22AIE213 – MACHINE LEARNING

Face recognition Using PCA and K-Means

Submitted by

BATCH-'B' GROUP-1

M. NILAVARASAN (CB.EN.U4AIE22139)

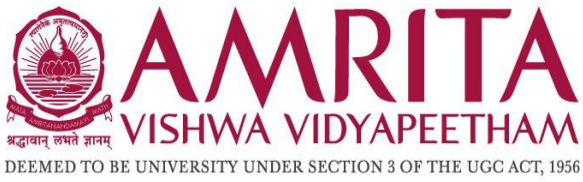
M SANTHOSH (CB.EN.U4AIE22152)

GOWTHAM NAIDU (CB.EN.U4AIE216)

YASHWANT (CB.EN.U4AIE22131)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
CSE(AI)



Centre for Computational Engineering and Networking

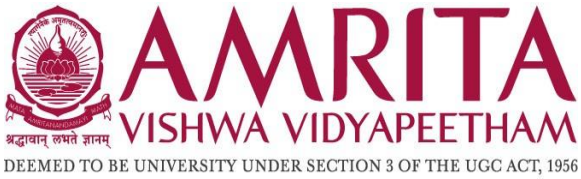
AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112 (INDIA)

JUNE- 2024

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE - 641 112



BONAFIDE CERTIFICATE

This is to certify that the thesis entitled "Face recognition Using PCA and K Means" submitted by M.NILAVARASAN(CB.EN.U4AIE22139), MSANTHOSH(CB.EN.U4AIE22152), GOWTHAM NAIDU (CB.EN.U4AIE22116), YASHWANT(CB.EN.U4AIE22131) for the award of the Degree of Bachelor of Technology in the "CSE(AI) " is a bonafide record of the work carried out by her under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

Dr. Abhishek

Project Guide

Dr. K.P.Soman

Professor and Head CEN

Submitted for the university examination held on 3/6/2024.

Acknowledgement

We would like to express our special thanks of gratitude to our teacher (Dr. Abhishek sir), who gave us the golden opportunity to do this wonderful project on the topic ("Face recognition Using PCA and K-Means"), which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given.

We would also like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

DECLARATION

I, M.NILAVARASAM (CB.EN.U4AIE22139), M SANTHOSH(CB.EN.U4AIE22152), GOWTHAM NAIDU (CB.EN.U4AIE22116), YASHWANT(CB.EN.U4AIE22131), hereby declare that this thesis entitled "Face recognition Using PCA and K-Means ", is the record of the original work done by me under the guidance of Dr. Abhishek, Assistant Professor, Centre for Computational Engineering and Networking, Amrita School of Artificial Intelligence, Coimbatore. To the best of my knowledge, this work has not formed the basis for the award of any degree/diploma/ associate ship/fellowship/or a similar award to any candidate in any University.

M.NILAVARASAM **M SANTHOSH**

GOWTHAM NAIDU **YASHWANT**

Place: Coimbatore **Signature of the Student**
Date: 3-06-2024

Table Of Contents

Table Of Figures.

Chapter-1 Principal Component Analysis Using SVD

1.1. Aim

The aim of this project is to train a model using face images from multiple individuals, enabling it to accurately identify and match faces during testing.

1.2.Abstract

The main goal of this project is to implement a face recognition algorithm in Python, leveraging machine learning techniques such as Principal Component Analysis (PCA) and k-means clustering. We utilize the ORL dataset, which is readily available for face recognition tasks and Also Created Own-Dataset by collecting images from students. The initial step involves converting the image data into a suitable format for processing.

Following this, we apply Principal Component Analysis as a preprocessing step to reduce the dimensionality of our dataset. PCA helps to simplify the dataset by transforming it into a set of orthogonal (uncorrelated) variables called principal components, which capture the most significant features of the data while reducing noise and redundancy. This dimensionality reduction makes the dataset more manageable and enhances the efficiency of subsequent processing steps.

After reducing the dimensions, we use the transformed data to compare the mean of each class with the test data. The classification is performed by identifying the class whose mean is closest to the test data, based on a defined distance metric. This approach allows us to classify the test data accurately into the corresponding class. By following this methodology, we successfully achieve our objective of face recognition.

