**An Efficient Approach to Real-Time Face Recognition From Digital images.**

**A Project Report**

**Submitted By**

**MD. Raihanul Kabir**

**ID No- 1608347**

**Session- 2020**

****

**DEPARTMENT OF STATISTICS**

**Hajee Mohammad Danesh Science And Technology University Dinajpur-5200**

**An Efficient Approach to Real-Time Face Recognition From Digital images.**

**A Project Report**

**Submitted By**

**MD. Raihanul Kabir**

**ID No- 1608347**

**Submitted to**

**Rajib Dey**

**(Associate Professor)**

**CO-SUPERVISOR**

**A.S.M Abu Saeed**

**(Assistant Professor)**

**SUPERVISOR**

**Prof. Dr. Md. Earfan Ali Khondaker**

**CHAIRMAN OF EXAMINATION COMMITTEE**

**DEPARTMENT OF STATISTICS**

**Hajee Mohammad Danesh Science And Technology University Dinajpur-5200**

**CHAPTER ONE**

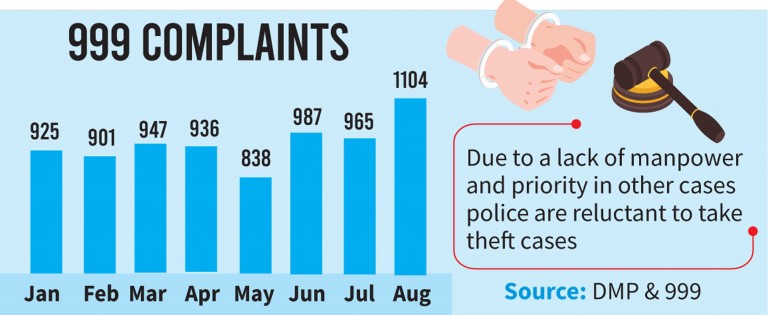
**1.1 Introduction**

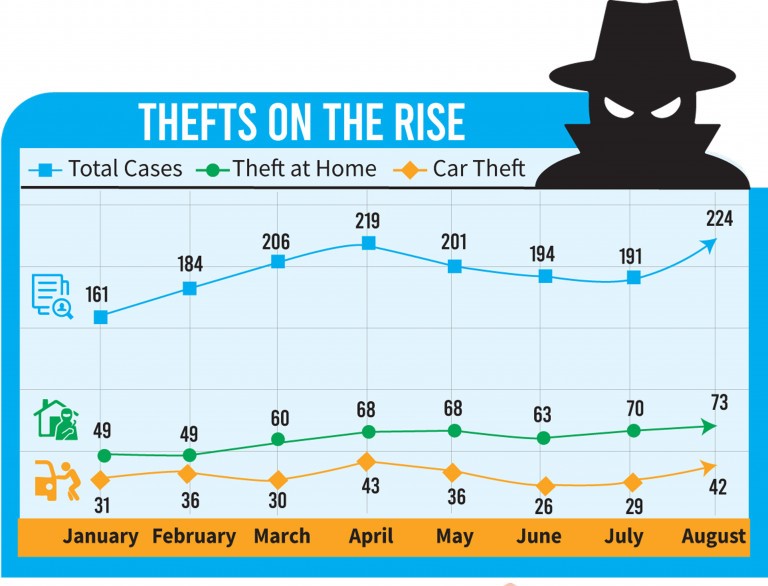
Human faces are significant components of human identity and a distinctive trait that varies from person to person. People are still using human guards to check their ID cards when entering a restricted or reserved area, which is neither an absolute nor efficient approach in this modern age. Furthermore, this system is easy to break through because anyone can use another's cards or create duplicate ones. Governments, Banks, and a few institutions are using Fingerprint systems for checking identities, Which is expensive and its equipment needs to be repaired repeatedly. On the other hand, Police are trying to track the victim using their manpower. It is also not an efficient way in this modern era.

