EE69210: Machine Learning for Signal Processing Laboratory
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Dimensionality Reduction

Anirvan Krishna | Roll no. 21EE38002

Keywords: Principal Component Analysis, Singular Value Decomposition

Grading Rubric

	Tick the best applicable per row			Points
	Below Expectations	Lacking in Some	Meets all Expectations	
Completeness of the report				
Organization of the report (5 pts)				
Quality of figures (5 pts)				
PCA Dataset 1 experiment concepts (20 pts)				
PCA Dataset 2 experiment concepts (25 pts)				
SVD Dataset 1 experiment concepts (20 pts)				
SVD Dataset 2 experiment concepts (25 pts)				
			TOTAL (100 pts)	

1. Principal Component Analysis (PCA)

1.1 Dataset-1: Synthetic data

1.1.1 Dataset generation, visualization and zero-centering

We generate a synthetic dataset of the form

$$\mathbf{X}^* = \left[\begin{bmatrix} x^*(0) \\ x^*(1) \\ x^*(2) \end{bmatrix}, \begin{bmatrix} x^*(0) \\ x^*(1) \\ x^*(2) \end{bmatrix}_1, \ldots, \begin{bmatrix} x^*(0) \\ x^*(1) \\ x^*(2) \end{bmatrix}_i, \ldots, \begin{bmatrix} x^*(0) \\ x^*(1) \\ x^*(2) \end{bmatrix}_{99} \right] = \begin{bmatrix} x^*(0) \\ x^*(1) \\ x^*(2) \end{bmatrix} \tag{1}$$

Here, $x^*(0)$, $x^*(1)$ and $x^*(2)$ are drawn as follows:

$$x^*(0)_i \sim U[0, 100] \tag{2}$$

$$x^*(1)_i = x^*(0)_i + \varepsilon_i; \quad \varepsilon_i \sim U[-10, 10] \tag{3}$$

$$x^*(2)_i = x^*(0)_i + \varphi_i; \quad \varphi_i \sim U[-1, 1] \tag{4}$$

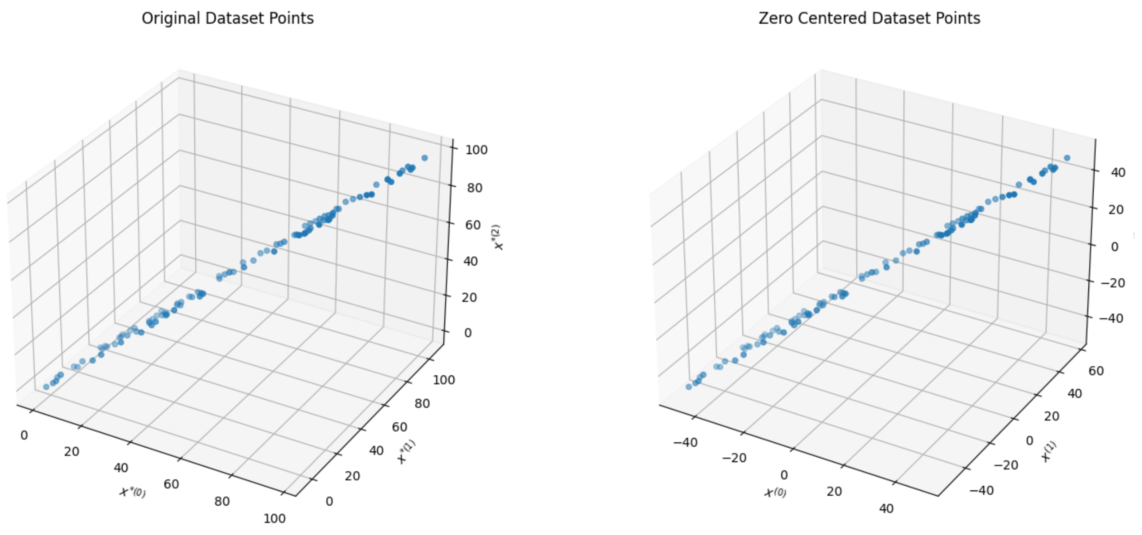

Fig. 1. Point cloud representation of $\mathbf{X}^*, \mathbf{X} \in \mathbb{R}^{3\times 100}$

We define $\mathbf{X} = \mathbf{X}^* - \mu$ where μ represents the mean of $\mathbf{X}^*$

1.2 Computation of correlation matrix and eigenvectors

We define the correlation matrix $\mathbf{R}_X$ as

$$\mathbf{R}_X = \frac{1}{n}\mathbf{X}\mathbf{X}^T \tag{5}$$

Where n is the number of samples. We then perform eigen decomposition on the correlation matrix $\mathbf{R}_X$ to obtain the eigenvalues and eigenvectors.

$$\mathbf{R}_X\mathbf{A} = \mathbf{A}\Lambda \tag{6}$$

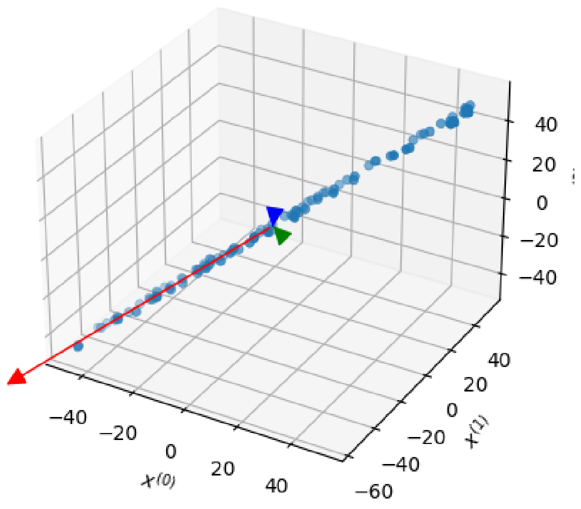

Fig. 2. 3D visualization of the eigenvalue-weighted eigenvectors of $\mathbf{R}_X$

where, $\mathbf{A}$ is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues. We get:

$$\mathbf{R}_X = \begin{bmatrix} 780.8734 & 758.3828 & 783.4875 \\ 758.3828 & 767.1951 & 760.1514 \\ 783.4875 & 760.1514 & 786.4848 \end{bmatrix} \tag{7}$$

$$\Lambda = \begin{bmatrix} 2313.03 \\ 21.35 \\ 0.17 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} -0.58 & -0.72 & 0.38 \\ -0.57 & 0.02 & -0.82 \\ -0.58 & 0.70 & 0.42 \end{bmatrix} \tag{8}$$