1.3.Introduction

Face recognition has become a very common system in the modern world. The data processed for image recognition is very high. Through Principal Component Analysis we can reduce the dimensionality while retaining the utmost detailing Compression is achieved while still maintaining the significant patterns and trends

This is achieved with the mathematical concepts of Eigen decomposition and Singular Value Decomposition(SVD). We analyse this concepts while trying to incorporate the same in different applications.

1.3.1 How PCA Implemented using SVD

PCA using SVD decomposes a data matrix into singular value, left singular vector, and right singular vector matrices. Principal components are derived from the right singular vectors, capturing maximum variance for dimensionality reduction.

2.1. Methodology

In this section we will see how we have developed the methodology in implementing our algo, we will tell the step-by-step idea of implementing(starting from the extracting-data till Detecting the face of a person. We will tell the theoretical view of the methodology.

2.1.1. Data Extraction

For this project, we utilize the ORL dataset, a well-known source of face images for recognition tasks, supplemented by our own generated data to enhance the diversity and robustness of the dataset. The images are first converted into a matrix form suitable for processing, using the image_to_matrix_class. This conversion is crucial as it transforms the visual information into a numerical format that can be effectively analysed and manipulated by machine learning algorithms.

2.1.2. Feature

In our approach to face recognition, each pixel of the grayscale image is considered a feature. Grayscale images represent pixel intensity values ranging from black to white, where each pixel's value denotes the brightness level at that specific location. By treating each pixel as a feature, we capture the grayscale variations across the entire image. This pixel-level granularity enables our model to discern intricate facial details and patterns, facilitating accurate recognition results. Leveraging grayscale images simplifies the computational complexity while retaining essential facial information, making our recognition system efficient and effective.

2.1.3. Preprocessing

In the preprocessing stage of our face recognition project, we employ two main techniques: cantering the data around the mean and applying Principal Component Analysis (PCA) using the Singular Value Decomposition (SVD) approach to reduce the dimensionality of the data.

2.1.4. Centering The Data

Centering the data involves subtracting the mean of each feature (pixel) across all samples. This process ensures that the data is cantered at the origin, which is a crucial step for PCA. By centering the data, we eliminate any biases or offsets, allowing PCA to capture the most significant variations in the data.

2.1.5. Applying PCA via SVD

PCA is a powerful technique used for dimensionality reduction by transforming the data into a lower-dimensional space while preserving its essential characteristics. In our implementation, we use the Singular Value Decomposition (SVD) approach to compute the principal components of the data.

2.1.6. Percentage Reduction

The percentage reduction in dimensionality needed for PCA depends on the desired balance between computational efficiency and information retention. Typically, we aim to retain a significant percentage of the variance in the data while reducing its dimensionality. A common approach is to select the number of principal components that collectively explain a certain percentage of the total variance in the dataset. For example, if we choose to retain 95% of the variance, we compute the cumulative explained variance ratio for each principal component and select the number of components that achieve this threshold.

2.1.7. Recasting Everything

Once we have determined the desired percentage reduction and selected the appropriate number of principal components, we recast the data using these components. This involves projecting the original data onto the lower-dimensional subspace spanned by the selected principal components. The resulting transformed data retains the essential information while reducing its dimensionality, making it more manageable and suitable for subsequent processing steps, such as classification.

In summary, preprocessing involves cantering the data around the mean and applying PCA via the SVD approach to reduce the dimensionality of the data. The percentage reduction needed for PCA depends on the desired balance between computational efficiency and information retention, and once determined, we recast the data using the selected principal components to obtain a lower-dimensional representation suitable for further analysis.

2.1.8. Classifying

In our face recognition project, the classification stage involves assigning a label (or class) to a given test image based on its similarity to the labelled training data. This process is supervised, meaning that the algorithm learns from labelled training data to make predictions on unseen test data.

Once the data is dimensionally reduced, we apply a classification algorithm to assign labels to the transformed data. In our case, as each dimension corresponds to a principal component, each sample in the dataset is represented by a set of principal component scores.

To classify a test image, we compare its transformed representation (the principal component scores) with the mean of each class in the training dataset. We use a distance metric, such as Euclidean distance, to measure the similarity between the test image and each class mean. The class with the smallest distance to the test image is considered the closest match The test image is assigned the label of the class with the minimum distance. This means that the test image is classified as belonging to the class whose mean is closest to its transformed representation in the reduced-dimensional space.