"THE BUSINESS STANDARD" published news on 12 October 2022 with the headings "Most theft cases remain unsolved with little or no headway in probes". [[1](https://www.tbsnews.net/bangladesh/crime/most-theft-cases-remain-unsolved-little-or-no-headway-probes-512182)]

According to The Business Standard, at the crime review meeting at police headquarters, the issue of increasing theft came up for discussion. in the meeting, there were 516 theft cases across the country in July. The number increased to 618 in August.

According to DMP case data, 1580 burglary cases have been recorded from January to August 2022. The national emergency service hotline 999 received 7,503 calls reporting burglary from January to August. In January 925 theft incidents were reported to 999 while 1,104 cases were reported in August. The Statistics are given table:





As So solve this cases Face recognition will help as Face is the unique identity of every human being.

Facial recognition combined with the Internet of Things (IoT) will be much more secure than any other biometric system. Face recognition technology is related to the Internet of Things (IoT) because it can be integrated with IoT devices to enhance their functionality. For example, face recognition can be used in smart homes to unlock doors, control lighting, and adjust the temperature based on the presence of recognized individuals. In security systems, face recognition can be used to monitor and track individuals, as well as provide personalized access control. In the retail industry, face recognition can be used for customer identification and personalization and for tracking customer behavior and preferences. Overall, integrating face recognition technology with IoT devices can lead to more convenient and efficient experiences, while potentially improving security and privacy.

There are two parts to our projects. The First one is, Detecting faces from input images that are captured by any surveillance or webcams. This model also can find out the persons in group photos. The Last one is, it can recognize human identities in real-time live cams.

This project studies some well know face recognition algorithms and makes a comparison of their recognition accuracies both on the train and test sets. Eigen-faces, PCA, SVM, KNN, and CNN are chosen in this experimental study. A variation of face viewpoints is the factor that has been used in the experiment to study the effect of recognition accuracy.

**1.2 Objectives of the study**

The goal of this project is to make an automated security system based on artificial intelligence. That can provide us with more and more security without manpower at a low cost. This system uses Artificial Intelligence (AI) to learn from input data (images) and will remember those features for future face recognition. The principal objectives are:

* To develop a face recognition system.
* To ensure the accuracy and reliability of the system.
* To identify individuals based on their facial features.
* To match individuals against a large database of stored images.
* To maintain the privacy and security of personal data.
* To design the system to adapt and learn from new data.
* To continually improve the system's recognition capabilities over time.

**Outline of this Paper:**

This paper is organized into Five chapters. In Chapter 2.1, I have mentioned the name and brief

**CHAPTER TWO**

**Literature Review:**

A significant number of studies have been conducted on face recognition systems that operate in real-time. A common approach to face recognition is to extract facial features such as eyes, nose, and mouth and then compare them to a database of known faces. This method is known as feature-based recognition, which can be computationally expensive and time-consuming.

To overcome these limitations, researchers have developed an efficient approach to real-time face recognition using deep learning techniques. Deep learning has shown remarkable performance in various applications, including face recognition. This approach involves training a convolutional neural network (CNN) to learn the features of faces from a large dataset. The CNN can then be used to classify new faces in real-time.

A recent study by Zhang et al. (2021) proposed an efficient face recognition system that combines deep learning and dimensionality reduction techniques. The system utilizes a pre-trained CNN to extract features from the face images and then applies principal component analysis (PCA) to reduce the dimensionality of the features. The reduced feature vectors are then compared to a database of known faces using Euclidean distance. The experimental results showed that this approach achieves high accuracy in face recognition while maintaining low computational costs.

Another study by Liu et al. (2020) proposed a real-time face recognition system that uses a Siamese network to learn the similarity between faces. The Siamese network is a type of CNN that uses two identical networks to process two input face images and produces a similarity score. This approach does not require a database of known faces, making it suitable for real-time applications.

**CHAPTER THREE**

**PRELIMIONARY CONCEPTS:**



**Introduction To Machine Learning**

Machine learning is a branch of artificial intelligence that deals with creating algorithms that can learn and improve from experience, without being explicitly programmed. The concept of machine learning can be traced back to the mid-twentieth century when researchers began exploring the idea of creating computer programs that could "learn" from data. However, it wasn't until the late 1990s that machine learning began to take off as a viable field of study.