1.2.1 Data projection on the eigenvectors

We project the given data onto the eigenvectors to find the matrix $\mathbf{Y}$

$$\mathbf{Y} = \mathbf{A}^T\mathbf{X} \tag{9}$$

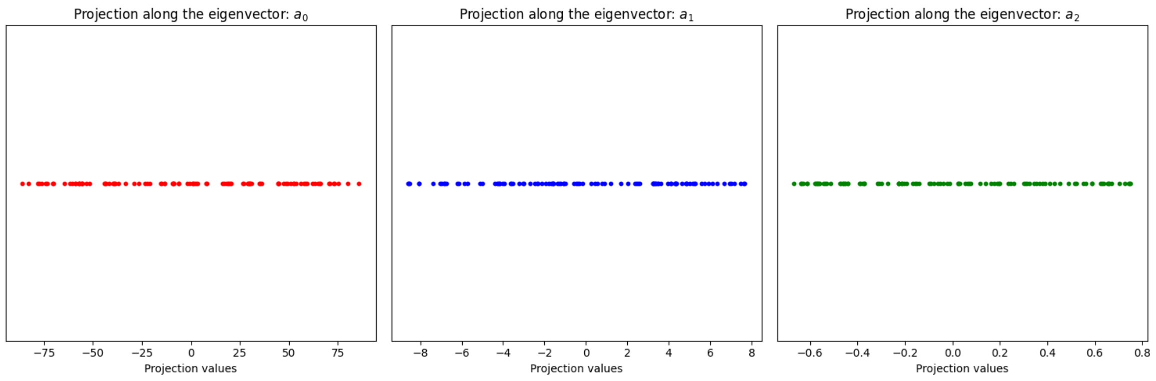

Fig. 3. Projection of X on the eigenvectors $a^{(0)}, a^{(1)}$ and $a^{(2)}$

1.2.2 Estimation of R_Y

We define the correlation matrix R_Y as:

$$\mathbf{R}_Y = \frac{1}{n}\mathbf{Y}\mathbf{Y}^T = \frac{1}{n}(\mathbf{A}^T\mathbf{X})(\mathbf{A}^T\mathbf{X})^T = \mathbf{A}\mathbf{R}_\mathbf{X}\mathbf{A}^T \tag{10}$$

$$\mathbf{R}_Y = \frac{1}{n}\mathbf{Y}\mathbf{Y}^T = \begin{bmatrix} 2276.19 & 0.00 & 0.00 \\ 0.00 & 20.84 & 0.00 \\ 0.00 & 0.00 & 0.18 \end{bmatrix} = \mathbf{A}\mathbf{R}_\mathbf{X}\mathbf{A}^T \tag{11}$$

1.2.3 Dimensionality reduction and recovery

We can consider representing $\mathbf{y}_i$ as a reduced-dimensional vector $\hat{\mathbf{y}}_i \in \mathbb{R}^{m\times 1}$ with $m \le D$. Thus, the reduced-dimensional equivalent of the dataset $\mathbf{X}$ can be represented as:

$$\hat{\mathbf{Y}} = \hat{\mathbf{A}}^\top \mathbf{X} \tag{12}$$

where $\hat{\mathbf{A}} \in \mathbb{R}^{D\times m}$ is defined as:

$$\hat{\mathbf{A}} = \left[\begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_0, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_1, \dots, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_j, \dots, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_{m-1} \right]. \tag{13}$$

Accordingly, we would also have:

$$\hat{\mathbf{Y}} = \left[\begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_0, \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_1, \dots, \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_i, \dots, \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_{N-1} \right]. \tag{14}$$

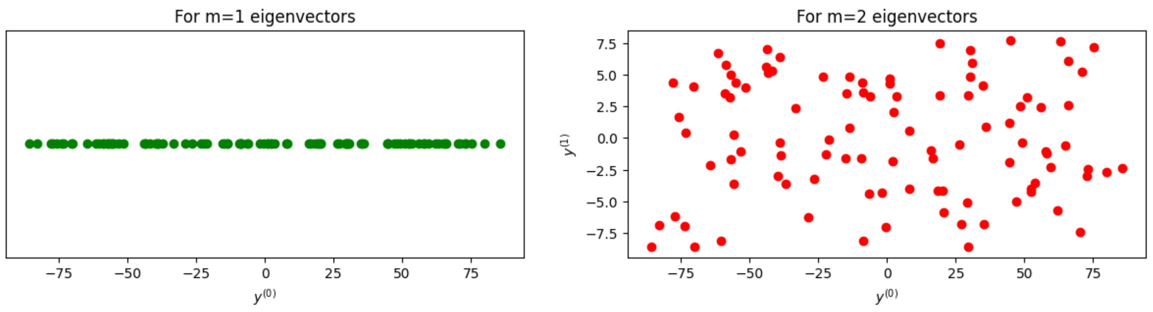

Fig. 4. Representation of compressed dataset $\mathbf{Y}$ by taking $m = 1$ and $m = 2$ principal components

We can recover the sample $\mathbf{x}_i$ through its corresponding reduced dimension form $\hat{\mathbf{y}}_i$, given as $\hat{\mathbf{y}}_i = \left[y_i^{(0)}, y_i^{(1)}, \dots y_i^{(j)}, \dots, y_i^{(m-1)}\right]$, where $\hat{\mathbf{y}}_i \in \mathbb{R}^m$ and the recovered sample $\hat{\mathbf{x}}_i$ is computed as:

$$\hat{\mathbf{x}}_i = \sum_{j=0}^{m-1} y_i^{(j)} \mathbf{a}_j. \tag{15}$$

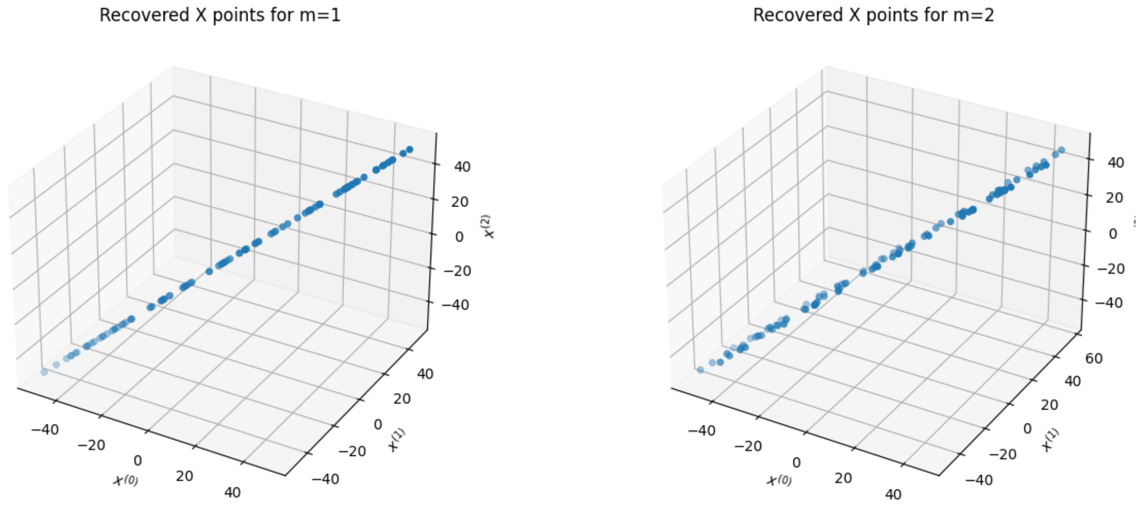

Fig. 5. Data points recovered $\hat{\mathbf{X}}$ from the compressed dataset $\mathbf{Y}$ after retention of $m = 1$ and $m = 2$ principal components

