EE5026: Assignment CA2

Face Recognition

WU QILONG

E1124649@u.nus.edu

A0274903B

November 16, 2023

AY23/24 Semester 1

My Project Github Repo

Abstract

123[^1]

1 PCA

1.1 Introduction for PCA

Principal Component Analysis (PCA) is a widely used technique in machine learning and data analysis for dimensionality reduction and feature extraction. In the context of face recognition and image processing, PCA helps in reducing the dimensionality of the face dataset while retaining the most significant features.

PCA via Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) offers an efficient way to perform PCA, especially for large datasets like images. The basic idea is to decompose the data matrix into three matrices which can then be used to find the principal components.

Given a data matrix X of size $m \times n$ (where m is the number of images and n is the number of pixels in each image), SVD decomposes X into:

$$X = U\Sigma V^T \tag{1}$$

Where:

- U is an $m \times m$ orthogonal matrix.
- Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal.
- V^T is an $n \times n$ orthogonal matrix (transpose of V).

Steps in PCA through SVD

1. **Mean Normalization**: Subtract the mean face from each face in the dataset.

$$X_{norm} = X - \text{mean}(X) \tag{2}$$

2. **Compute SVD**: Perform SVD on the normalized data matrix X_{norm}.

$$X_{norm} = U\Sigma V^T \tag{3}$$

3. **Select Principal Components**: The columns of V (right singular vectors) are the principal components. For dimensionality reduction, select the first k columns corresponding to the largest k singular values in Σ.

$$V_k = V[:, 1:k] \tag{4}$$

4. **Project Data**: Project the original data onto the new feature space using V_k.

$$X_{proj} = X_{norm} \times V_k \tag{5}$$

Application in Face Recognition

In face recognition, PCA via SVD is used to extract the most significant features from face images. This not only reduces the computational burden but also enhances the performance by focusing on the features that contribute most to the variance in the data.

1.2 Experiment

In this experiment, the training dataset was constructed as follows: Initially, 7 images were randomly selected from the selfies dataset and included in the training set. Additionally, a proportional number of images were randomly sampled from each category of the PIE dataset and added to the training set. This combined dataset was used for training the PCA model, totaling 500 images.

Following the training, the PCA model was applied to the entire test dataset, which consisted of 1298 test images, for performance evaluation. This process involved projecting the test images onto the principal components of the PCA model to achieve dimensionality reduction and feature extraction. The experimental design and data processing flow aimed to validate the effectiveness of PCA in facial image processing.

Moreover, for the initial input data in this experiment, grayscale images with dimensions of 32 × 32 × 1 were used. These images were initially converted into 1D vectors with a size of 1024 through a stretching process. After preprocessing, Principal Component Analysis (PCA) was applied to these image vectors. The first three most significant components were selected to serve as the axes for the plot, and the image vectors were projected onto these axes, as depicted in **Figure 1**. It's worth noting that the PCA model was trained using only 500 images from the CMU PIE training set and subsequently tested on the entire set of 1298 test images.

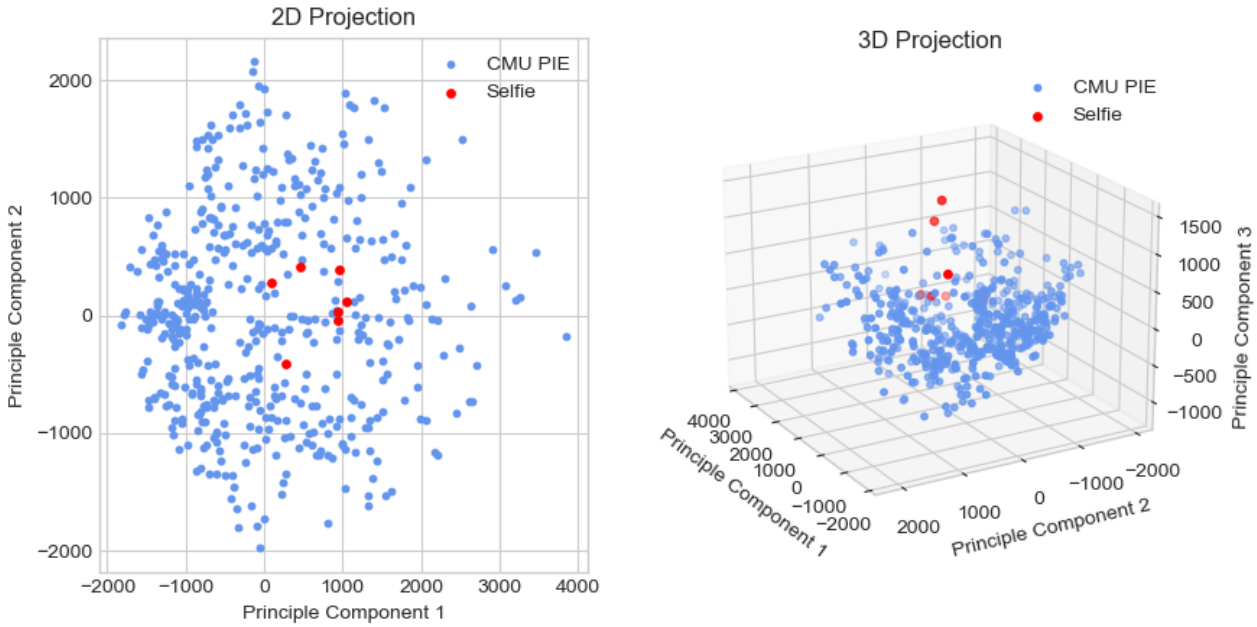

Figure 1: Data Projection in 2D and 3D via PCA

Simultaneously, in **Figure 2**, we can observe the computation and visualization of several key components:

1. The "Mean Face," represented as a virtual face image, is created by averaging the values of all the training image data.
2. Three "Eigen Faces" are also displayed, which correspond to the three most significant components obtained from the image vectors through PCA.