This is the key-steps that we planned for implementing the project, with this idea that we have formulated. It is now easy, just convert the above logic/idea to code. And this is what we have also done, we have converted the above idea to code. Below Section we will explain the code's that we have done for this methadology

2.2. Code's In the Project

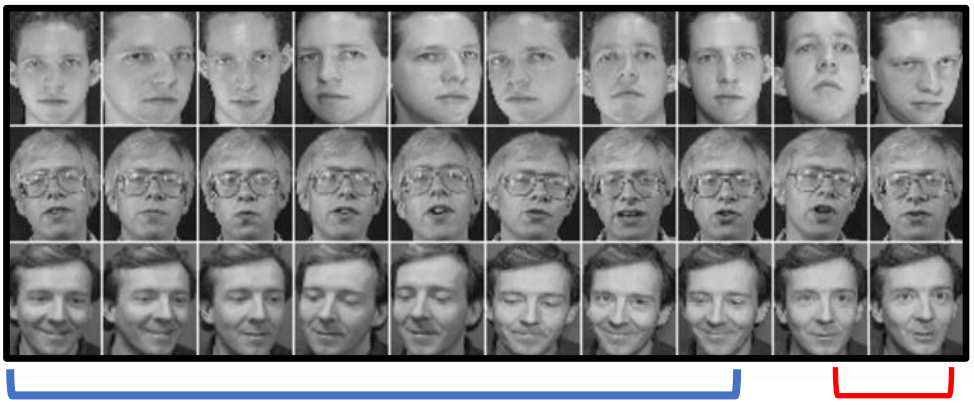
In this section we will see how we have developed the code for the above methadology in implementing our algo, we will explain the python-class's, and the role of the class's that we have implemneted in this project. And a outline how all class's work together in this section.

2.2.1. class's created

There are few class's are constructed, and are named below.

- Dataset Class
- Images to matrix
- Main
- Algo

2.2.1.1. Dataset class



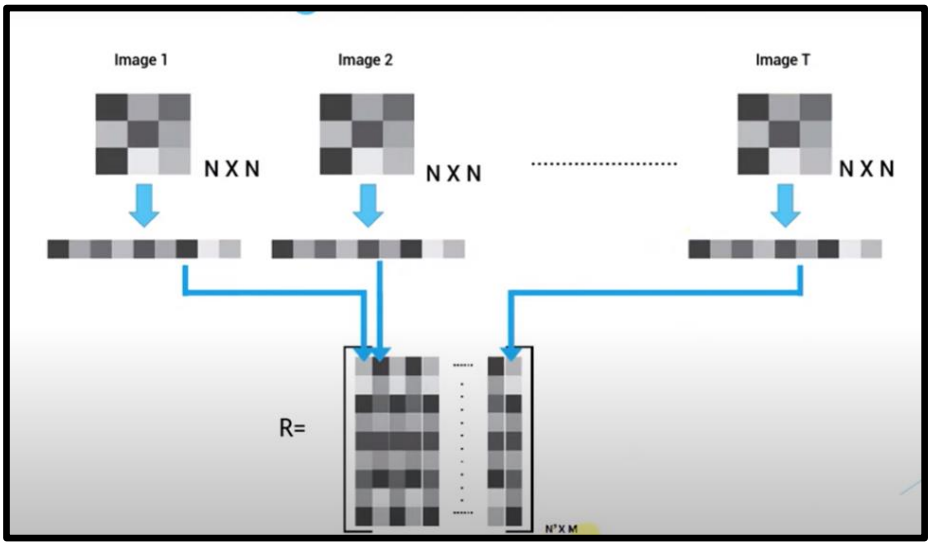
Add images Path to training array **Test array**

Label person Array [P1 P2 P3 P4 P5 P6 ..]

Here the dataset class is will store the images in array, the blue colored one has 8 images and all the path of the images are stored in a training rray, and is utilized for the sake of training, and the test array is for storing two other images, and reason we need this is to check if our model after training is able to find the images in the test array. And there is another array lable person, which stores ID of each person as a array.

2.2.1.2. Image to matrix

The need of this class is converting a image to matrix, so the baove dataset class give set images, in our case it in 2Dim, but there can be cases where the image can be in RGB so we need to convert first to greyscale image then, after converting we will convert now the 2dim array to a single 1dim.



So each images are converted to a single 1Dim, and this are appended as a column in the matrix R, and has dimensions as NxM. This class after converting the images given from dataset class, now this images are stored as matrix, and now this full whole matrix is returned back to the main class, this whole matrix is nothing but a 1Dimension of each image.