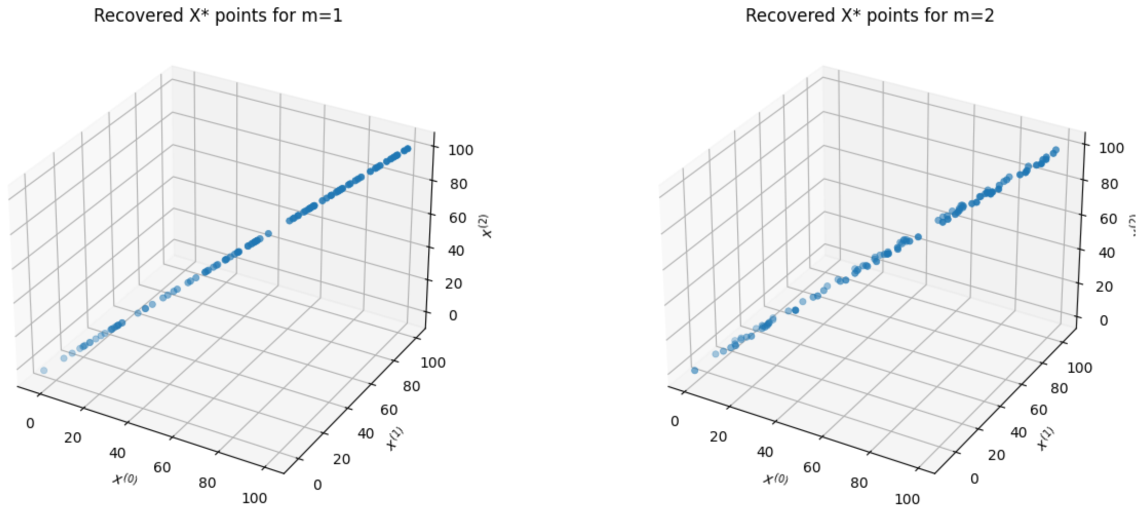

Fig. 6. Data points recovered $\mathbf{X}^*$ from the compressed dataset $\mathbf{Y}$ after retention of $m = 1$ and $m = 2$ principal components and mean addition

Mean squared error after reconstruction: In this experiment we measure:

$$MSE = ||\mathbf{X}^* - \mathbf{X}^*_{\mathbf{rec}}||_F$$

. We get MSE for m=1: 6.97, MSE for m=2: 0.055

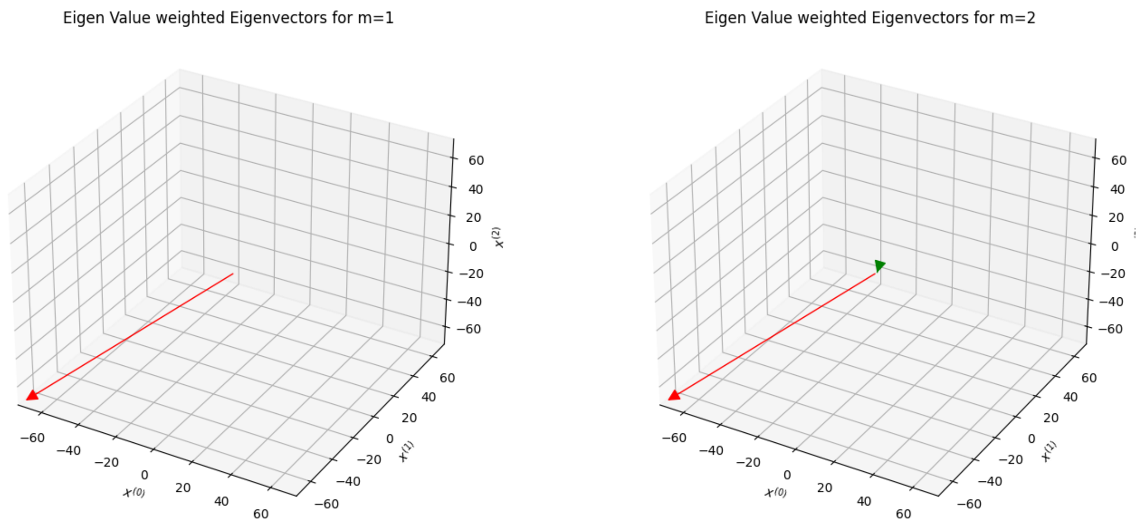

Fig. 7. Eigenvalue-weighted eigenvectors used for reconstruction after retention of $m = 1$ and $m = 2$ principal components

1.3 Dataset-2: Olivetti faces

1.3.1 Correlation matrix and eigendecomposition

For the Olivetti Faces dataset $(\mathbf{X})$ the correlation matrix $(\mathbf{R_X})$ and the eigenvalue matrix (Λ) are:

$$\mathbf{R_X} = \begin{bmatrix} 0.03 & 0.03 & 0.03 & \cdots & -0.01 & -0.01 & -0.00 \\ 0.03 & 0.04 & 0.03 & \cdots & -0.01 & -0.01 & -0.01 \\ 0.03 & 0.03 & 0.04 & \cdots & -0.01 & -0.01 & -0.01 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -0.01 & -0.01 & -0.01 & \cdots & 0.04 & 0.03 & 0.03 \\ -0.01 & -0.01 & -0.01 & \cdots & 0.03 & 0.03 & 0.03 \\ -0.00 & -0.01 & -0.01 & \cdots & 0.03 & 0.03 & 0.03 \end{bmatrix} \tag{16}$$

$$\Lambda = \begin{bmatrix} 18.79 & 0 & 0 & \cdots & 0 \\ 0 & 11.04 & 0 & \cdots & 0 \\ 0 & 0 & 6.29 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{17}$$

1.3.2 Plotting eigenfaces

Eigenfaces are a dimensionality reduction technique used in facial recognition systems, based on Principal Component Analysis (PCA). They represent facial images as a weighted combination of principal components, which are eigenvectors of the covariance matrix of a dataset of face images. These eigenvectors capture the most significant variations in the dataset, focusing on features such as contours, eyes, and mouth. Eigenfaces reduce data dimensionality, enabling efficient storage and processing while preserving facial features necessary for recognition.

