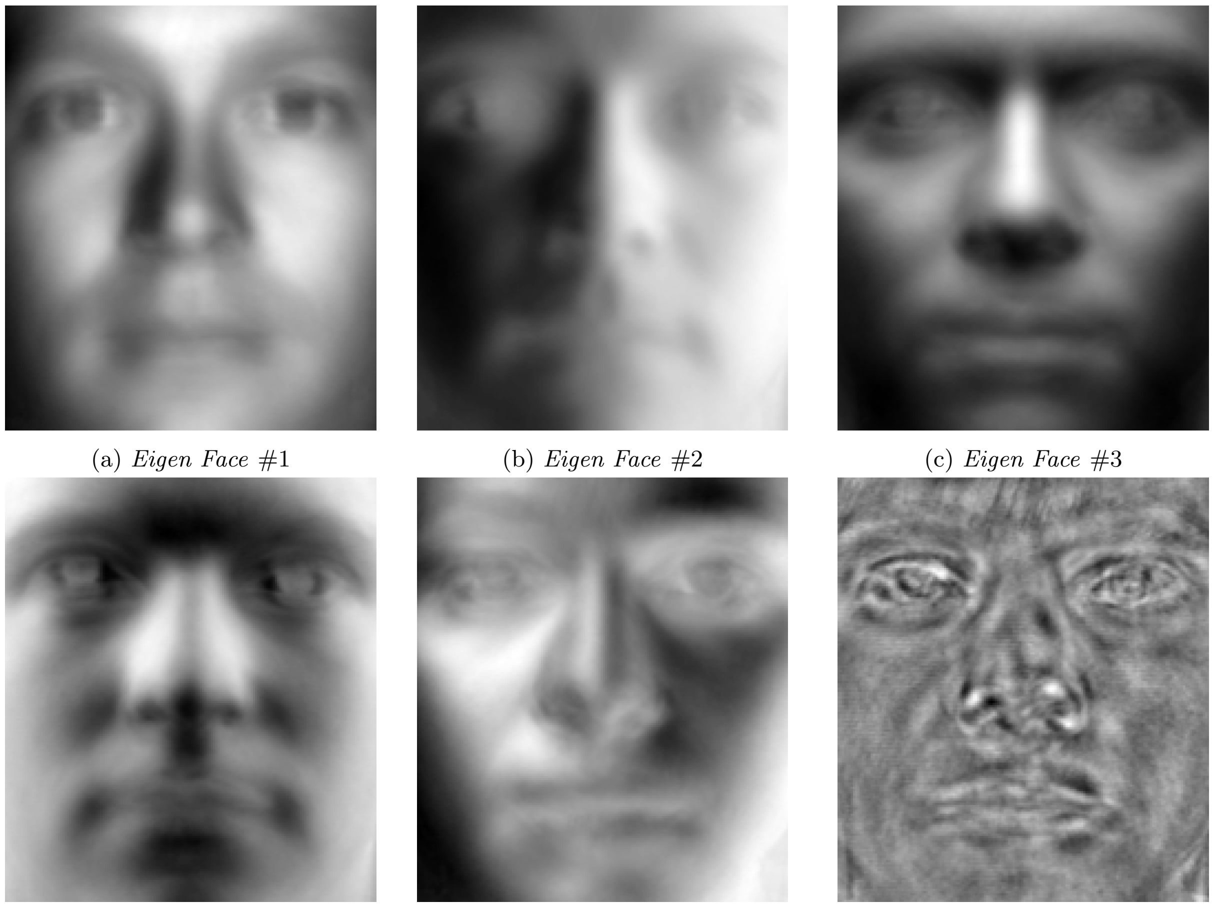
For the exploration of facial recognition The same processed outlined above was followed for the normalization and the singular value decomposition (the principle component analysis) of the data array. Then, each normalized image vector in the training set was projected onto the 300 first left singular vectors. These 300-dimensional image vectors were categorized by individual [2] [Appendix E].