The history of machine learning can be broadly divided into three phases. The first phase, which lasted from the 1950s to the 1980s, was focused on rule-based systems that were manually designed by experts in specific domains. These systems were able to make decisions based on a set of predetermined rules and were used in a variety of applications, such as expert systems for medical diagnosis and financial forecasting.

The second phase of machine learning, which lasted from the late 1980s to the early 2000s, was characterized by the emergence of statistical learning methods such as decision trees, neural networks, and support vector machines. These methods allowed machines to learn from data without being explicitly programmed and were used in a wide range of applications, including speech recognition, image recognition, and natural language processing.

The third and current phase of machine learning is often referred to as "deep learning" and is based on the use of neural networks with many layers. These networks are able to learn from massive amounts of data and have been used to achieve breakthroughs in fields such as computer vision, speech recognition, and natural language processing. Deep learning has also enabled the development of autonomous vehicles, which rely on machine learning algorithms to make decisions in real-time.

Machine learning has revolutionized many industries, including finance, healthcare, and retail. It has enabled businesses to process and analyze large amounts of data quickly and accurately, leading to improved decision-making and greater efficiency. Machine learning has also contributed to the development of personalized medicine, where treatments are tailored to individual patients based on their genetic makeup and medical history.

**Popular Machine Learning Method**

**Supervised Learning**

Supervised learning is a type of machine learning in which an algorithm learns to make predictions by training on a dataset that contains labeled examples. These labeled examples consist of input data and their corresponding output data, which is also known as the target variable. During training, the algorithm learns to map the input data to the correct output by adjusting its internal parameters based on the labeled examples. Once the model is trained, it can be used to make predictions on new, unseen data.

Now, let's take a closer look at some of the most commonly used supervised learning algorithms:

**Linear Regression:**

Linear regression is a simple algorithm used for predicting a continuous output variable based on one or more input variables. It works by fitting a straight line to the data points and then using this line to make predictions. Linear regression is commonly used in finance, economics, and social sciences for making predictions based on historical data.

**Logistic Regression:**

Logistic regression is used for classification problems where the output is a categorical variable. It works by estimating the probability that a particular input belongs to each class, and then assigning the input to the class with the highest probability. Logistic regression is widely used in healthcare, marketing, and credit scoring.

**Decision Tree:**

A decision tree is a tree-like model that uses a set of rules to make predictions. It works by recursively partitioning the data into subsets based on the values of the input variables, and then using a simple rule to make a prediction for each subset. Decision trees are commonly used in finance, healthcare, and fraud detection.

**Random Forest:**

Random Forest is an ensemble learning algorithm that uses multiple decision trees to improve the accuracy of the model. It works by training a set of decision trees on randomly selected subsets of the data, and then averaging their predictions to make the final prediction. Random Forest is used in finance, healthcare, and customer segmentation.

**Support Vector Machine (SVM):**

Support Vector Machine is a popular algorithm used for classification problems. It works by finding a hyperplane that maximally separates the input data into different classes. SVM is used in image recognition, bioinformatics, and text classification.

**K-Nearest Neighbor (KNN):**

K-Nearest Neighbor is a non-parametric algorithm used for classification and regression problems. It works by finding the k nearest neighbors to a given input, and then assigning the input to the class or predicting the output based on the values of its nearest neighbors. KNN is used in pattern recognition, recommender systems, and anomaly detection.

**Neural Networks:**

Neural Networks are powerful algorithms used for solving complex problems such as image recognition, natural language processing, and speech recognition. They work by simulating the structure and function of the human brain, using layers of interconnected nodes that process and transform the input data.

**Naive Bayes:**

Naive Bayes is a probabilistic algorithm used for classification problems. It works by estimating the probability of each input belonging to each class, based on the frequency of each input feature in the training data. Naive Bayes is used in text classification, spam filtering, and sentiment analysis.