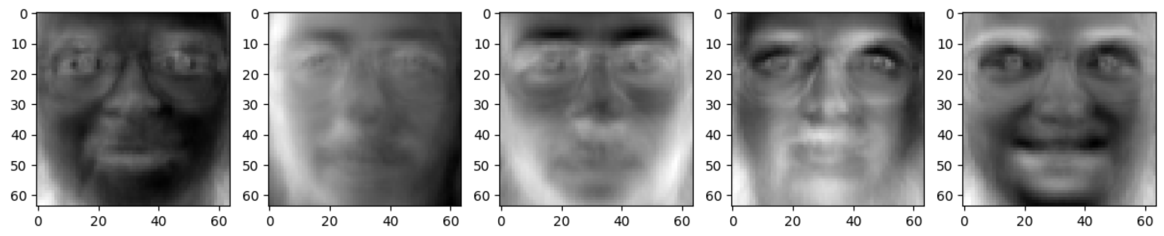

Fig. 8. Visualising the top-5 eigenvectors of the olivetti faces

1.3.3 Number of components required for a given reconstruction error

Steps for inding the Number of Components m for a Given ε_m.

Reconstruction Error Formula: The reconstruction error for Principal Component Analysis (PCA) is:

$$\varepsilon_m = 1 - \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{18}$$

where λ_i: Eigenvalues corresponding to the principal components (sorted in descending order), n is the total number of eigenvalues (or components). m number of components used for reconstruction.

The goal is to find m suchthat, $\varepsilon_m \leq \theta$, or equivalently:

$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 1 - \theta \tag{19}$$

Steps to Calculate m

(1) Sort Eigenvalues in Descending Order:
Ensure the eigenvalues are sorted in decreasing order of magnitude ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$).

(2) Compute Total Variance:
Compute the total variance as:

$$\text{Total Variance} = \sum_{i=1}^{n} \lambda_i \tag{20}$$

(3) Compute Cumulative Variance Ratio:
Compute the cumulative variance ratio for the first m components as:

$$\text{Cumulative Variance Ratio}_m = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{21}$$

(4) Determine m:
Find the smallest m such that:

$$\text{Cumulative Variance Ratio}_m \geq 1 - \theta \tag{22}$$

Dimensionality Reduction for a given m**:** Select the top k eigenvectors corresponding to the k largest eigenvalues. Form a projection matrix:

$$\mathbf{A} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k] \in \mathbb{R}^{D \times k}$$

- Project the original data onto the lower-dimensional subspace:

$$\mathbf{Z} = \mathbf{A}^\top \mathbf{X}_{\text{centered}} \in \mathbb{R}^{k \times N}$$

where $\mathbf{Z}$ is the compressed representation in the reduced k-dimensional space.

Reconstruction: Reconstruct the data from the reduced representation by projecting back to the original space:

$$\mathbf{X}_{\text{reconstructed}} = \mathbf{A}\mathbf{Z} + \mu \in \mathbb{R}^{D\times N}$$

Here, $\mathbf{A}$ maps the data back to the original d-dimensional space, and μ is added to reintroduce the original mean.

Reconstruction Error (ε_m)	$\varepsilon_m \leq 1\%$	$\varepsilon_m \leq 5\%$	$\varepsilon_m \leq 10\%$	$\varepsilon_m \leq 20\%$
Number of Components (m)	260	123	66	27

Table 1. Reconstruction Error and Components

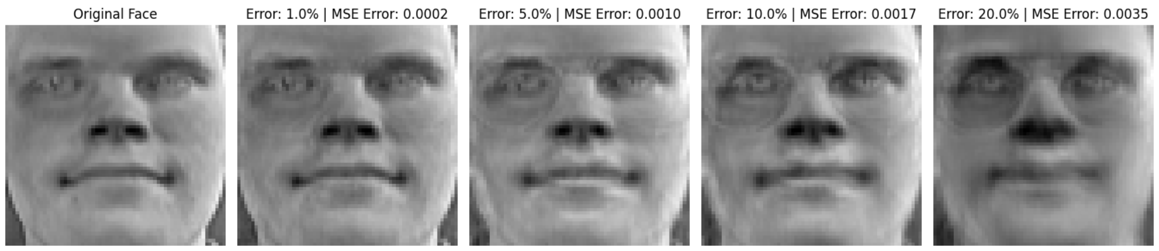
Original Face
Error: 1.0% | MSE Error: 0.0002
Error: 5.0% | MSE Error: 0.0010
Error: 10.0% | MSE Error: 0.0017
Error: 20.0% | MSE Error: 0.0035

Fig. 9. Original and recovered samples for 1%, 5%, 10% and 20% errors

2. Singular Value Decomposition

Singular Value Decomposition (SVD) is a powerful linear algebra technique used for dimensionality reduction, noise reduction, and data compression. It factorizes a matrix into three other matrices, capturing the essential patterns and structures in the data.

2.1 Dataset-1: Synthetic data

Let us generate a sample dataset $\mathbf{X} \in \mathbb{R}^{3\times 1000}$ as below:

$$\mathbf{X} = \left[\begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_0, \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_1, \cdots, \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_i, \cdots, \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_{999} \right] = \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix} \tag{23}$$

where we generate these components as:

$$x_i^{(0)} \sim \mathscr{U}[-10,10], \quad x_i^{(1)} \sim \mathscr{U}[-10,10], \quad x_i^{(2)} \sim \mathscr{U}[-1,1] \tag{24}$$

2.1.1 SVD computation and determination of top-2 dominant eigenvectors

Given a matrix $\mathbf{X}$ of dimensions $m \times n$, the SVD of $\mathbf{X}$ is given by:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \tag{25}$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix whose columns are the left singular vectors of $\mathbf{X}$. Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal, known as the singular

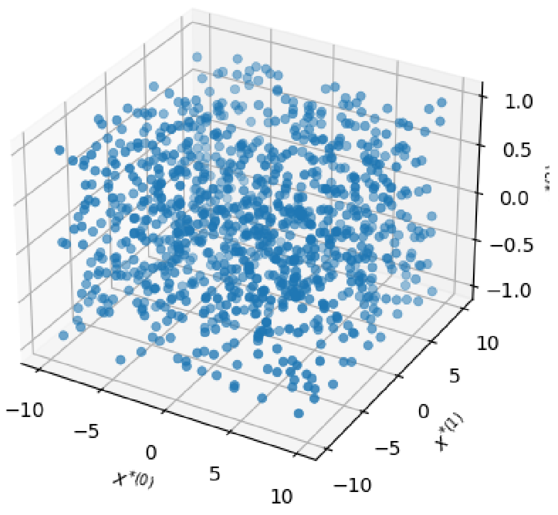

Fig. 10. 3D visualization of points in dataset $\mathbf{X}$

values of $\mathbf{X}$ and $\mathbf{V}$ is an $n \times n$ orthogonal matrix whose columns are the right singular vectors of $\mathbf{X}$.