Differences between faces of individuals can be visualized by looking at the prominence of specific *eigen faces* in their facial composition. This can be done by projecting all of an individual's picture onto two or three arbitrary left singular vectors and observing the difference between the distribution associated to distinct individuals. By plotting these two or three dimensional vectors, we can observe the separation between distributions. This allows to make a qualitative prediction of what individual is represented in a new picture. We plotted the projections of the images of subjects yaleB04 and yaleB22 onto the second and third left singular vectors as well as onto *eigen faces* #7, #10, and #14. Note that those choices of left singular vectors are purely arbitrary – exhibiting the underlying flaw of this approach. We also plotted the vectors from the testing set associated to these individuals to observe how they fall in the distribution. This is a very qualitative approach as it can be misleading and the choice of *eigen faces* to project onto can greatly affect the definiteness of the results [2].

A more accurate and systematic approach to determining the identity of an individual in a new picture is through discriminant analyses. Here, linear discriminant analysis is used over quadratic discriminant analysis, as LDA requires much less computations. LDA calculates only one covariance matrix instead of one for each individual, making the process much faster for data sets with a large number of classes like the *Extended Yale Face Database B* [3]. The 300-dimensional image vectors were categorized by individual underwent linear discriminant analysis using the *Scikit-learn* Python module [8] [10] [Appendix E]. To test the effectiveness of the discriminant function in recognizing and telling apart people, each image from the testing set was normalized (using the average face from the training set), projected onto the 300 first left singular vectors and then inputted in the linear discriminant function [Appendix E].

3 Results

The left singular vectors, which can be interpreted as the *eigen faces*, were found using singular value decomposition Figure 4. The first few *eigen faces* illustrate common facial features across the data set. We can see the first left singular vector represents basic face features like eyes, a nose, a mouth, and general facial structure which is prominent amongst all subjects [Figure 4a]. *Eigen faces* further down the line on the other hand depict more isolated and rare facial characteristics. For instance, the 200th left singular vector seems to account for bangs (*i.e.*, hair hanging over the forehead) [Figure 4f].