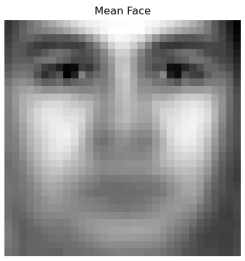

Figure 2: Mean and Eigen Faces

Moving forward, PCA was employed to reduce the dimensionality of the input images from 1024 to three different levels: 40, 80, and 200. These reduced-dimensional representations were then fed into a K-Nearest Neighbors (KNN) Classifier for the classification task. During this process, for each image vector that underwent PCA, their top 40, 80, or 200 components were respectively utilized in each classification experiment.

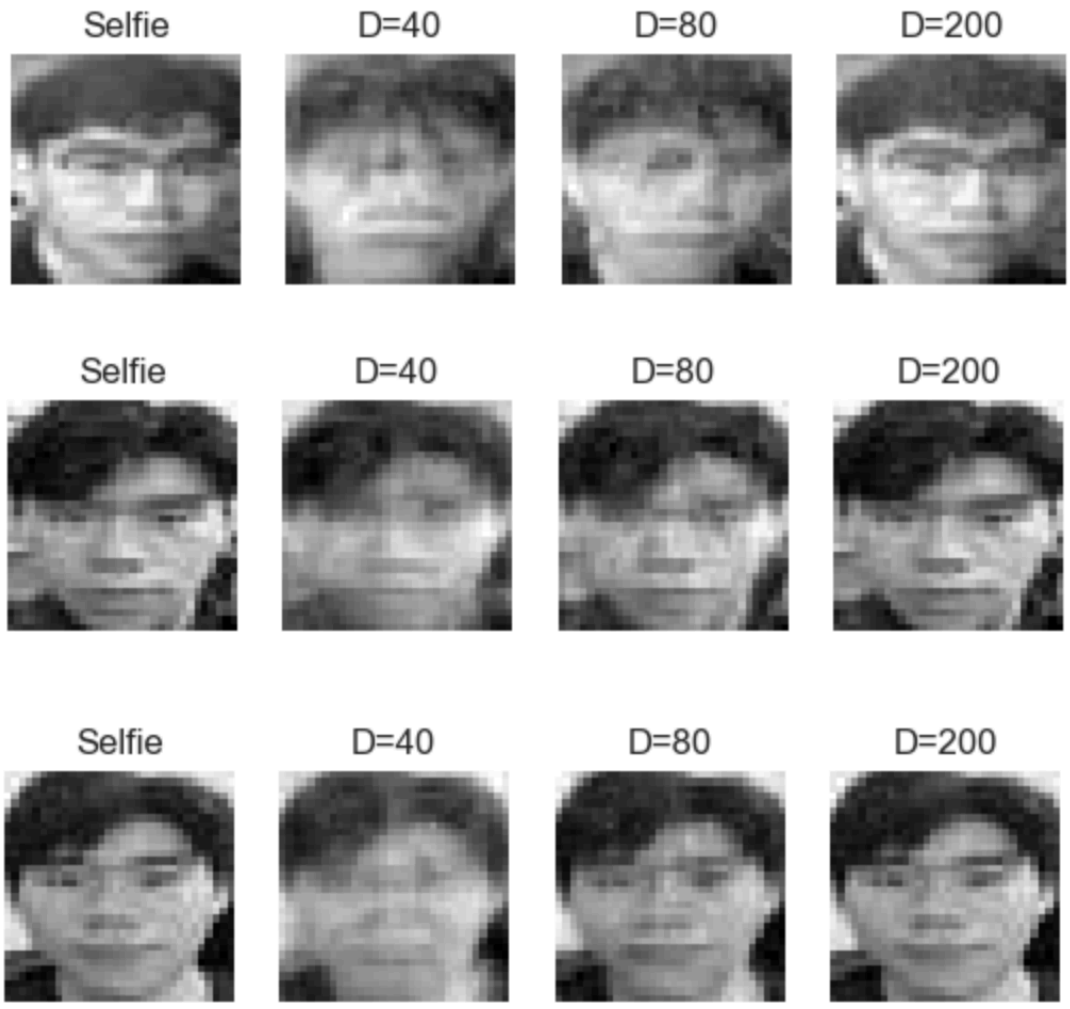

Figure 3: PCA Reconstructed Faces of My Selfies in Original, 40-D, 80-D, 200D, Space

Furthermore, an inverse projection was applied to these compressed image vectors, allowing the reconstruction of my selfies' faces, as demonstrated in **Figure 3**. To assess performance, the error rates of KNN classification were recorded for each batch of PCA dimensionality-reduced data and visualized in Figure 4. Additionally, a baseline KNN classification test was conducted on the original image vectors with a size of 1024, without any PCA processing. The results of this baseline test are also included in **Figure 4** for

comparison.