Steps to Compute SVD

(1) Compute $\mathbf{X}^T\mathbf{X}$:
   - Calculate the matrix $\mathbf{X}^T\mathbf{X}$, which is an $n \times n$ matrix.

(2) Eigen Decomposition of $\mathbf{X}^T\mathbf{X}$:
   - Perform eigen decomposition on $\mathbf{X}^T\mathbf{X}$ to obtain the eigenvalues and eigenvectors:

$$\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}\Lambda$$

   where $\mathbf{V}$ is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

(3) Compute Right Singular Vectors:
   - The columns of $\mathbf{V}$ are the right singular vectors of $\mathbf{X}$.

(4) Compute Singular Values:
   - The singular values σ_i are the square roots of the eigenvalues λ_i:

$$\sigma_i = \sqrt{\lambda_i}$$

(5) Compute Left Singular Vectors:
   - The left singular vectors $\mathbf{U}$ are computed as:

$$\mathbf{U} = \mathbf{X}\mathbf{V}\Sigma^{-1}$$

   where Σ^{-1} is the inverse of the diagonal matrix Σ.

SVD can be used to approximate a matrix with a lower rank by truncating the singular values and corresponding singular vectors. For a given rank k, the best rank-k approximation of $\mathbf{X}$ is given by:

$$\mathbf{X}_k = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T \tag{26}$$

where $\mathbf{U}_k$, Σ_k, and $\mathbf{V}_k$ are the matrices containing the top k singular values and corresponding singular vectors. In this case we have $\mathbf{U} \in \mathbb{R}^{3\times1000}$, $\Sigma \in \mathbb{R}^{3\times1000}$, $\mathbf{V} \in \mathbb{R}^{1000\times1000}$, $\Lambda_r \in$

$\mathbb{R}^{2\times2}$, $\mathbf{U}_r \in \mathbb{R}^{3\times2}$, $\mathbf{V}_r \in \mathbb{R}^{1000\times2}$. In this case, the top 2 eigenvalues and corresponding eigenvectors are as follows:

$$\Lambda_2 = \begin{bmatrix} 185.08 \\ 181.94 \end{bmatrix}, \qquad \mathbf{U}_2 = \begin{bmatrix} 0.7241 & 0.6897 \\ -0.6897 & 0.7241 \\ 0.0044 & 0.0016 \end{bmatrix} \tag{27}$$

The projection of data points onto the subspace spanned by two principal eigenvectors can be represented as:

Given point set $\mathbf{X} \in \mathbb{R}^{N\times1}$ and two dominant eigenvectors $\mathbf{v}_1, \mathbf{v}_2$:

Project onto 2D subspace:

$$\mathbf{X}_{\text{proj}} = [\mathbf{v}_1 \ \mathbf{v}_2]^T \mathbf{X} \tag{28}$$

Each projected point $\mathbf{y}_i$ has coordinates:

$$y_{i1} = \mathbf{v}_1^T \mathbf{x}_i, \qquad y_{i2} = \mathbf{v}_2^T \mathbf{x}_i \tag{29}$$

The projection preserves the maximum variance in the data along these principal directions while reducing dimensionality from N-D to 2-D.

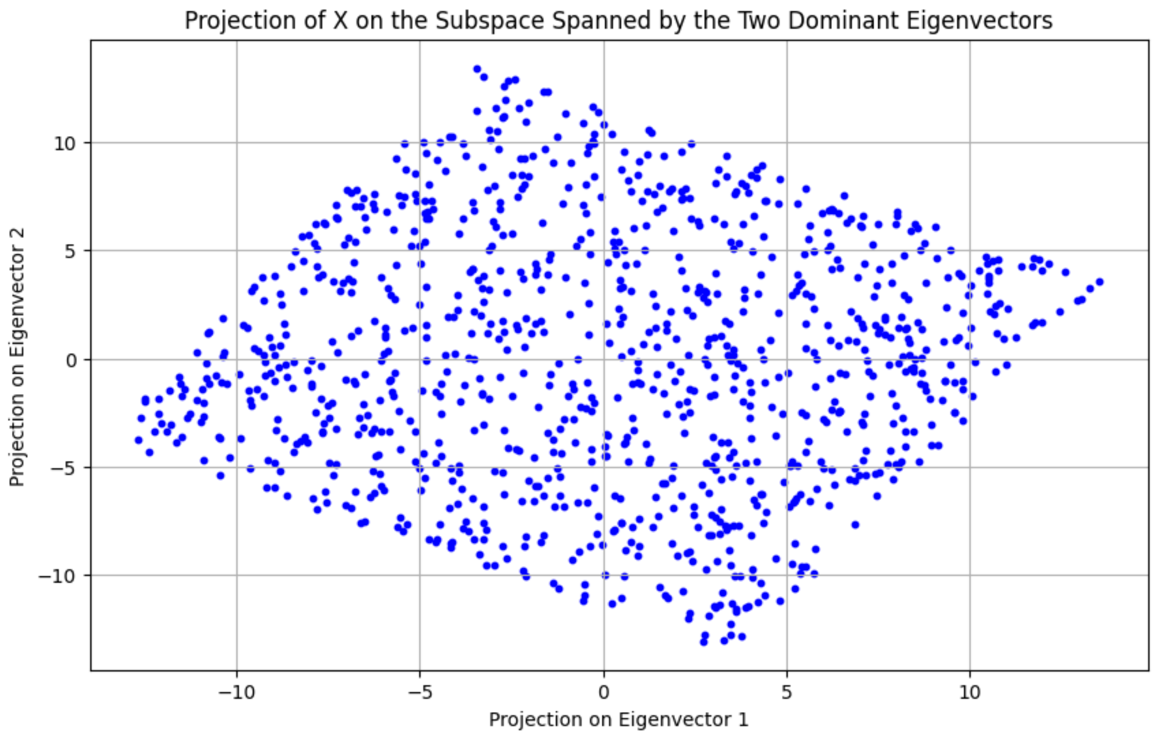

Fig. 11. Projection of datapoints in **X** to the subspace spanned by the 2 dominant eigenvectors