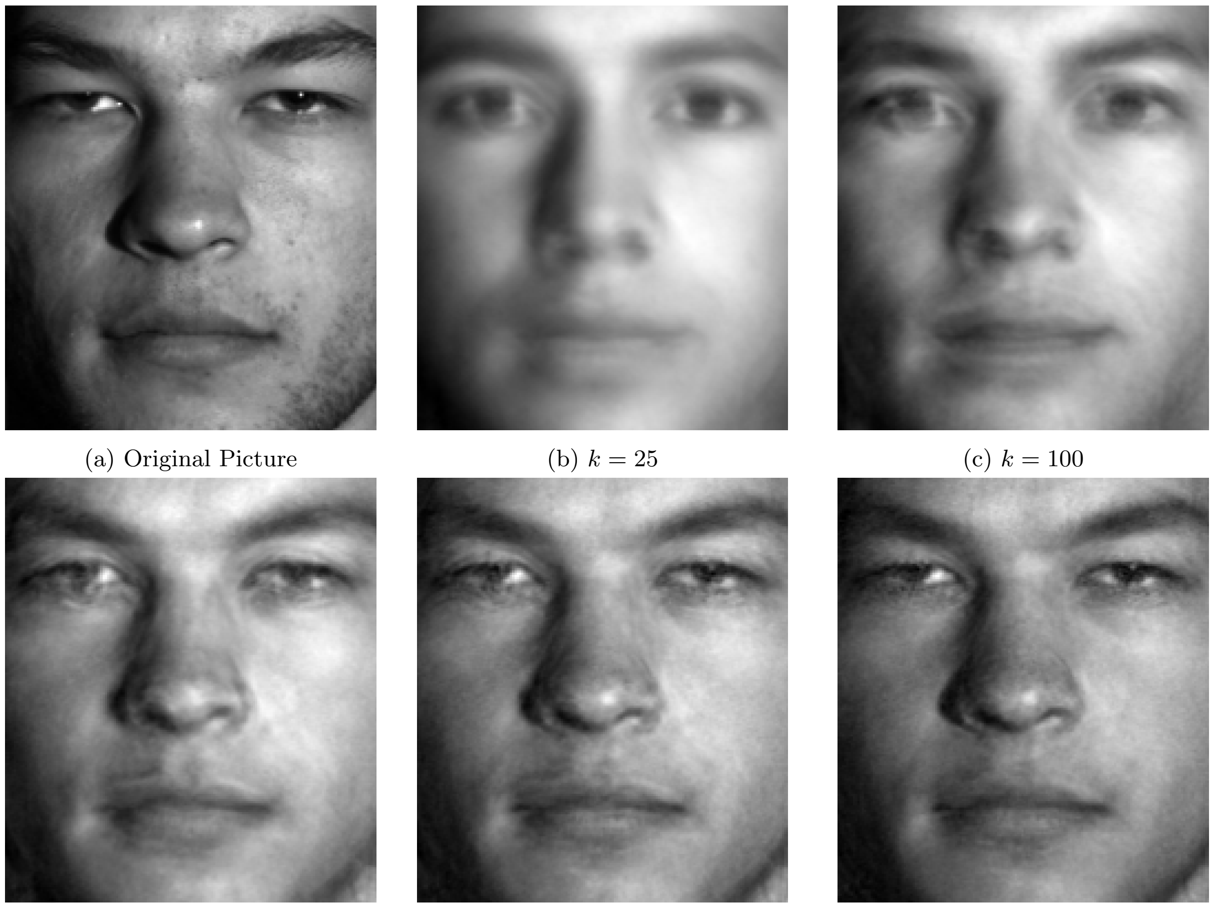


(a) *Eigen Face* #1 (b) *Eigen Face* #2 (c) *Eigen Face* #3

(d) *Eigen Face* #4 (e) *Eigen Face* #5 (f) *Eigen Face* #200

Figure 4: 6 of the 2 304 *Eigen Faces* of the Training Data Set

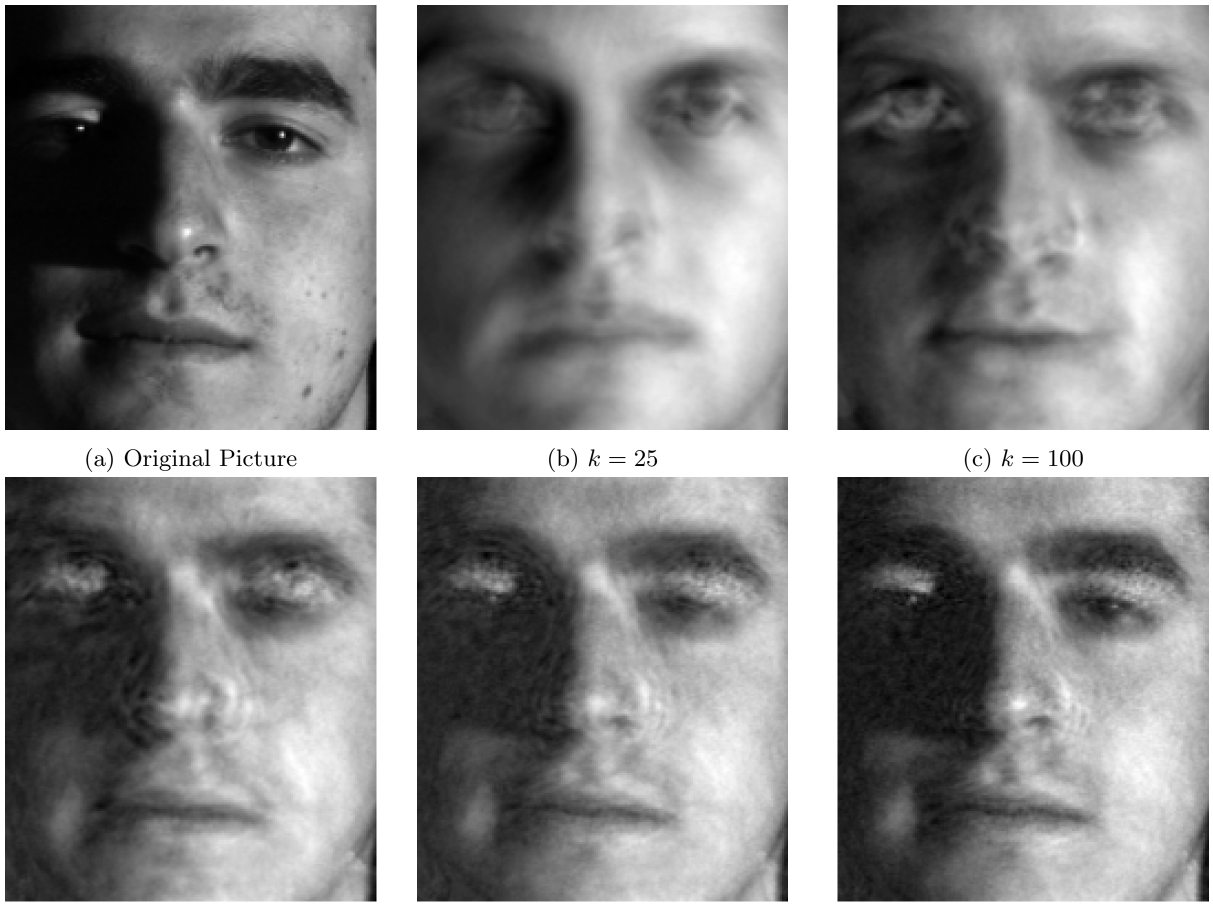
Images of the two individuals in the testing set (yaleB38 and yaleB39) were successfully compressed and decompressed using the the truncated version of the left singular matrix with rank 25, 100, 200, 400, and 1600 [Figure 5] [Figure 6]. Although, they are not identical, we can start recognizing specific facial characteristics as k increases [Figure 5] [Figure 6].



(a) Original Picture (b) $k = 25$ (c) $k = 100$

(d) $k = 400$ (e) $k = 800$ (f) $k = 1600$

Figure 5: Facial Reconstruction of Individual yaleB38 when Projected Onto a Portion of the Left Singular Vectors


(a) Original Picture (b) $k = 25$ (c) $k = 100$

(d) $k = 400$ (e) $k = 800$ (f) $k = 1600$

Figure 6: Facial Reconstruction of Individual yaleB39 when Projected Onto a Portion of the Left Singular Vectors

We also observed the distributions of subjects yaleB04 and yaleB22 by projecting their images onto the left singular vectors #2 and #3 [Figure 7] as well as onto vectors #7, #10, and #14 [Figure 8]. We notice that there is no evident separation between the distribution of individuals yaleB04 and yaleB22 when projected on *eigen faces* #2 and #3 [Figure 7]. On the other hand, when projected onto vectors #7, #10, and #14, there is a clear difference between the two distributions, making it possible to qualitatively classify the two test images looking at there position with respect to the distribution [Figure 8]. Note that this qualitative analysis can be influenced greatly by perspective and can be misleading.