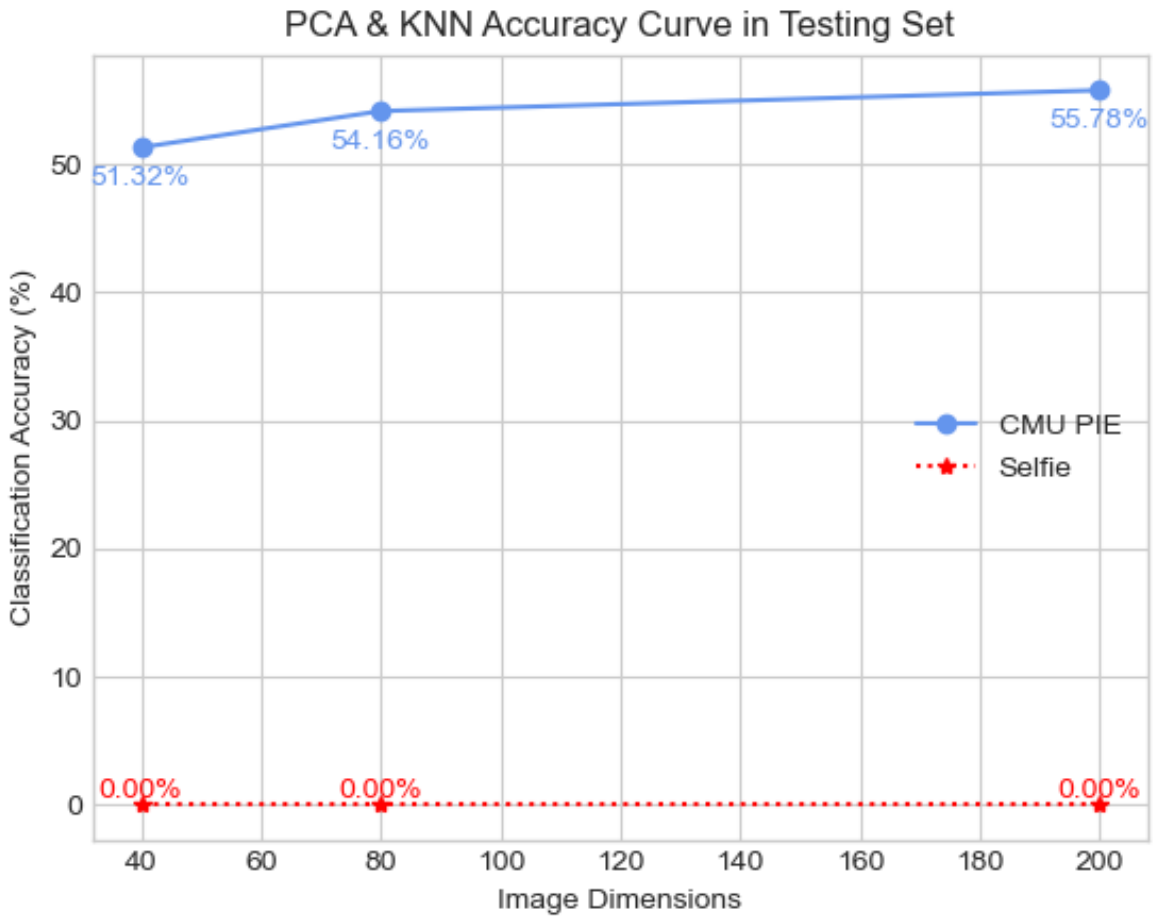

Figure 4: PCA and KNN Classification Accuracy Curve in Testing Set

1.3 Discussion

Figure 1 provides valuable insights into the data distribution and separation:

- In the 2D plot on the left side of **Figure 1**, the red data points representing the selfie photos blend seamlessly with the CMU PIE dataset. This blending makes it challenging to establish a distinct boundary between the two groups, thereby confirming that the meticulously curated selfie photos dataset closely resembles the original CMU PIE dataset. From the viewpoint of the first two principal components, the distinction between the two groups appears minimal.
- However, when examining the 3D plot in **Figure 1**, an interesting observation emerges. On the third principal component, the two groups of data start to diverge. Specifically, the selfie photos exhibit higher values along the z-axis. This observation suggests that, from the perspective of the K-Nearest Neighbors (KNN) algorithm, all data points belonging to the selfie class are closely clustered together, making it more convenient for the KNN algorithm to make accurate classifications.

Figure 2 depicts: the four faces displayed—comprising one mean face and three eigen faces—represent significant features extracted from the entire dataset. These faces can be regarded as statistical summaries of all the faces within the dataset, each accentuating a distinct essential aspect.

- For instance, the most crucial eigen face bears a striking resemblance to the mean face. Both of these faces capture essential common features found on a human face, including eyes, noses, mouths, outlines, and more. Meanwhile, the second and third eigen faces prioritize highlighting variations in view perspective and angles of illumination, which are pivotal features within the CMU PIE dataset.
- This underscores the remarkable capability of PCA in extracting the most critical information and features from a vast dataset with high dimensions.

Figure 3 illustrates: a compelling comparison between the original face image and the reconstructed face images.

- The visual analysis reveals a clear trend: as an increasing number of eigen faces are superimposed onto the reconstructed image, the face gradually becomes more distinguishable, showcasing finer details and approaching the appearance of the original image.
- Even when using a considerably lower number of dimensions compared to the original image (which had 1024 dimensions), the reconstructed face images demonstrate their capability to recapture most of the original image's details, rendering the face recognizable to human observers. This outcome serves as compelling evidence that PCA can be an outstanding choice for reducing data dimensionality. It achieves this while preserving the most critical features within the data and effectively reducing the data's overall dimensionality.

Figure 4 depicts: the K-Nearest Neighbors (KNN) classifier consistently achieved an accuracy of 0.0 for the selfie class throughout the entirety of the experiment, regardless of the number of dimensions considered. This remarkable performance suggests that the selfie class might be exceptionally straightforward for the KNN algorithm. This outcome was quite predictable given the small total number of training and testing photos for this class, which amounted to just 10 images. Moreover, as observed in **Figure 1**, the PCA results had already demonstrated that the selfie photos could be distinctly separated from the other classes.