2.1.2 Pairwise distance computation

$$\hat{\mathbf{X}} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{k-1}] \begin{bmatrix} \sqrt{\lambda_0}\mathbf{v}_0^H \\ \sqrt{\lambda_1}\mathbf{v}_1^H \\ \vdots \\ \sqrt{\lambda_{k-1}}\mathbf{v}_{k-1}^H \end{bmatrix} = \mathbf{U}_k\mathbf{A}_k \tag{30}$$

where $\mathbf{U}_k = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{k-1}] \in \mathbb{R}^{D\times k}$ consists of the first k columns of $\mathbf{U}$, $\mathbf{A}_k = [\mathbf{a}_0, \mathbf{a}_1, \ldots \mathbf{a}_{N-1}] \in \mathbb{R}^{k\times N}$ consists of the column vectors of the product matrix $\Lambda_k^{\frac{1}{2}}\mathbf{V}_k^H$, where $\mathbf{V}_k$ consists of the first k columns of $\mathbf{V}$, and $\Lambda_k^{\frac{1}{2}}$ is the diagonal matrix having the square root of the respective k eigenvalues.

Each sample $\mathbf{x}_i$ can be approximated by a column vector $\hat{\mathbf{x}}_i$ as

$$\hat{\mathbf{x}}_i = \mathbf{U}_k\mathbf{a}_i \tag{31}$$

$$\begin{bmatrix} \hat{x}_i^{(0)} \\ \hat{x}_i^{(1)} \\ \vdots \\ \hat{x}_i^{(D-1)} \end{bmatrix}_i = \sum_{j=0}^{k-1} \mathbf{u}_j a_i^{(j)}$$

where $\mathbf{a}_i = \begin{bmatrix} a_i^{(0)} & a_i^{(1)} & \ldots & a_i^{(k-1)} \end{bmatrix}^T \in \mathbb{R}^{k\times 1}$. Simplifying it, the higher D-dimensional sample vector $\mathbf{x}_i$ can be approximated by the k-dimensional vector $\mathbf{a}_i$, lying in the subspace spanned by $\mathbf{u}_j$. This also denotes that $\mathbf{a}_i$ is the projection of $\mathbf{x}_i$ on the subspace spanned by $\mathbf{u}_j$.

Further, due to orthonormality of the columns $\mathbf{u}_j$ of $\mathbf{U}_k$, we see that the distance between two samples $\mathbf{x}_i$ and $\mathbf{x}_m$ where $i, m = 0, 1, \ldots, n-1$ can be represented as

$$\|\mathbf{x}_i - \mathbf{x}_m\| \approx \|\mathbf{U}_k(\mathbf{a}_i - \mathbf{a}_m)\| = \|\sum_{j=0}^{k-1} \mathbf{u}_j(a_i^{(j)} - a_m^{(j)})\| = \|\mathbf{a}_i - \mathbf{a}_m\| \tag{32}$$

In order to search for a pair of similar patterns $(\mathbf{x}_i, \mathbf{x}_m)$, instead of computing the distance between a pair of D-dimensional vectors, we can now compute the same value using their corresponding k-dimensional representation $(\mathbf{a}_i, \mathbf{a}_m)$. When $k \ll D$, substantial computational savings are obtained.

Here, we randomly sample 10 points from the dataset and find the distance matrix between them in the full rank and reduced rank forms. Please refer to the IPYNB file submitted for the details of the selected points and distance matrices.

We need to find: $||\mathbf{D} - \hat{\mathbf{D}}||_F$. Based on these values, we get: $||\mathbf{D} - \hat{\mathbf{D}}||_F = 1.02$

2.2 Dataset-2: Olivetti Faces

2.2.1 Singular value decomposition and finding eigenfaces

For the face dataset $\mathbf{X} \in \mathbb{R}^{4096\times 400}$, we perform the singular value decomposition. We have: $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$. We also have the reduced rank representation: $\hat{\mathbf{X}} = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T$ such that $\mathbf{U} \in \mathbb{R}^{4096\times 400}$, $\Sigma \in \mathbb{R}^{4096\times 400}$, $\mathbf{V} \in \mathbb{R}^{400\times 400}$, $\Lambda_r \in \mathbb{R}^{5\times 5}$, $\mathbf{U}_r \in \mathbb{R}^{4096\times 5}$, $\mathbf{V}_r \in \mathbb{R}^{400\times 5}$

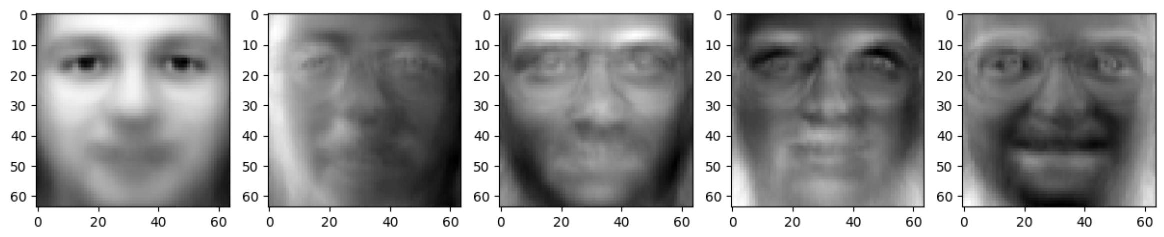
Fig. 12. Top-5 eigenvectors for the Olivetti faces found using SVD

2.2.2 Number of components for a given reconstruction error

To determine the number of components required for a given reconstruction error, we rely on the concept of energy preservation in Singular Value Decomposition (SVD). The total energy in a matrix is the sum of the squares of its singular values, σ_i^2. The cumulative energy ratio up to the k-th component is given by:

$$\text{Cumulative energy ratio} = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2} \tag{33}$$

We select the smallest k such that the cumulative energy ratio exceeds $1 - \text{error threshold}$, ensuring that the reconstruction error stays below the desired threshold.