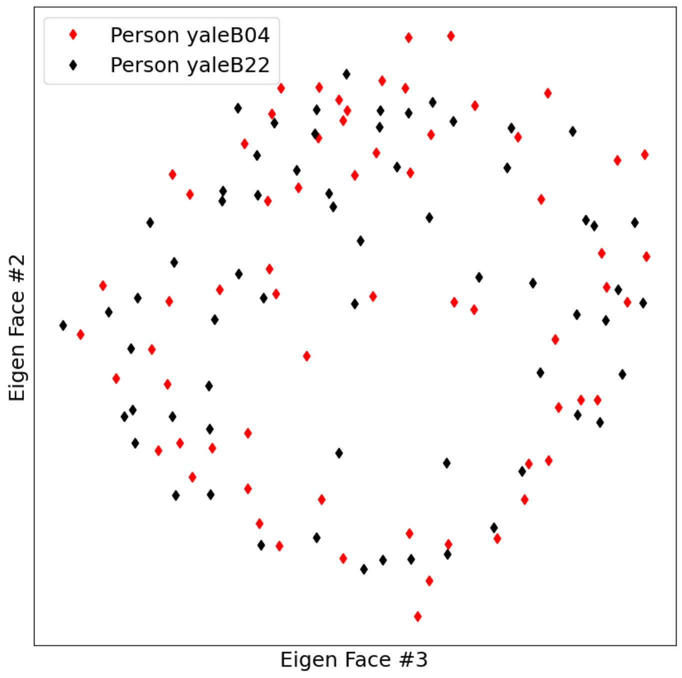


Figure 7: Projection of the Images of Subjects yaleB04 and yaleB22 Onto the Second and Third Left Singular Vectors

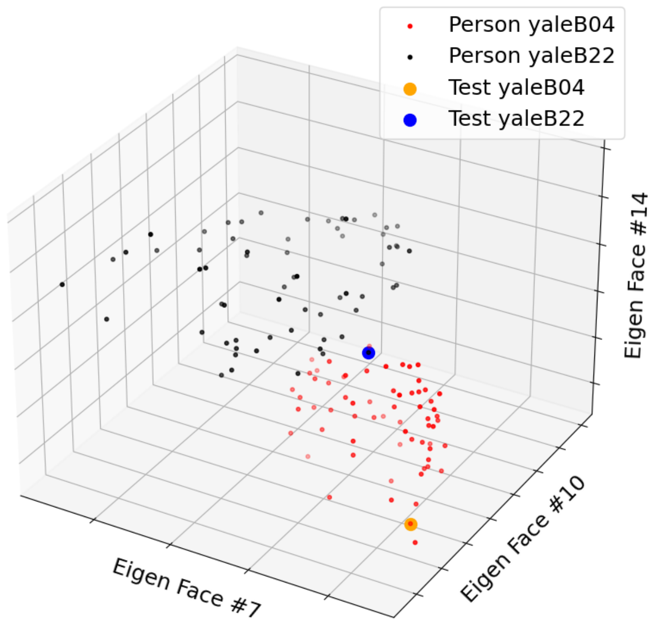


Figure 8: Projection of the Images of Subjects yaleB04 and yaleB22 Onto the Seventh, tenth, and fourteenth Left Singular Vectors

Taking a more systematic approach, using linear discriminant analysis, the application was able to to recognize the individual in all 38 test images without fail [Appendix F].

4 Analysis and Conclusions

The goal of this research project was to investigate various linear algebra concepts like singular value decomposition and discriminant analysis in the scope of image treatment and compressing as well as computer vision. More specifically this paper explores the feasibility of creating a facial recognition software using SVD and LDA. The *Extended Yale Face Database B* [3] composed of centered pictures of 38 individuals under varying lighting conditions was used to investigate this.

We were successful in compressing and decompressing images using the singular value decomposition of the data matrix composed of the pictures of a portion of the subjects represented as column vectors of pixel values [Figure 5] [Figure 6]. This was done by using a portion of the left singular vectors as a basis, compressing pictures of dimension 32 256 to dimensions of 25 [Figure 5b] [Figure 6b], 100 [Figure 5c] [Figure 6c], 400 [Figure 5d] [Figure 6d], 800 [Figure 5e] [Figure 6e], and 1600 [Figure 5f] [Figure 6f] with varying degrees of precision and accuracy. We notice that the faces is barely distinguishable for low rank values like 25 [Figure 5b] [Figure 6b] and 100 [Figure 5c] [Figure 6c]. Nevertheless, as the rank increases the faces become closer and closer to the original pictures. Already at a rank of 800 [Figure 5e] [Figure 6e] and 1600 [Figure 5f] [Figure 6f] we can associate the reconstructions to the individual.