**Unsupervised Learning**

Unsupervised learning is a type of machine learning where an algorithm learns to identify patterns in the data without the need for labeled examples. Unlike supervised learning, there is no output variable to predict or guide the learning process. Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any explicit supervision. The goal of unsupervised learning is to discover patterns and relationships within the data without prior knowledge of the underlying structure. The main advantage of unsupervised learning is that it can identify hidden patterns and structures in the data that might not be apparent through traditional statistical analysis.

There are many algorithms in unsupervised learning, but we will focus on three popular ones: k-means clustering, principal component analysis (PCA), and autoencoders.

**K-means clustering:**

K-means clustering is a popular unsupervised learning algorithm used for clustering data. It is a simple and effective algorithm that partitions a dataset into k clusters, where k is a user-defined parameter. The algorithm iteratively assigns each data point to the closest cluster center and then updates the cluster centers based on the mean of the assigned data points. This process continues until the cluster centers no longer change or a maximum number of iterations is reached.

The k-means clustering algorithm is useful for identifying natural clusters in data, such as customer segments, market segments, or clusters of genes. One of the limitations of the k-means clustering is that it assumes that the clusters are spherical and of equal size. If the data is not well separated or the clusters are of different sizes, the algorithm may not work well.

**Principal component analysis (PCA):**

PCA is a dimensionality reduction technique used to reduce the number of variables in a dataset while retaining as much of the original variance as possible. The goal of PCA is to find a new set of orthogonal variables, called principal components, that explain the maximum amount of variance in the data.

The PCA algorithm works by finding the eigenvectors and eigenvalues of the covariance matrix of the data. The eigenvectors are the principal components, and the eigenvalues represent the amount of variance explained by each principal component. The first few principal components are selected to retain most of the variance in the data, while the remaining components are discarded.

PCA is useful for visualizing high-dimensional data and identifying the most important variables in the data. It can also be used to preprocess data before applying other machine learning algorithms. One of the limitations of PCA is that it assumes a linear relationship between variables, so it may not work well for nonlinear data.

**Autoencoders:**

Autoencoders are a type of neural network used for unsupervised learning. They consist of an encoder network that maps the input data to a compressed representation and a decoder network that reconstructs the input data from the compressed representation. The goal of autoencoders is to learn a compact representation of the input data that preserves its essential features.

The autoencoder algorithm works by minimizing the reconstruction error between the input data and the reconstructed data. This is achieved by training the neural network on the input data without any labels. Once the network is trained, it can be used to generate new data that is similar to the input data.

**Convolutional Neural Network:**

A Convolutional Neural Network (CNN) is a deep learning model used primarily for image processing and computer vision tasks. CNNs are designed to recognize visual patterns directly from pixel images with minimal preprocessing.

The CNN architecture consists of convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform a mathematical operation called convolution that filters the input image and extracts features. The pooling layers downsample the output of the convolutional layers, reducing the number of parameters and computation required. The fully connected layers connect all the neurons in one layer to every neuron in the next layer, leading to a high number of parameters.

CNNs have demonstrated outstanding performance in image classification, object detection, segmentation, and other computer vision tasks. Additionally, CNNs have been applied to non-image processing tasks, such as natural language processing and speech recognition.

**CHAPTER FOUR**

**METHODOLOGY**



**PRINCIPAL COMPONENT ANALYSIS:**

Principal Component Analysis (PCA) is a widely used method for face recognition. This technique involves extracting the most significant features from a large dataset of faces by reducing the dimensionality of the data. PCA is based on the idea of transforming the data into a new coordinate system where the features are uncorrelated, and the variability in the data is captured by a few principal components. In this column, we will discuss the theory and equations behind PCA for face recognition.

First, let's define some terms. A face image can be represented as a matrix of pixel values. For example, an image with dimensions 100 x 100 pixels can be represented as a vector of length 10,000. A dataset of face images consists of a collection of these vectors. Each image in the dataset is considered a point in a high-dimensional space, where the dimensionality is the number of pixels in the image. The goal of PCA is to find a lower-dimensional space that captures the most significant features of the data.