- On the CMU PIE dataset, the accuracy curve for the KNN classifier exhibited a noticeable decline, starting at 51.32% and gradually increasing to around 55.78% by the end of the experiment. This trend aligns with expectations: as the number of data components increased, the classification error decreased, steadily approaching the baseline error rate obtained from the original images. It's important to note that the top principal components played a more significant role in improving classification accuracy compared to the principal components (or eigen faces) at the tail end of the spectrum. These top components contained more crucial information about the facial images.
- In practical applications, a similar curve plot could serve as a reference to strike a balance between the number of data dimensions and the desired level of accuracy. However, it's worth acknowledging that the results were not entirely satisfactory, partly due to the limitation of having a relatively small training set (comprising only 500 randomly selected images) used to train the PCA model. This limited training data may have hindered the model's ability to generalize well when applied to the larger test set, which included 1303 images.

2 LDA

2.1 Introduction for LDA

Linear Discriminant Analysis (LDA) is a valuable technique for dimensionality reduction, particularly when applied to facial recognition tasks. LDA is a supervised learning method that aims to maximize the separation between classes while minimizing the variation within each class. This makes it particularly effective for tasks where class information is crucial, such as facial recognition.

Mathematical Principles of LDA

Given a dataset X consisting of m samples, each belonging to a class C_i, LDA seeks to find a projection matrix W to map the high-dimensional data into a lower-dimensional subspace. This projection process can be broken down into the following steps:

1. Compute the within-class mean vectors (μ_i), representing the mean of each class:

$$\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x \tag{6}$$

   Where n_i is the number of samples in class C_i.

2. Compute the overall mean vector (μ), representing the global mean:

$$\mu = \frac{1}{m} \sum_{i=1}^{m} \mu_i \tag{7}$$

3. Calculate two scatter matrices:
   - Within-Class Scatter Matrix (S_W):

$$S_W = \sum_{i=1}^{m} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \tag{8}$$

   - Between-Class Scatter Matrix (S_B):

$$S_B = \sum_{i=1}^{m} n_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{9}$$

4. Solve the generalized eigenvalue problem:

$$S_B W = \lambda S_W W \tag{10}$$

   Where λ represents the generalized eigenvalues, and W contains the corresponding generalized eigenvectors.

5. Select the top k eigenvectors corresponding to the largest eigenvalues and construct the projection matrix W.

6. Project the original dataset X into the lower-dimensional subspace:

$$X_{lda} = XW \tag{11}$$

Application of LDA in Facial Recognition

In facial recognition, LDA plays a critical role in extracting discriminative features, leading to enhanced recognition performance. By selecting an appropriate lower-dimensional subspace, LDA projects facial data into a space where different faces become more distinguishable, thus improving classifier performance. Additionally, LDA can be applied to feature fusion, face verification, and other facial recognition-related tasks.

LDA is a valuable tool in facial recognition due to its ability to maximize class separability while reducing dimensionality.

2.2 Experiment

In this experiment round, the input data underwent a consistent vectorization process, which was explained in the previous section, before being input into the LDA model. Similarly, just as before, the first three most significant axes were chosen as the three axes for the plot. The image vectors were then projected onto these axes, as depicted in **Figure 5**.

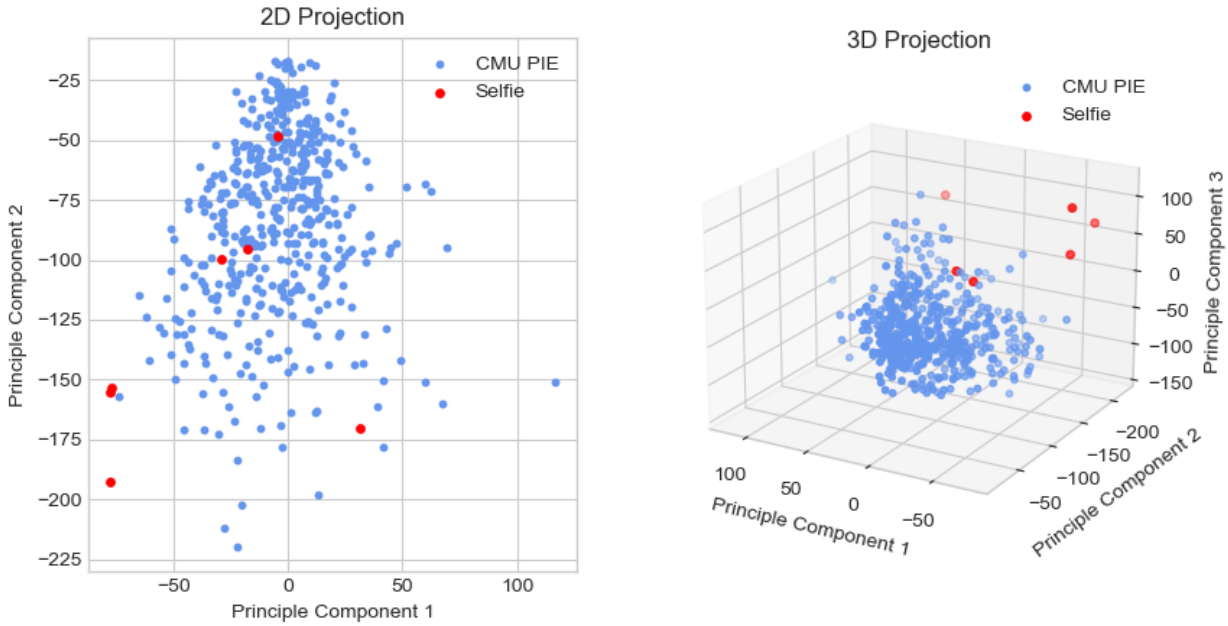

Figure 5: Data Projection in 2D and 3D via LDA

Next, dimentionality reduction was performed on the data using LDA. The input image vectors with 1024 dimensions were reduced to to 2, 3 and 9 respectively, and feed into the same K-Nearest Neighbours Classifier as in the previous section to perform classification task. The KNN classifier was first fit with the dimentionality-reduced training set (2978 images), then tested on the test set (1303 images). Results were collected and plotted in **Figure 6**.