2.2.1.3. algo (PCA)

This class is for applying the algorithms that we know, and below are the functions that comes under this class.

- Create Function For
  1. Reduce Dimension And Give New Coordinates
     i. Apply SVD and project old coordinates on new bases by dot product
- Image To Array
- New Coordinates For New lmage
  2. Project old coordinates on new bases by dot produşt
- Show Image
- Recognize Face
  3. Find Mean Of Each Class in New Coordinates
  4. Find distance of image from each mean
- mean with minimum distance is class of images

these are the functions that I have created in this class, now next is the main class where all the dots dicussed above meet in a common point that is this main class.

2.2.1.4. main

- Import Useful Libraries And Classes
- Get Images Name For Training And Testing Using Dataset Class
- Get Scaled Matrix of Training lmages
- Get reduced dimension, new-coordinates from PCA class
- Recoginize Test image using PCA Class.

This are the roles of the main class.

So above I discussed is the basic methadology and the class's that are involved, in the project next we will explain the working of the code, code explanation etc.

2.3. Experiment

In the next part of the report, I'll explain the code in detail. I'll break down how it works and what each part does. To make it even clearer, I'll also include examples of the actual code so you can see it in action. This way, you can get a better idea of how everything fits together and works. So we will see all the above dicussed class's code, and the code explanation under this section. First we will see dataset class.

2.3.1. Code for face_recognition class

Below is the code and the explnation for the face_recognition class, for clear understanding of the need of this class refer above methadology section of this report.