PCA starts by centering the data around the mean face image. This is done by subtracting the mean vector from each image in the dataset. The mean vector is calculated as follows:

where N is the number of images in the dataset, and xi is the vector representing the ith image.

Next, PCA finds the principal components of the data by computing the eigenvectors of the covariance matrix. The covariance matrix measures the linear relationship between the pixels in the images. It is calculated as follows:

where T denotes the transpose operation. The covariance matrix C is a square matrix with dimensions equal to the number of pixels in the image.

The eigenvectors of the covariance matrix are computed as follows:

where v is an eigenvector of C, and λ is the corresponding eigenvalue. The eigenvectors are sorted in descending order of their eigenvalues. The eigenvectors with the highest eigenvalues capture the most significant variations in the data. These eigenvectors are known as the principal components.

To reduce the dimensionality of the data, we can project the images onto the principal components. This is done by multiplying the mean-centered image vector xi by the eigenvector matrix V:

where yi is the vector of principal component scores for the ith image. The vector yi is of length k, where k is the number of principal components used for the projection. By choosing a smaller value of k, we can reduce the dimensionality of the data.

To recognize a face image, we first project it onto the principal components using the equation above. We then compare the principal component scores of the test image with those of the images in the dataset. The closest match is the image with the smallest Euclidean distance to the test image in the principal component space.

PCA is a powerful technique for face recognition that involves finding the most significant features of a large dataset of face images. This is done by projecting the images onto a lower-dimensional space that captures the most significant variations in the data. By choosing a smaller value of k, we can reduce the dimensionality of the data, making the recognition process faster and more efficient.

Facial expressions. The flowchart of our Eigenfaces algorithm is mentioned-

input image

Convert the image into face vector

Flatteing of the vector

Split of the vector obtain into train and test set

Standardise both train and test set vector

Fit the model on the test set and compare the accuracy

Train on train set classification Algorithm (KNN, SVM)

Extract PCA components which capture 95% of the variance

Apply PCA on both train and test set

**Support Vector Machine (SVM)**

SVMs are a type of supervised learning algorithm that can be used for classification and regression tasks. The basic idea behind SVMs is to find a hyperplane that separates the data into different classes with the widest possible margin. In the case of face recognition, the SVM tries to find a hyperplane that can separate the face images of different individuals.

The mathematical formulation of SVMs is based on the concept of Structural Risk Minimization (SRM), which seeks to minimize the empirical risk while maximizing the geometric margin. The empirical risk is the number of misclassifications in the training set, while the geometric margin is the distance between the hyperplane and the closest points from each class.

Suppose we have a set of face images {(x1, y1), (x2, y2), ..., (xn, yn)}, where xi is the i-th face image, and yi is the corresponding label indicating the identity of the person in the image. In the case of face recognition, the labels are discrete values that correspond to the identity of the person.

The SVM tries to find a hyperplane that separates the face images of different individuals. The hyperplane is defined by the equation-

where w is a p-dimensional vector and b is a scalar. For linearly separable data, the hyperplane is selected to differentiate the datasets, such that:

And

if yi = -1

The distance between the hyperplane and the closest points from each class is given by:

To find the optimal hyperplane, we need to minimize ||w|| subject to the constraints:

, for all i = 1, 2, ..., n

This is a constrained optimization problem that can be solved using Lagrange multipliers. The Lagrange multipliers introduce a set of dual variables αi, which are used to obtain the optimal hyperplane. The optimization problem can be written as:

To find the optimal solution, we need to solve the saddle point equation:

subject to: αi ≥ 0, for all i = 1, 2, ..., n

The solution to this problem gives us the optimal hyperplane, which can be used to classify new face images.

**K Nearest Neighbors**

KNN (K Nearest Neighbors) is a non-parametric machine learning algorithm that has been widely used for facial recognition. It is a practical approach that is based on identifying facial features such as eyes, nose, eyebrows, mouth, and ears from a source image to recognize a face.