This means, that if in possession of the left singular matrix, one could store and process these pictures as 1 600 dimensional vectors instead of 32 256 dimensional vectors, remaining relatively authentic to the original photo. These compressed images are more than 20 times lesser in size then their original counterparts. This method of image storing and treatment could be especially useful for large data sets where the compressed size advantage would compound quickly.

Note that this compression technique is only truly accurate and advantageous for face images taken with the same specification as in the *Extended Yale Face Database B*. This compression technique relies on selecting the predominant and most important trends in a data set to give an approximation of an image in terms of these features. If these features are not present in an image, the approximation will simply not be as good as if they were. It is possible to express any image using a portion of the left singular vectors as a basis; however, the number of vectors needed to render an accurate reconstruction of an image will be much greater in a picture where those *eigen features* are not as apparent, limiting the size reduction advantage sought using this compression technique.

Apart from compressing face images, we also explored how pictures of different individuals contrast each other when projected onto specific left singular vectors. By projecting the pictures of individuals yaleB04 and yaleB22 on *eigen faces* #2 and #3, we do not observe any significant separation between the two distributions [Figure 7]. Conceptually, this is as expected, as the first few left singular vectors depict the most prominent features across the whole data set [Figure 4]. These basic features are often structural ones like the presence of eyes, a mouth, a nose, facial shape, which are relatively constant across all subjects [Figure 4]. This means, that we can expect all individuals to have similar distributions when projected onto the first few *eigen faces*.

We have to pick left singular vectors a bit further down the line to observe a considerable difference in the distribution of two individuals. The separation between the two individuals is apparent when projected onto *eigen faces* #7, #10, and #14 [Figure 8]. The distributions seem to cluster exhibiting a significant difference between the two individuals with respect to these three *eigen features* [Figure 8]. By plotting the two test images associated to yaleB04 and yaleB22, we can see that they fall near or within the clusters of their corresponding identity [Figure 8].

This method of classifying images is highly inefficient an inaccurate, as the choice of left singular vectors to project onto is arbitrary and does not necessarily guarantee clear separation between the distributions. Furthermore, this qualitative approach is flawed in the way that it relies heavily on perception. In other words, a new test image might seem well into a distribution from one perspective, but far from it from another view. This method is also limited as it does not allow to examine differences between individuals in more than 3 dimensions as it intrinsically relies on visualization.

This reliance on visualization and perspective is negated when using discriminant analysis. Discriminant analysis is an objective way to quantify the likely hood of a test image being of a specific individual based on the data set. Discriminant analysis allows us to quantify the separation between classes across as many left singular vectors as we want. Using linear discriminant analysis across the first 300 *eigen faces*, we were able to predict the identity of the test images of the 38 subjects with 100% accuracy rate [Appendix F]. This shows that this process of finding the individual that maximizes the discriminant function for a test image is a objective method to make precise and accurate predictions.

This research paper was successful in investigating the properties of singular value decomposition and discriminant analysis and their applications in facial recognition and image compression. We were able to express facial images as a linear combination of *eigen faces* using a truncated version of the left singular matrix compressing images by a factor of 20. Also, using these compressed images, we were able to create a working facial recognition model relying on linear discriminant analysis.

It has to be noted that all of these results, including the facial recognition software, is highly sensitive to image input format. All of these facial images taken from the *Extended Yale Face Database B* are formatted such that facial features like the nose and eyes are always in the same position across individuals. Our findings heavily rely on that assumption as the exploited *eigen features* are only significant and apparent if they are constantly in the same place. Hence, any change in position of the faces within the pictures (a translation or a rotation) would probably lead to inconclusive results.

This limitation extends to facial expressions. All of the images used depict people with similar neutral facial expressions. Different facial expressions are, therefore, not accounted for in the *eigen features*. A different facial expression could significantly diminish the software's ability to associate a picture to an individual.