2.3.1.1. Code for Face_Recognition class

```
import cv2
import time
import numpy as np

# importing algorithms
from PCA import pca_class
from TwoDPCA import two_d_pca_class
from TwoD_Square_PCA import two_d_square_pca_class

# importing feature extraction classes
from images_to_matrix import images_to_matrix_class
from images_matrix_for_2d_square_pca import
images_to_matrix_class_for_two_d
from dataset import dataset_class


if __name__ == '__main__':

    # Algo Type (pca, 2d-pca, 2d2-pca)
    algo_type = input('enter the algo listed below by their number
\n1.)pca\n2.)2d-pca\n3.)2d2-pca')

    # for single image = 0
    # for video = 1
    # for group image = 2
    reco_type = int(
        input('enter the application listed below \n1.)for single
image = 0\n2.)for video = 1\n3.)for group image = 2'))

    # No of images For Training(Left will be used as testing Image)
    no_of_images_of_one_person = 8
    dataset_obj = dataset_class(no_of_images_of_one_person)

    # Data For Training
    images_names = dataset_obj.images_name_for_train
```

```
    y = dataset_obj.y_for_train
    no_of_elements = dataset_obj.no_of_elements_for_train
    target_names = dataset_obj.target_name_as_array

    # Data For Testing
    images_names_for_test = dataset_obj.images_name_for_test
    y_for_test = dataset_obj.y_for_test
    no_of_elements_for_test = dataset_obj.no_of_elements_for_test

    training_start_time = time.process_time()
    img_width, img_height = 50, 50

    if algo_type == "pca":
        i_t_m_c = images_to_matrix_class(images_names, img_width,
img_height)
    else:
        i_t_m_c = images_to_matrix_class_for_two_d(images_names,
img_width, img_height)

    scaled_face = i_t_m_c.get_matrix()

    if algo_type == "pca":
        cv2.imshow("Original Image",
        cv2.resize(np.array(np.reshape(scaled_face[:, 1],
[img_height, img_width]), dtype=np.uint8),(200, 200)))
        cv2.waitKey()
    else:
        cv2.imshow("Original Image", cv2.resize(scaled_face[0],
(200, 200)))
        cv2.waitKey()

    # Algo
    if algo_type == "pca":
        my_algo = pca_class(scaled_face, y, target_names,
no_of_elements, 90)
    elif algo_type == "2d-pca":
        my_algo = two_d_pca_class(scaled_face, y, target_names)
    else:
        my_algo = two_d_square_pca_class(scaled_face, y,
target_names)

    new_coordinates = my_algo.reduce_dim()
    if algo_type == "pca":
        my_algo.show_eigen_face(img_width, img_height, 50, 150, 0)

    if algo_type == "pca":
        cv2.imshow("After PCA Image", cv2.resize(

np.array(np.reshape(my_algo.original_data(new_coordinates[1, :]),
[img_height, img_width]), dtype=np.uint8),
            (200, 200)))
        cv2.waitKey()
    else:
        cv2.imshow("After PCA Image",
cv2.resize(np.array(my_algo.original_data(new_coordinates[0]),
dtype=np.uint8), (200, 200)))
```

```
            cv2.waitKey()

        training_time = time.process_time() - training_start_time

        # Reco
        if reco_type == 0:
            time_start = time.process_time()

            correct = 0
            wrong = 0
            i = 0
            net_time_of_reco = 0

            for img_path in images_names_for_test:

                time_start = time.process_time()
                find_name =
my_algo.recognize_face(my_algo.new_cord(img_path, img_height,
img_width))
                time_elapsed = (time.process_time() - time_start)
                net_time_of_reco += time_elapsed
                rec_y = y_for_test[i]
                rec_name = target_names[rec_y]
                if find_name is rec_name:
                    correct += 1
                    print("Correct", " Name:", find_name)
                else:
                    wrong += 1
                    print("Wrong:", " Real Name:", rec_name, "Rec Y:",
rec_y, "Find Name:", find_name)
                i += 1

            print("Correct", correct)
            print("Wrong", wrong)
            print("Total Test Images", i)
            print("Percent", correct / i * 100)
            print("Total Person", len(target_names))
            print("Total Train Images", no_of_images_of_one_person *
len(target_names))
            print("Total Time Taken for reco:", time_elapsed)
            print("Time Taken for one reco:", time_elapsed / i)
            print("Training Time", training_time)

        # For Video

        if reco_type == 1:
            face_cascade =
cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xm
l')

            cap = cv2.VideoCapture(0)

            while True:
                ret, frame = cap.read()

                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
            faces = face_cascade.detectMultiScale(gray,
scaleFactor=1.5, minNeighbors=7)

            i = 0
            for (x, y, w, h) in faces:
                roi_gray = gray[y:y + h, x:x + w]
                scaled = cv2.resize(roi_gray, (img_height,
img_width))
                rec_color = (255, 0, 0)
                rec_stroke = 2
                cv2.rectangle(frame, (x, y), (x + w, y + h),
rec_color, rec_stroke)

                new_cord = my_algo.new_cord_for_image(scaled)
                name = my_algo.recognize_face(new_cord)
                font = cv2.FONT_HERSHEY_SIMPLEX
                font_color = (255, 255, 255)
                font_stroke = 2
                cv2.putText(frame, name + str(i), (x, y), font, 1,
font_color, font_stroke, cv2.LINE_AA)
                i += 1

            cv2.imshow('Colored Frame', frame)
            if cv2.waitKey(20) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()

    # For Image

    if reco_type == 2:
        face_cascade =
cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xm
l')
        dir = r'images/Group/'

        frame = cv2.imread(dir + "group_image.jpg")

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=3)

        i = 0

        for (x, y, w, h) in faces:
            roi_gray = gray[y:y + h, x:x + w]
            scaled = cv2.resize(roi_gray, (img_height, img_width))
            rec_color = (0, 255, 0)
            rec_stroke = 5
            cv2.rectangle(frame, (x, y), (x + w, y + h), rec_color,
rec_stroke)

            new_cord = my_algo.new_cord_for_image(scaled)
```

```
            print("New Cord PCA" + str(i), new_cord)
            name = my_algo.recognize_face(new_cord)
            font = cv2.FONT_HERSHEY_SIMPLEX
            font_color = (255, 0, 0)
            font_stroke = 5
            cv2.putText(frame, name + str(i), (x, y), font, 8,
font_color, font_stroke, cv2.LINE_AA)
            i += 1
            # cv2.imshow('Face', scaled)
            # cv2.waitKey()

        frame = cv2.resize(frame, (1080, 568))
        cv2.imshow('Colored Frame', frame)
        cv2.waitKey()
```

2.3.1.2. Code explanation for face_recognition class

This Python script constitutes a comprehensive facial recognition system that employs various dimensionality reduction algorithms, including PCA (Principal Component Analysis), 2D PCA (Two-Dimensional PCA), and 2D2 PCA (Two-Dimensional Square PCA). The code initiates by importing essential libraries and algorithm classes, such as OpenCV for image processing and the implemented PCA-related classes. User interaction is facilitated by prompts for selecting the algorithm type (pca, 2d-pca, 2d2-pca) and the application scenario (single image, video, or group image). Following this, the script initializes a dataset class to manage the organization of facial image data. It proceeds to extract relevant information for both training and testing sets, including file paths, labels, and the number of elements per class. Subsequently, the chosen algorithm is applied to reduce the dimensionality of the facial data, and the original and processed images are displayed for visual inspection.