Error Threshold (%)	0.1	1.0	5.0	10.0	20.0
Components Needed	219	34	1	1	1
Actual Error (%)	0.1	0.99	4.57	4.57	4.57

Table 2. Error Threshold vs. Components Needed

2.2.3 Low-rank approximated form for 2 images

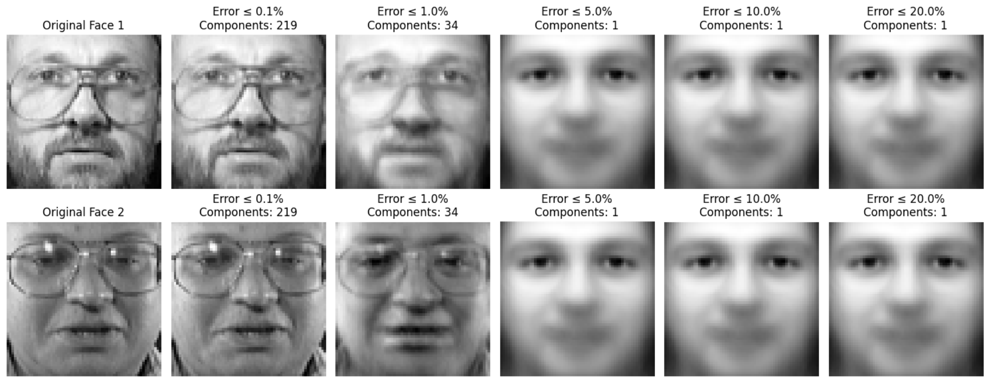

Fig. 13. Low-rank approximated form for two randomly selected images for different values of reconstruction error

Norm Comparision: Let $\mathbf{X} \in \mathbb{R}^{64\times64}$ be an image and $\hat{\mathbf{X}}_r \in \mathbb{R}^{64\times64}$ be its reconstructed form for r components. We report frobenius norms of $\mathbf{X}$ and $\hat{\mathbf{X}}_r$.

The lowering of the Frobenius norm with decrease in the number of components signifies the loss in information as we keep on dropping number of SVD components used in reconstruction. As we increase the number of components, the magnitude of the increase in norm keeps on

Original Norm	**219 Components**	**34 Components**	**1 Component**
734.1992	733.8446	730.5750	717.2214

Table 3. Norms for Different Component Counts

decreasing. This shows that the eigenvalues are of lower magnitude and lesser importance.

2.2.4 Nearest neighbour search

We use the following snippet to find the nearest neighbour of a given image sample in the dataset.

```python
def find_closest_neighbor(query, search_space):
    min_dist = float('inf')
    min_idx = -1
    for i in range(search_space.shape[1]):
        dist = euclidean(query, search_space[:, i])
        if dist < min_dist:
            min_dist = dist
            min_idx = i
    return min_idx, min_dist
```

We then compare the search time required for different numbers of components, i.e., for different values of reconstruction error. As expected, we can see a positive correlation between the number of reconstruction components and the time complexity.

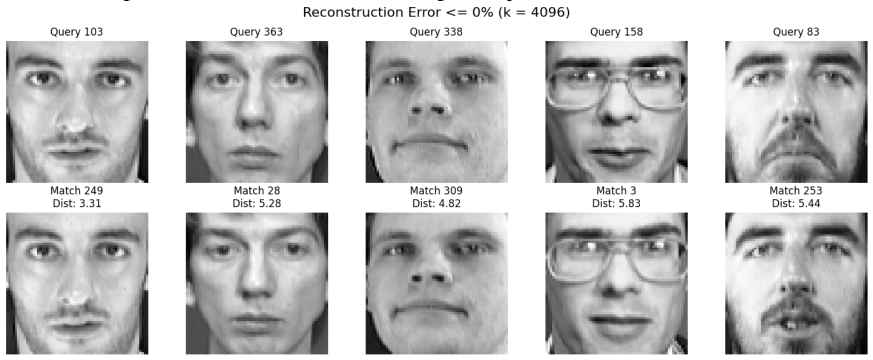

Fig. 14. Nearest neighbour search for 5 random faces in full rank representation

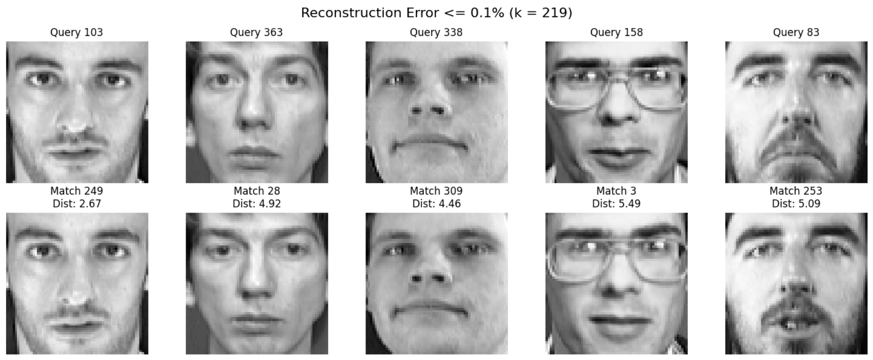

Fig. 15. Nearest neighbour search for 5 random faces for reconstruction error $\leq 0.1\%$

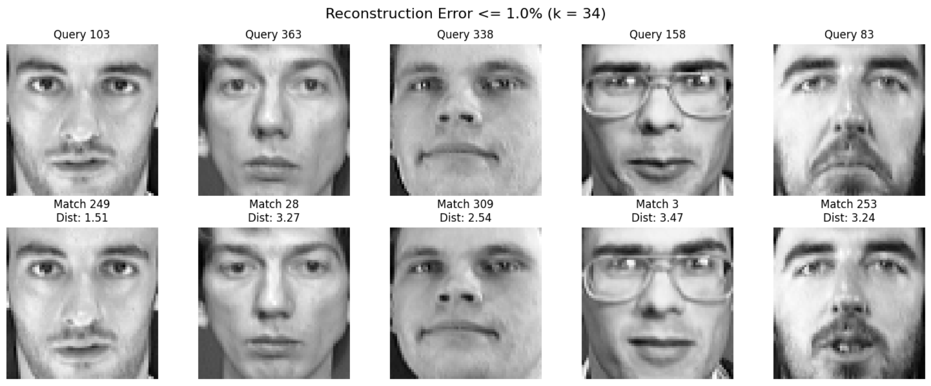

Fig. 16. Nearest neighbour search for 5 random faces for reconstruction error $\leq 1\%$

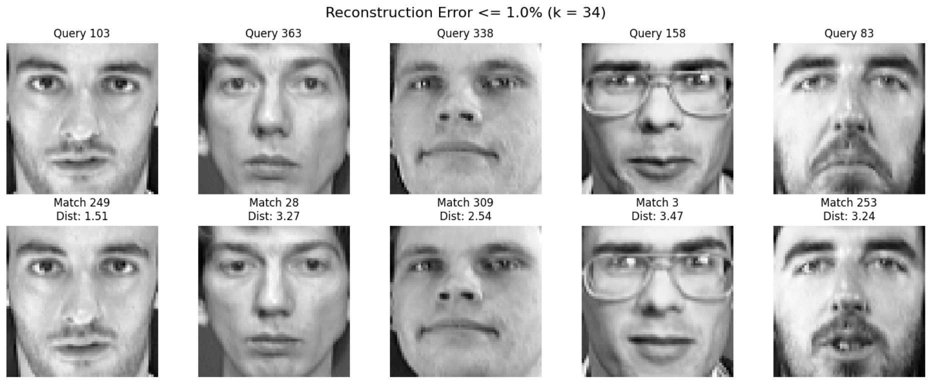

Fig. 17. Nearest neighbour search for 5 random faces for reconstruction error $\leq 5\%$