Also, it must be noted that darker skin tones are not well represented in the data set [Figure 2a. This lack of skin tone diversity could result in the model being less effective on darker skinned individuals. Sadly, this weakness is not isolated to this model. This trend extends to many computer vision softwares, as they are often created and trained on white individuals.

For future research, it would be interesting to extend the capabilities of this facial recognition software by extending the training data set to include various facial expressions and more ethnic diversity. It would also be interesting to explore the possibility of combining a facial detector software with this facial recognition model to extract and process faces from non-formatted images.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006. URL: `https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf`.

[2] Steven L. Brunton and J. Nathan Kutz. *Data Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* 2017. URL: `http://databookuw.com/databook.pdf`.

[3] Athinodoros Georghiades, Peter Belhumeur, and David Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:643–660, 06 2001. `doi:10.1109/34.927464`.

[4] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. `doi:10.1038/s41586-020-2649-2`.

[5] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

[6] Daniela; Hastie Trevor; James, Gareth; Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R: Second Edition.* Springer, 2021. URL: `https://hastie.su.domains/ISLR2/ISLRv2_website.pdf`.

[7] Peter J. Olver and Shakiban Chehrzad. *Applied Linear Algebra: Second Edition.* Springer, 2018. URL: `https://warin.ca/ressources/books/2018_Book_AppliedLinearAlgebra.pdf`.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012. URL: `http://arxiv.org/abs/1201.0490`, `arXiv:1201.0490`.

[9] David Poole. *Linear Algebra A Modern Introduction: Third Edition.* Brooks/Cole, 2011. URL: `https://edisciplinas.usp.br/pluginfile.php/5572235/mod_resource/content/1/David%20Poole%20-%20Linear%20Algebra_%20A%20Modern%20Introduction-Brooks%20Cole%20%282011%29.pdf`.

[10] Python Core Team. *Python: A dynamic, open source programming language.* Python Software Foundation, 2021. Python version 3.10. URL: `https://www.python.org/`.

[11] Gilbert Strang. *Linear Algebra and learning from data.* Wellesley-Cambridge Press, 2019.

A Appendix: Similar Matrices Have the Same Characteristic Polynomial Proof

If $B = Q^{-1}AQ$, then

$$\begin{aligned}\det(B - \lambda I) &= \det(Q^{-1}AQ - \lambda Q^{-1}Q)\\ &= \det(Q^{-1}(A - \lambda I)Q) = \det(Q^{-1})\det(A - \lambda I)\det(Q)\\ &= \det(A - \lambda I)\end{aligned}$$

$$A \sim B \to A \text{ and } B \text{ have the same characteristic polynomial}$$

B Appendix: Data Extraction

This code was used to do extract the previously downloaded pictures from the *Yale Extended Face Database B* [3]. It converts each 168×192 PGM image array into a 32 256-dimensional vector. All of the image vectors from each person are appended to respective lists, which are then added to a global dictionary with the person's ID as its key.

```
yale = ['yaleB{:02d}'.format(i) for i in range(1, 40)]
yale.remove('yaleB14')

AllFaces = {}

for i in os.listdir('./CroppedYale'):
  print(i)
  p = os.path.join('./CroppedYale', i)
  listTemp = []
  for j in os.listdir(p):
    if j.endswith('.pgm'):
      f = os.path.join(p, j)
      img = np.array(imageio.imread(f))
      imgVec = img.reshape(-1,1)
      shp = int(imgVec.shape[0])
      if shp == 168*192:
        listTemp.append(imgVec)
  AllFaces[i] = listTemp
```