2.3.2. experiment for PCA class

Below is the code and the explnation for the PCA a class, for clear understanding of the need of this class refer above methadology section of this report.

2.3.2.1. Code for PCA class

```
import numpy as np
import cv2
import scipy.linalg as s_linalg


class pca_class:


    def give_p(self, d):
        sum = np.sum(d)
```

```
        sum_85 = self.quality_percent * sum/100
        temp = 0
        p = 0
        while temp < sum_85:
            temp += d[p]
            p += 1
        return p

    def reduce_dim(self):

        p, d, q = s_linalg.svd(self.images, full_matrices=True)
        p_matrix = np.matrix(p)
        d_diag = np.diag(d)
        q_matrix = np.matrix(q)
        p = self.give_p(d)
        self.new_bases = p_matrix[:, 0:p]
        self.new_coordinates = np.dot(self.new_bases.T, self.images)
        return self.new_coordinates.T


    def __init__(self, images, y, target_names, no_of_elements,
quality_percent):
        self.no_of_elements = no_of_elements
        self.images = np.asarray(images)
        self.y = y
        self.target_names = target_names
        mean = np.mean(self.images, 1)
        self.mean_face = np.asmatrix(mean).T
        self.images = self.images - self.mean_face
        self.quality_percent = quality_percent

    def original_data(self, new_coordinates):
        return self.mean_face + (np.dot(self.new_bases,
new_coordinates.T))


    def show_eigen_face(self, height, width, min_pix_int,
max_pix_int, eig_no):
        ev = self.new_bases[:, eig_no:eig_no + 1]
        min_orig = np.min(ev)
        max_orig = np.max(ev)
        ev = min_pix_int + (((max_pix_int - min_pix_int)/(max_orig -
min_orig)) * ev)
        ev_re = np.reshape(ev, (height, width))
        cv2.imshow("Eigen Face " + str(eig_no),
cv2.resize(np.array(ev_re, dtype = np.uint8),(200, 200)))
        cv2.waitKey()

    def new_cord(self, name, img_height, img_width):
        img = cv2.imread(name)
        gray = cv2.resize(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY),
(img_height, img_width))
        img_vec = np.asmatrix(gray).ravel()
        img_vec = img_vec.T
        new_mean = ((self.mean_face * len(self.y)) +
img_vec)/(len(self.y) + 1)
```

```
        img_vec = img_vec - new_mean
        return np.dot(self.new_bases.T, img_vec)

    def new_cord_for_image(self, image):
        img_vec = np.asmatrix(image).ravel()
        img_vec = img_vec.T
        new_mean = ((self.mean_face * len(self.y)) + img_vec) /
(len(self.y) + 1)
        img_vec = img_vec - new_mean
        return np.dot(self.new_bases.T, img_vec)

    def recognize_face(self, new_cord_pca, k=0):
        classes = len(self.no_of_elements)
        start = 0
        distances = []
        for i in range(classes):
            temp_imgs = self.new_coordinates[:, int(start):
int(start + self.no_of_elements[i])]
            mean_temp = np.mean(temp_imgs, 1)
            start = start + self.no_of_elements[i]
            dist = np.linalg.norm(new_cord_pca - mean_temp)
            distances += [dist]
        min = np.argmin(distances)

        #Temp Threshold
        threshold = 100000
        if distances[min] < threshold:
            print("Person", k, ":", min, self.target_names[min])
            return self.target_names[min]
        else:
            print("Person", k, ":", min, 'Unknown')
            return 'Unknown'
```

2.3.2.2. Code explanation for PCA class

`pca_class` for performing Principal Component Analysis (PCA) on a dataset of facial images. The class includes methods for dimensionality reduction, displaying eigenfaces, computing new coordinates for input images, and recognizing faces based on a given set of training images. It utilizes libraries such as NumPy, OpenCV (`cv2`), and SciPy's linear algebra module (`scipy.linalg`). The PCA is computed using Singular Value Decomposition (SVD), and the class encapsulates functionalities for calculating principal components, reducing dimensions, and recognizing faces by comparing distances between new coordinates and mean vectors of known classes. The code also incorporates methods to visualize eigenfaces and handles the recognition process, indicating the recognized person or 'Unknown' if the face falls below a specified threshold.