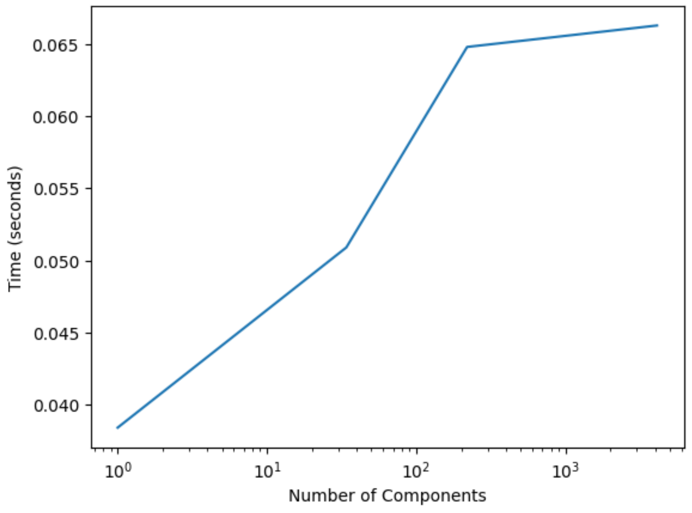

Fig. 18. Search time vs. number of reconstruction components in searching the nearest neighbour of an image in a given dataset. As expected, this graph shows a positive correlation between time taken for search and the number of components

2.2.5 Searching top-10 nearest neighbours

no. of components	full rank	219	34	1
time (seconds)	0.0513	0.0380	0.0300	0.0289

Table 4. Comparison of search times for different values of reconstruction error

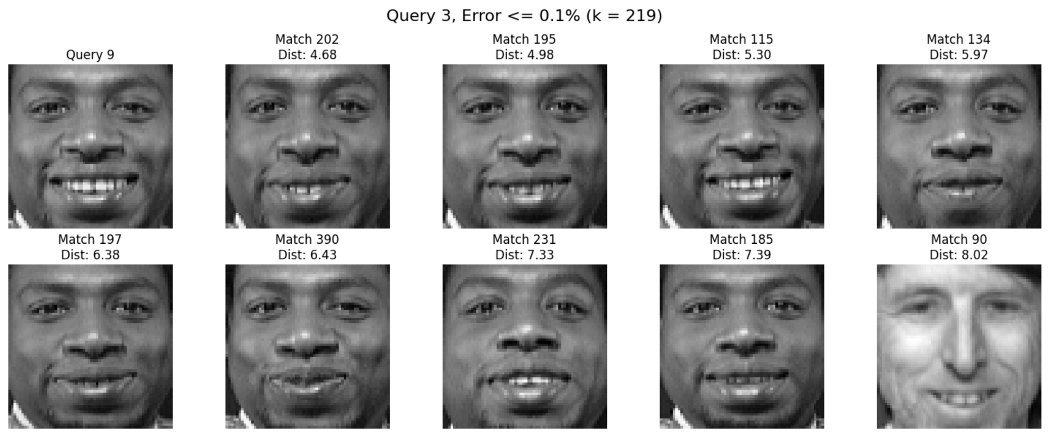

Fig. 19. 10-nearest neighbour search reconstruction error $\leq 0.1\%$

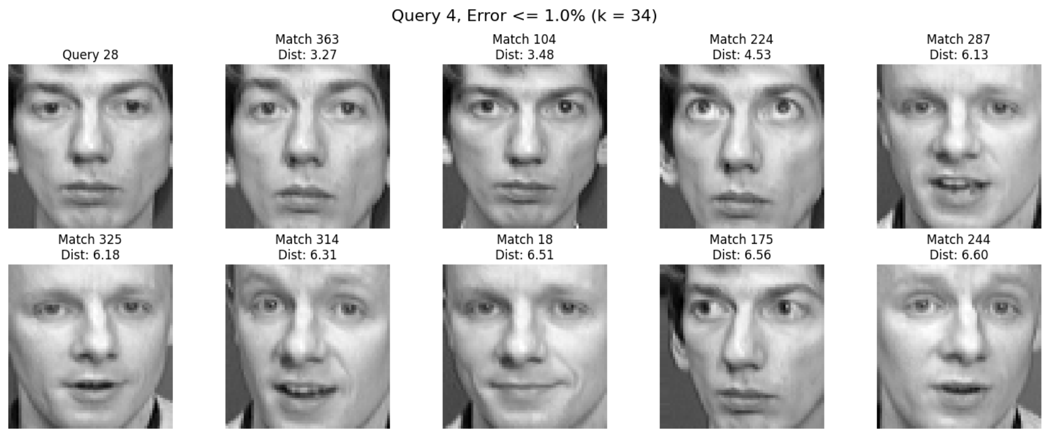

Fig. 20. 10-nearest neighbour search reconstruction error $\leq 1\%$

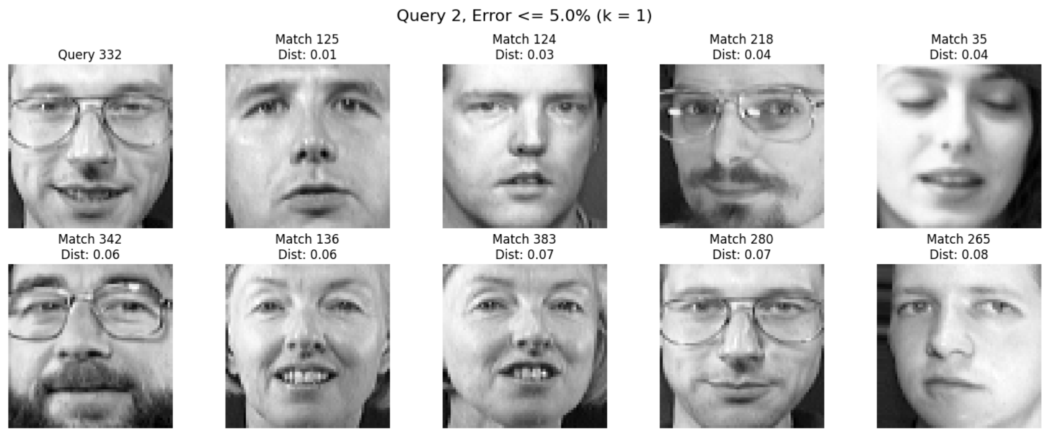

Fig. 21. 10-nearest neighbour search reconstruction error $\leq 5\%$