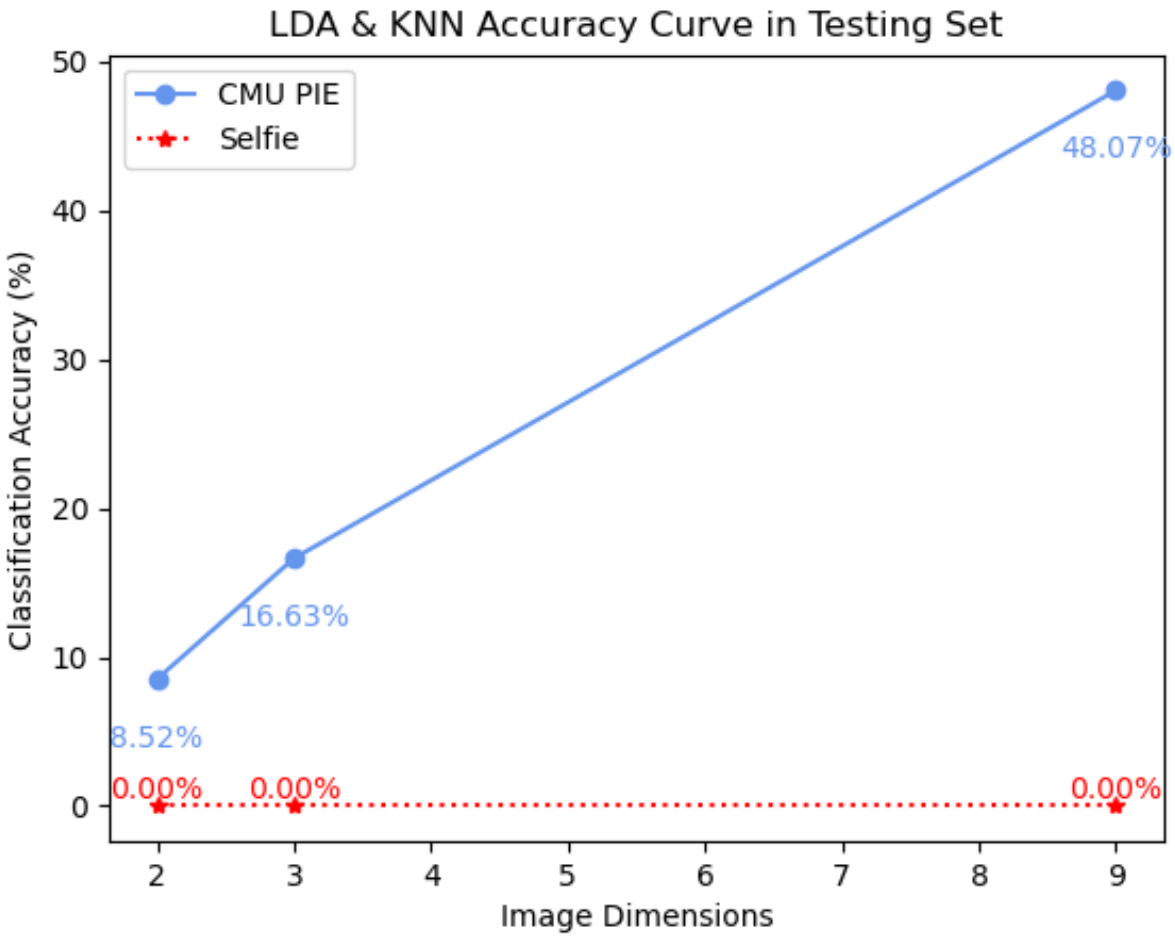

Figure 6: LDA and KNN Classification Accuracy Curve in Testing Set

2.3 Discussion

In **Figure 5**, it is evident that there is a significant separation between different data groups on the plot, particularly in the direction of axis 2. Similar to the previous section's plot, the selfie data points are closely clustered together, even closer than in the previous method. Other classes from the CMU PIE dataset also tend to form distinct clusters on the plot.

In **Figure 6**, the performance of the K-Nearest Neighbors (KNN) classifier on the selfie class appears to be subpar, with an initial accuracy of 0.00%, which then keeps the same. However, the line representing the classification accuracy for the PIE dataset demonstrates good performance. It shows that the accuracy rapidly increases as the number of data dimensions increases.

It's worth noting that the classification accuracy for the CMU PIE dataset at D = 2 and D = 3 in this experiment were 8.52% and 16.63%, respectively, which were relatively inaccurate. However, LDA managed to significantly reduce this error rate to 48.07% when the dimensionality reached D = 9, underperforming the previous PCA-KNN method that used a higher number of dimensions, but the speed of that relatively increasing faster. This result suggests that in terms of classification tasks, LDA may achieve higher accuracy with fewer dimensions compared to PCA.