3.1. Result

Above are the code, and explanation now we will see the input faces that we have given, and the output that we have got after executing this codes. First we will see the output images genrated.

3.1.1. input ORL dataset

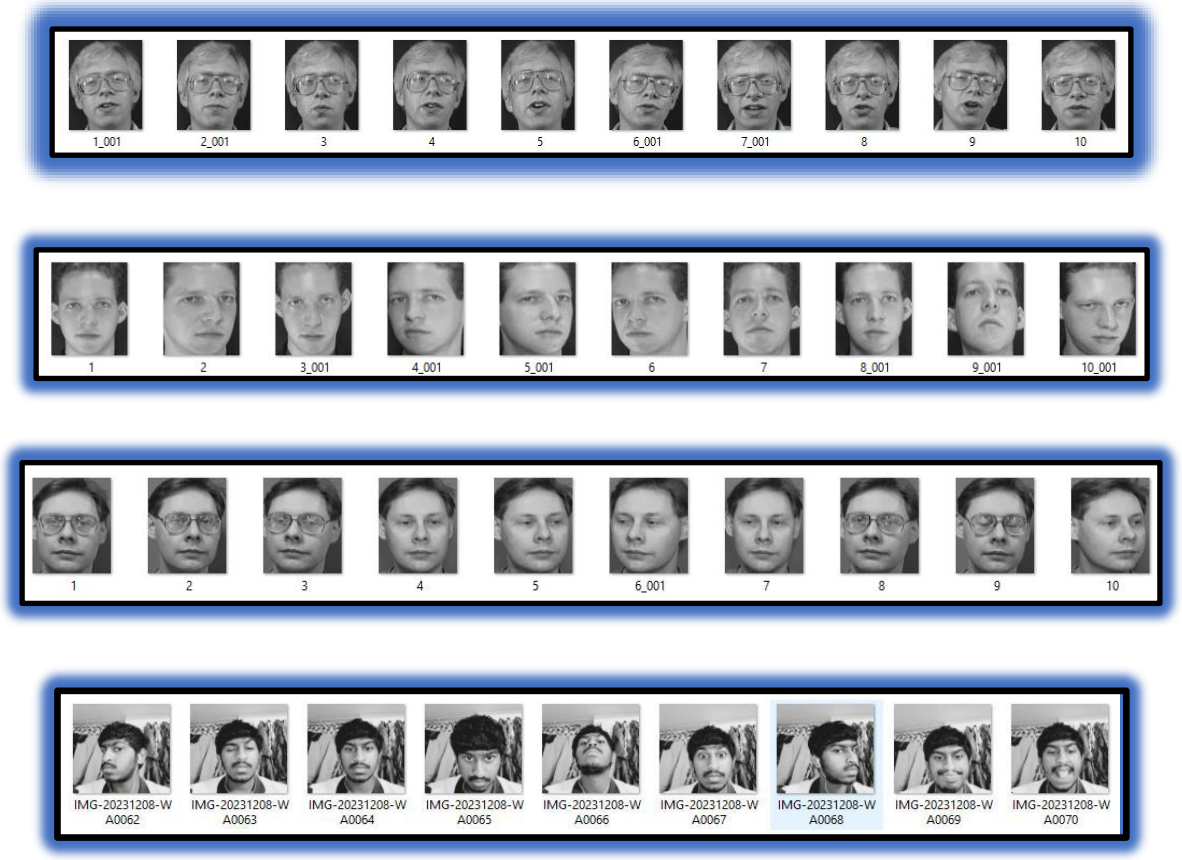


Figure 1 *input images of three faces*

Above are the input for single person's, obtained from ORL dataset.