The main advantage of KNN for facial recognition is its robustness. This is achieved by normalizing the size and orientation of the face. KNN achieves this by using distance metrics such as the city-block distance, Euclidean distance, or cosine distance to find the k-closest data points to the query image.

The distance metric used by KNN is defined by the following equations:

City-Block Distance:

Euclidean Distance:

Cosine Distance:

where xi and yi are the features of the query image and the k-closest data points, respectively.

Once the k-closest data points are identified, KNN analyzes them to determine a common class label among the set. The most common class label is then assigned to the query image.

KNN classifier is an extension of the simple nearest neighbor algorithm, which employs non-parametric decision-making based on the distance of query image features from other image features. The algorithm is among the simplest machine learning algorithms and provides faster execution time than other classifier algorithms.

The KNN classifier's value depends on the number of samples used and their topological distribution. Larger values of k reduce noise in classification, but a good k value is selected through heuristic techniques.

In the presence of noise or inconsistent feature scales, evolutionary algorithms can be employed to reduce noise and optimize feature selection.

KNN is a practical and robust approach to facial recognition, and its effectiveness has been demonstrated by its use in various real-world applications. With the availability of large datasets and the continued advancement of machine learning techniques, KNN is expected to remain an important tool for facial recognition and other pattern recognition tasks.

**KNN Model:**

Take K Nearest Neighbour

End

Initialization define K

compute distance between input and trainig data

Sort Distance

Yes

**Fig:** Block Diagram Representation for KNN Model

**Convolutional Neural Networks (CNN)**

Convolutional Neural Networks (CNN) are a powerful class of Artificial Neural Networks (ANNs) that are designed to analyze visual imagery. Developed by Yan LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner in 1998, CNNs have been widely applied in facial recognition, handwritten numeral recognition, and real-time image and video recognition, among other applications.

CNNs are based on the principle of convolution, which is a mathematical operation that describes the relationship between two functions. In the context of CNNs, convolution involves applying a set of filters to an input image in order to extract relevant features, such as edges and textures. This process is performed by a convolutional layer, which is the core building block of a CNN.

A typical CNN consists of multiple layers, including input, output, and hidden layers. The convolutional layer is typically the first hidden layer in a CNN and is responsible for extracting features from the input image. The output of the convolutional layer is then passed to a pooling layer, which reduces the spatial dimensions of the feature map and aggregates the output of nearby neurons.

After the pooling layer, the resulting feature map is flattened into a one-dimensional vector, which is then fed into a fully connected layer. The fully connected layer connects each neuron in the current layer to every neuron in the next layer, allowing the CNN to learn complex representations of the input data. The final output layer of the CNN provides the classification or regression result based on the input image.

One of the key advantages of CNNs is their ability to automatically learn relevant features from raw data without the need for explicit feature engineering. CNNs are also capable of handling input images of various sizes and can learn to recognize features at multiple scales. Additionally, CNNs are capable of dealing with noisy or incomplete data and can be trained using backpropagation, a technique that helps to resolve the vanishing or exploding gradient problem.

CNNs are a highly effective method for analyzing visual data, and their layered architecture allows for the automated extraction of relevant features. By leveraging convolutional layers, pooling layers, and fully connected layers, CNNs are capable of achieving state-of-the-art performance in a variety of image and video recognition tasks.

Fully connected and polling

inception modules

CNN Pooling

convolutional Filters

Image input

loss

**Face Classification and Recognition Module:**

* **Pre-Processing:** Each Image is read from the database and converted into a matrix of dimensions with respect to the image. The images are then standardized and ultimately divided into train and test sets in the ratio of 0.2.
* **PCA:** Using PCA, both training and test data are then analyzed to extract their distinctive features from the images. Then the Eigenfaces are computed from the PCA components which explain 95% of the variance.
* **Classification Module:** The machine learning module receives the matrix thus obtained from PCA and uses it for training.
* **Comparison Module:** Then, we compared the accuracy of recognition produced by SVM, KNN, and CNN on both the test and training sets, concluding that CNN produced the highest accuracy.