3 GMM

3.1 Introduction for GMM

Gaussian Mixture Models (GMM) are a powerful statistical technique widely used in the field of face recognition and image processing. GMM is a probabilistic model that represents a complex distribution of data using a combination of simple Gaussian distributions. It has gained popularity for its effectiveness in modeling the underlying structure of high-dimensional data, making it particularly valuable in facial recognition tasks.

GMMs are versatile and adaptable, capable of capturing intricate patterns and variations within face images. They offer a flexible framework for modeling the statistical characteristics of face data, making them well-suited for applications such as face verification, facial expression analysis, and even face generation.

In face recognition, GMMs can be employed to model the statistical distribution of facial features, allowing for robust and accurate identification and verification of individuals. By capturing the inherent variability in facial images and encoding it in the model, GMMs contribute to the development of robust face recognition systems that can handle variations in pose, lighting conditions, and facial expressions.

This introduction provides an overview of the role of Gaussian Mixture Models in face recognition, emphasizing their adaptability and effectiveness in modeling complex data distributions.

3.2 Experiment

Gaussian Mixture Models (GMM) is a clustering algorithm that employs a probabilistic approach to group data points. In the context of GMM, it is assumed that a known, finite number of clusters exists within the dataset, and these clusters are distributed following Gaussian distributions. During the training process, GMM learns the parameters that define these Gaussian distributions and computes the probability of a data point belonging to a particular cluster.

Before applying the GMM clustering algorithm, all input data undergo the same vectorization process, followed by a PCA (Principal Component Analysis) dimensionality reduction pre-processing step. This pre-processing step results in image vectors with dimensionality reduced to either 80 or 200, depending on the chosen configuration. Subsequently, a GMM model with three Gaussian components is trained using the provided training set data.

Figure 7 and **Figure 8** visually represent the results of this clustering process. These figures display sample images that have been randomly selected from each cluster, demonstrating the outcomes for dimensionality settings of D=80 and D=200 in the training datasets, respectively. And the results are quite similar.

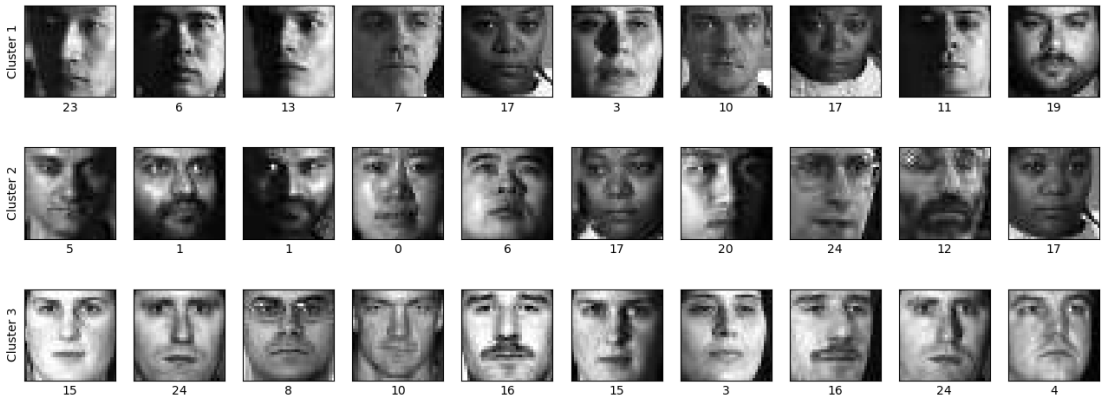

Figure 7: Gaussian Mixed Model Clustering Results in 80-D Space

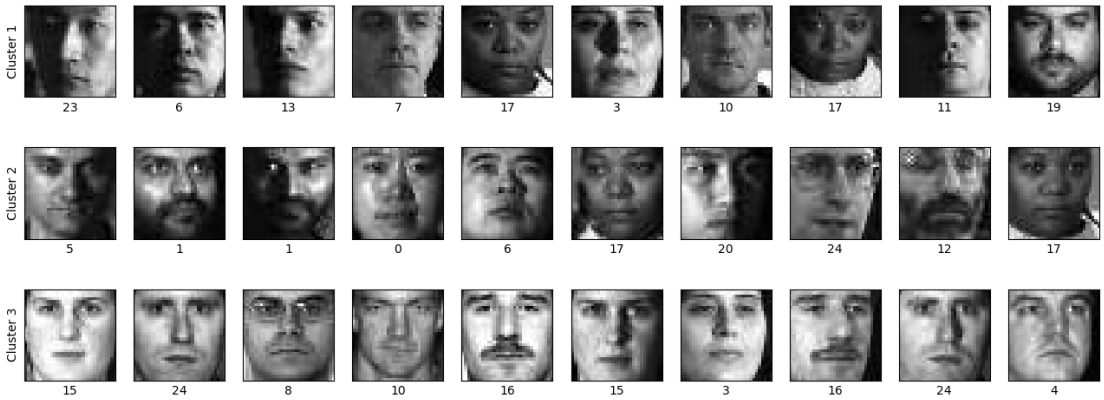

Figure 8: Gaussian Mixed Model Clustering Results in 200-D Space

3.3 Discussion

In **Figure 7** and Figure 8, each image is accompanied by a numerical class label assigned to it. Analyzing the results, it becomes evident that most of the images sharing the same class labels tend to be grouped into the same cluster. For instance, in **Figure 7**, the two images labeled as "1" were clustered together into "Cluster 2"

However, there are instances where images from the same class end up in different clusters. This occurs because the experiment employed only three clusters. Despite the inaccuracies and ambiguity in how these faces were grouped, certain patterns emerge from the images. Notably, there are varying degrees of similarity in terms of facial pose angles and illumination angles within each cluster.