C Appendix: Training and Testing Sets

This code was used to separate the data into a training set and two testing sets. The first testing set is used for the facial reconstruction effort, testing the compression capabilities of principal component analysis. It contains all the pictures of the last two individuals in the data set. The second testing set is composed of the last picture of each individual and is used to test the facial recognition abilities of the software. The training set contains the rest of the images classified by individuals.

```
trainFaces = {}
testFaces = {}
testFaces2 = {}

num = 38
testNum = 2

for i in range(num):
  if num-i-testNum>0:
        trainFaces[yale[i]] = AllFaces[yale[i]][:-1]
        testFaces2[yale[i]] = AllFaces[yale[i]][1]
    else:
        testFaces[yale[i]] = AllFaces[yale[i]]

A = []
for i, val in trainFaces.items():
    for j in val:
        A.append(j)
A = np.squeeze(np.array(A))

B = []
for i, val in testFaces.items():
    for j in val:
        B.append(j)
B = np.squeeze(np.array(B))
```

D Appendix: Facial Reconstruction

Using principal component analysis, this code was used to single out the most prominent features in the training data set. Then, an image from the first testing data set was projected onto various numbers of left singular vectors to investigate the similarity of the projection to the original image.

```
avgFace = np.mean(A, 0)
avgMat = np.tile(avgFace, (awidth, 1))
An = A-avgMat

U,S,V = np.linalg.svd(An.T, full_matrices=False)
S = np.diag(S)

# Facial Reconstruction Test
test = B[100]
testn = test-avgFace

r_list = [25,100,400,800]

for r in r_list:
    Ur = U[:,:r]
    alpha = Ur.T@testn
    reconTest = Ur@alpha+ avgFace
```

E Appendix: Linear Discriminant Analysis

This code uses linear discriminant analysis to classify and distinguish individuals from new pictures. It starts by projecting every normalized image in the training data set onto the 300 first previously found left singular vectors. This matrix undergoes linear discriminant analysis. Using the LDA class from SkLearn, we can then predict and recognize the individuals in new images.

```
X = []
person = []
r = 300
Ur = U[:,:r]
for i, val in trainFaces.items():
  for j in val:
    xn = np.squeeze(np.array(j))-avgFace
    alpha = Ur.T@xn
    X.append(alpha)
    person.append(i)
  print(i)
person = np.array(person)

lda = LDA(n_components=35)
lda.fit(X, person)

# Facial Recognition Test
Ur = U[:,:r]
for i, val in testFaces2.items():
  print(f'test: {i}')
  name.append(i)
  xn = np.squeeze(np.array(val))-avgFace
  new_face = Ur.T@xn

  predicted_class = lda.predict([new_face])[0]
  if i!=predicted_class:
    print('Fail')
  else:
    print('Success')
  print(f'prediction: {predicted_class} \n')
```

F Appendix: Facial Recognition Results

Individual	Prediction	Status
yaleB01	yaleB01	Success
yaleB02	yaleB02	Success
yaleB03	yaleB03	Success
yaleB04	yaleB04	Success
yaleB05	yaleB05	Success
yaleB06	yaleB06	Success
yaleB07	yaleB07	Success
yaleB08	yaleB08	Success
yaleB09	yaleB09	Success
yaleB10	yaleB10	Success
yaleB11	yaleB11	Success
yaleB12	yaleB12	Success
yaleB13	yaleB13	Success
yaleB15	yaleB15	Success
yaleB16	yaleB16	Success
yaleB17	yaleB17	Success
yaleB18	yaleB18	Success
yaleB19	yaleB19	Success
yaleB20	yaleB20	Success
yaleB21	yaleB21	Success
yaleB22	yaleB22	Success
yaleB23	yaleB23	Success
yaleB24	yaleB24	Success
yaleB25	yaleB25	Success
yaleB26	yaleB26	Success
yaleB27	yaleB27	Success
yaleB28	yaleB28	Success
yaleB29	yaleB29	Success
yaleB30	yaleB30	Success
yaleB31	yaleB31	Success
yaleB32	yaleB32	Success
yaleB33	yaleB33	Success
yaleB34	yaleB34	Success
yaleB35	yaleB35	Success
yaleB36	yaleB36	Success
yaleB37	yaleB37	Success