3.1.2. result for single person and using PCA

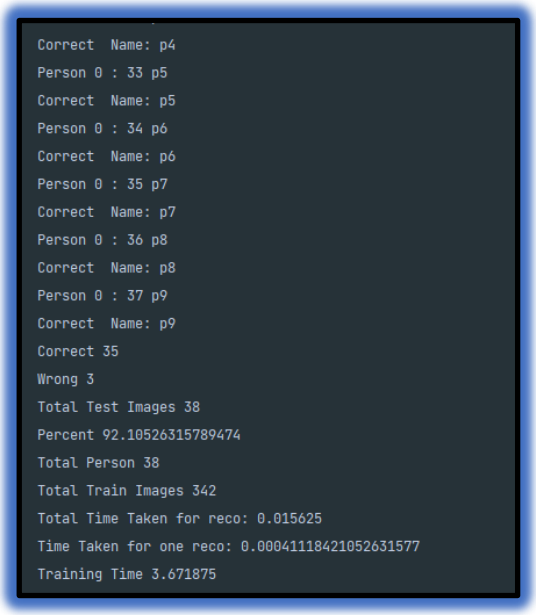


Figure 2 *output of PCA of single after running*

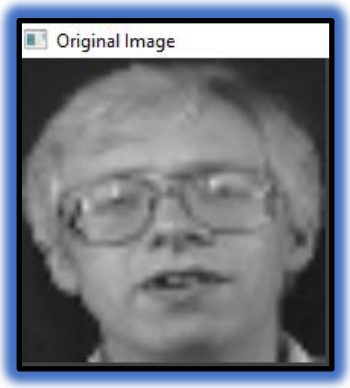


Figure 3 *Origial image of a person 1*

Figure 4 *eigen face of person 0*

3.1.3. Output – After Face Recognized

Figure 5 *output o some of the faces after recoginizing*

The above are the results that we have got after applying PCA on single images. Next we will conclude this chapter, with the conclusion and inference.

4.1. Discussion

The challenge of dimensionality in image processing has long posed a significant obstacle, affecting the efficiency and accuracy of recognition systems. However, in our project, we successfully mitigate this issue by employing Principal Component Analysis (PCA) to reduce the dimensionality of our data. Specifically, we adopt the Singular Value Decomposition (SVD) approach, which has demonstrated superior effectiveness compared to methods such as the covariance matrix eigenvalue decomposition.

By leveraging PCA, we effectively simplify the representation of facial features, emphasizing the most significant components while reducing noise and redundancy. This streamlined representation not only enhances the computational efficiency of our face recognition system but also contributes to its accuracy. By focusing on the most informative features, PCA enables our model to discern subtle patterns and variations crucial for accurate identification.

Moreover, our model provides valuable insights into its performance, including metrics such as total correct classifications, incorrect classifications, and the percentage of accuracy achieved. These metrics offer a comprehensive evaluation of the model's effectiveness in recognizing and classifying faces.

Overall, the successful implementation of PCA, particularly through the SVD approach, underscores its importance as a powerful technique in image processing and face recognition. By addressing the dimensionality curse and optimizing the representation of facial data, PCA significantly enhances the capabilities of our recognition system, paving the way for more efficient and accurate identification of individuals.

4.2. Conclusion

In conclusion, the utilization of Principal Component Analysis (PCA) in face recognition has demonstrated remarkable effectiveness. By condensing facial features into a concise set of essential components, PCA allows for the prioritization and identification of crucial facial variations, facilitating precise recognition while reducing data complexity. This method simplifies the representation of faces, streamlining the recognition process and rendering it more efficient and adaptable for a wide range of applications. Through PCA, we achieve a significant advancement in image processing, harnessing dimensionality reduction techniques to extract and highlight the fundamental characteristics of facial data, thereby enhancing the robustness and efficiency of recognition systems.

4.3. Inference

Inference: The use of Principal Component Analysis (PCA) for face recognition simplifies the representation of facial features, allowing for efficient identification by emphasizing the most crucial components. This reduction in complexity enhances the accuracy and speed of the face recognition process, making PCA a valuable approach in the field.

4.4. Future Work

- **Video Analysis:** Implement face tracking model to follow individuals across multiple frames, improving recognition accuracy and reducing computational load.
- **Group Image Analysis:** The model to handle group images, accurately detecting and recognizing multiple faces within a single frame.
- **Real-time Face Recognition:** Extend the model to process in real-time, detecting and recognizing faces frame-by-frame.

4.5. References

- PRINCIPAL COMPONENT ANALYSIS IN IMAGE PROCESSING by M. Mudrov´, A. Proch´azka mudrova.pdf (vscht.cz)
- An improved face recognition technique based on modular PCA approach by Rajkiran Gottumukkal - An improved face recognition technique based on modular PCA approach - ScienceDirect
- Introduction-to-Machine-Learning/Face Recognition Using PCA at master · codeheroku/Introduction-to-Machine-Learning (github.com)