In general, Gaussian Mixture Models (GMM) can be an excellent choice for clustering if the data distribution follows or approximates a Gaussian distribution. GMM can also serve as a valuable data preprocessing technique. However, one drawback to consider is that the number of clusters needs to be determined prior to applying the method, which can be a challenge in certain situations.

4 SVM

4.1 Introduction for SVM

Support Vector Machines, often referred to as SVM, are a powerful and widely-used class of machine learning algorithms with a diverse range of applications, including face recognition. SVM is particularly well suited for solving classification problems where the goal is to distinguish between different categories or classes of data, making it a valuable tool in the field of facial recognition.

At its core, SVM seeks to find an optimal hyperplane that best separates data points from different classes while maximizing the margin between these classes. This hyperplane is chosen in such a way that it minimizes the classification error and provides a clear separation between the data points. In the context of face recognition, SVMs are utilized to classify face images into distinct categories, such as identifying individuals based on their facial features.

SVM's ability to handle high-dimensional data and its effectiveness in handling complex decision boundaries make it a popular choice for various facial recognition tasks. This introductory overview sets the stage for understanding how SVM can be applied to the challenging and important task of recognizing faces in images.

4.2 Experiment

SVM accuracy 80-D (%)

C\gamma	0.1	0.01	0.001
0.01	3.99	12.27	40.44
0.1	3.99	12.27	36.76
1.0	3.99	11.89	21.02

SVM 200-D accuracy (%)

C\gamma	0.1	0.01	0.001
0.01	3.99	10.20	43.13
0.1	3.99	10.20	43.28
1.0	3.99	10.20	22.94

Figure 9: SVM classification accuracy Results in 80-D and 200-D Space

Similar to the previous experiment, all the input data went through the same vectorization process and a PCA dimensionality reduction pre-processing technique, forming image vectors with dimensionality of 80 and 200 respectively. Different SVMs with a penalty parameter C value of 0.01, 0.1, and 1.0 along with 0.1, 0.01, 0.001gamma value of were tested in this experiment on the two pre-processed datasets with different dimensions. The classification error rates recorded were showed in **Figure 9**.

4.3 Discussion

The results presented in **Figure 9** indicate a comparative analysis of SVM classification accuracy in 80-dimensional (80-D) and 200-dimensional (200-D) spaces after applying PCA for dimensionality reduction. The experiment involved testing SVMs with different penalty parameter C values of 0.01, 0.1, and 1.0, and gamma values of 0.1, 0.01, and 0.001.

In the 80-D space, the lowest error rates are observed with the highest gamma value (0.001), suggesting that the model may perform better with a more complex decision boundary when the dimensionality is reduced. However, as the gamma value decreases (indicating a simpler decision boundary), the error rates increase significantly, which might imply over-simplification of the model for the given data complexity.

Conversely, in the 200-D space, while the error rates are generally lower than in the 80-D space, indicating that the additional dimensions may provide more information for the classification task, the trend with respect to gamma values is less pronounced. The error rates are relatively stable across different values of C for a given gamma, suggesting that the penalty for misclassification does not significantly affect the model's performance in higher-dimensional spaces.

It is noteworthy that the classification accuracy does not improve significantly with increased C values in either dimensional space, which could imply that the chosen values for C are already sufficient or that the model is not very sensitive to this parameter within the tested range.

Overall, these results could suggest that while PCA is effective in reducing dimensionality with a minimal loss of information, the choice of hyperparameters for the SVM (particularly the gamma value) is crucial and may need to be carefully tuned depending on the dimensionality of the data. Furthermore, the lack of substantial differences across various C values might indicate that the data is not heavily overlapped, and the margin size (controlled by C) is not a critical factor in the classification accuracy for this specific dataset.

5 CNN & ResNet

5.1 Introduction for CNN

Convolutional Neural Networks, commonly referred to as CNNs, are a class of deep learning models that have revolutionized the field of computer vision and have found widespread applications in various domains. CNNs are designed to automatically and adaptively learn patterns, features, and hierarchical representations from raw data, making them particularly well-suited for tasks such as image recognition, object detection, and image classification.

The fundamental idea behind CNNs is inspired by the human visual system, which processes visual information hierarchically, starting from simple features like edges and shapes and gradually building up to more complex concepts. CNNs mimic this process by employing layers of convolutional and pooling operations to extract features from images. These networks have the remarkable ability to automatically detect and learn intricate patterns, textures, and structures in data, allowing them to excel at tasks that involve visual perception.

CNNs have been instrumental in achieving state-of-the-art results in various computer vision challenges, including image classification competitions like ImageNet. Their versatility has also led to applications in fields beyond computer vision, such as natural language processing, autonomous vehicles, and medical image analysis.

5.2 Introduction for ResNet-18

ResNet-18 is a deep convolutional neural network architecture that has played a pivotal role in the advancement of deep learning, particularly in the domain of computer vision. It is a member of the ResNet (Residual Network) family, which was introduced to address the challenges of training extremely deep neural networks.

Developed by Microsoft Research, ResNet-18 is renowned for its depth while maintaining efficient training and excellent performance. The "18" in its name signifies the total number of weight layers, making it a relatively shallow variant of the ResNet family, which includes deeper models like ResNet-50 and ResNet-101.

At the core of ResNet-18 is the concept of residual learning, where shortcuts or "skip connections" are added to the network. These skip connections enable the network to learn residual functions, making it easier to train extremely deep networks. This architectural innovation alleviates the vanishing gradient problem, allowing for the successful training of networks with hundreds of layers.

ResNet-18 has demonstrated remarkable capabilities in a wide range of computer vision tasks, including image classification, object detection, and image segmentation. Its architecture has been adopted as a backbone in many state-of-the-art models for these tasks.

5.3 Experiment & Results

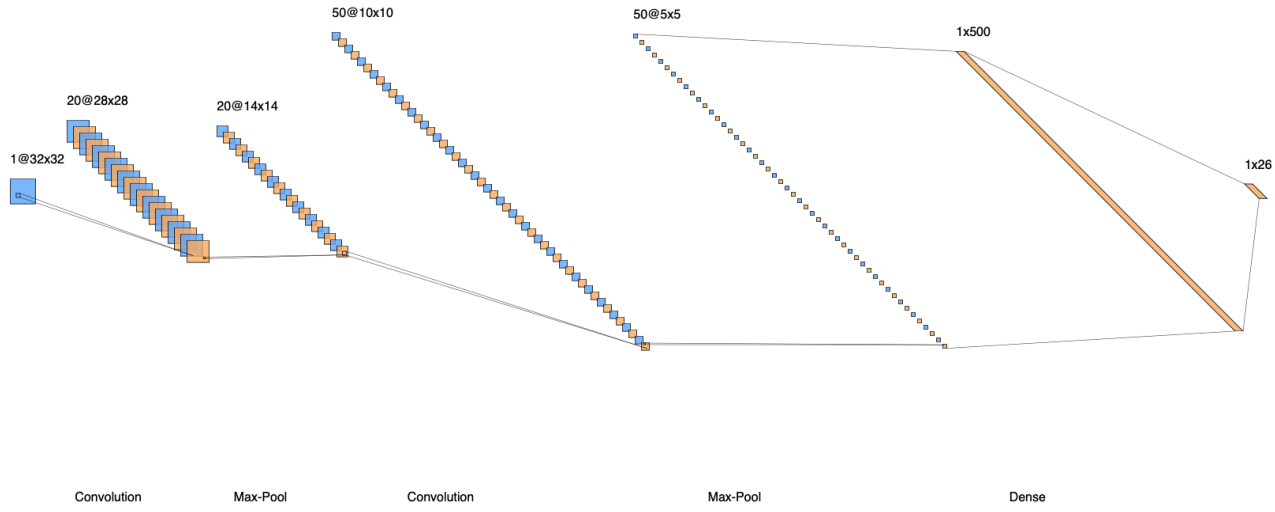

Figure 10: CNN Structure

In this particular experiment, there was a departure from the previous ones in terms of dataset composition. Instead of using 25 CMU PIE classes along with 1 selfie photo class, only 20 CMU PIE classes were utilized, along with a single class for selfie photos.

Prior to feeding the input images into the model, several preprocessing steps were applied. Firstly, all images were converted to grayscale. Subsequently, normalization was performed, and a data augmentation technique was applied, involving random horizontal flipping of the images.

For this experiment, a relatively simple custom-designed Convolutional Neural Network (CNN) model was employed. You can find its architectural structure depicted in **Figure 10**. Following training for 10 epochs, with a batch size of 32 and a learning rate of 1e-3, the classification accuracy for this straightforward CNN and ResNet-18 model was determined to be 97.7% and 98.0% respectively.

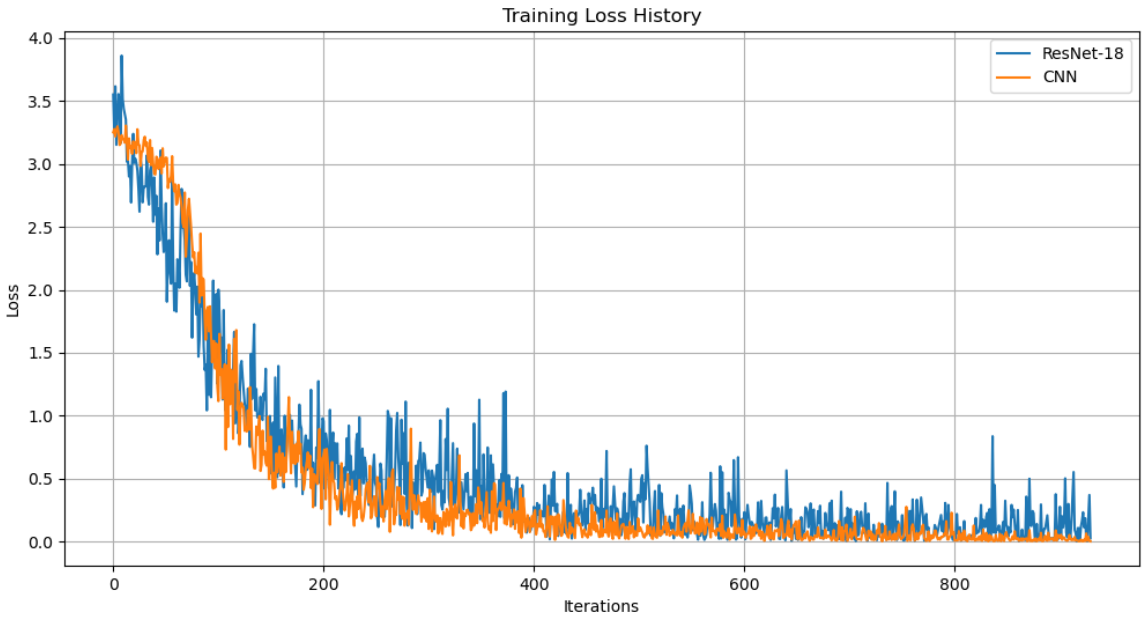

Figure 11: Training loss for CNN and ResNet18