**Result:**

**Image Matrix:**

Image Matrix of all images:

[[108 151 165 ... 119 157 155]

[208 208 202 ... 215 216 215]

[157 161 161 ... 77 5 23]

...

[ 38 34 33 ... 167 183 200]

[ 50 52 60 ... 225 116 7]

[123 117 123 ... 95 104 96]]

**Total variance Explained:**

Explained Variance

[1044.96146236 514.94244767 439.48294333 164.94755354 142.88948637

135.17429636 106.76478966 89.28926781 87.63188124 70.5007032

58.57900602 57.40924637 49.03239528 47.14442927 44.09362809

42.02877498 39.22562811 36.02144231 33.96468677 32.45516198

31.73011156 30.44831432 28.28800548 26.61107598 26.35052889

25.14546033 24.85419385 23.27451521 22.78192259 20.65286742

20.34785796 18.89045421 18.33332302 17.40738698 16.68827802

16.32351673 15.40616363 14.68735873 13.96548641 13.93960964

13.61299542 13.27820527 12.87241751 12.60604509 12.29740295

11.80127212 11.41636071 11.10508559 10.9223075 10.70359127

10.39062657 10.08774735 9.79729512 9.45456012 9.14104689

8.96387902 8.7673688 8.41590407 8.1744506 8.08144787

7.9001581 7.59129688 7.49264268 7.31899647 7.22496132

7.0113969 6.96563779]

**PCA Components:**Out of all components we extracted only top 13 components

PCA Components

[[ 0.00451932 0.00406024 0.00345499 ... 0.00813999 0.00713925

0.00723183]

[ 0.02315906 0.02121137 0.02168128 ... 0.01012521 0.00970881

0.00805907]

[ 0.00946961 0.00918824 0.00877639 ... 0.02332733 0.02390956

0.02284894]

...

[-0.02580162 -0.0263666 -0.00925347 ... 0.06055146 0.03023031

0.01625666]

[ 0.0020675 0.00313824 0.00058816 ... -0.02013126 -0.01019577

-0.01134499]

[ 0.00394744 -0.00370526 -0.00384111 ... -0.00160172 -0.00638982

-0.00062367]]

**Transformed Train Matrix:**

Transformed Train Matrix

[[ 15.16182111 -18.16278818 -1.7928073 ... -2.80691332 -5.6665057

-4.10498062]

[-12.2955737 22.04021532 -4.11590343 ... 1.10216229 -2.63196489

-0.42561761]

[ 32.52787767 -59.7337106 10.63405539 ... -1.38342171 2.71253791

1.94568351]

...

[ -8.15607232 23.7479027 13.33085399 ... -4.59556212 6.05577815

3.63658431]

[ 18.37137478 -52.83071374 -43.93723205 ... -0.73213989 3.28354865

1.53991994]

[-20.17717934 15.57503052 -3.1455211 ... -3.0349411 3.74513494

-2.00745506]]

**Transformed Test Matrix:**

Transformed Test Matrix

[[-1.92878485e+01 -1.22934451e+01 -8.63157332e+00 ... 2.46830837e+00

2.74962191e-02 5.56914551e-01]

[-4.39606975e+01 1.94193093e+00 8.60007787e+00 ... 1.92383261e+00

2.27961581e-02 -1.09052606e-01]

[ 5.63193054e+01 -7.68769973e+00 -1.22412564e+01 ... -2.56847949e-01

-6.98840447e-01 6.90509785e-01]

...

[-8.35230313e+00 -2.13819963e+01 -3.96905153e+00 ... 1.91229060e+00

3.96863431e+00 -2.28857436e+00]

[ 3.97882669e+01 3.03777531e+01 -1.34123145e+01 ... 2.30059222e-01

-1.09385017e+00 -2.03598861e-01]

[-3.98731190e+01 -6.59519092e+00 2.78416366e-01 ... -1.06902713e+00

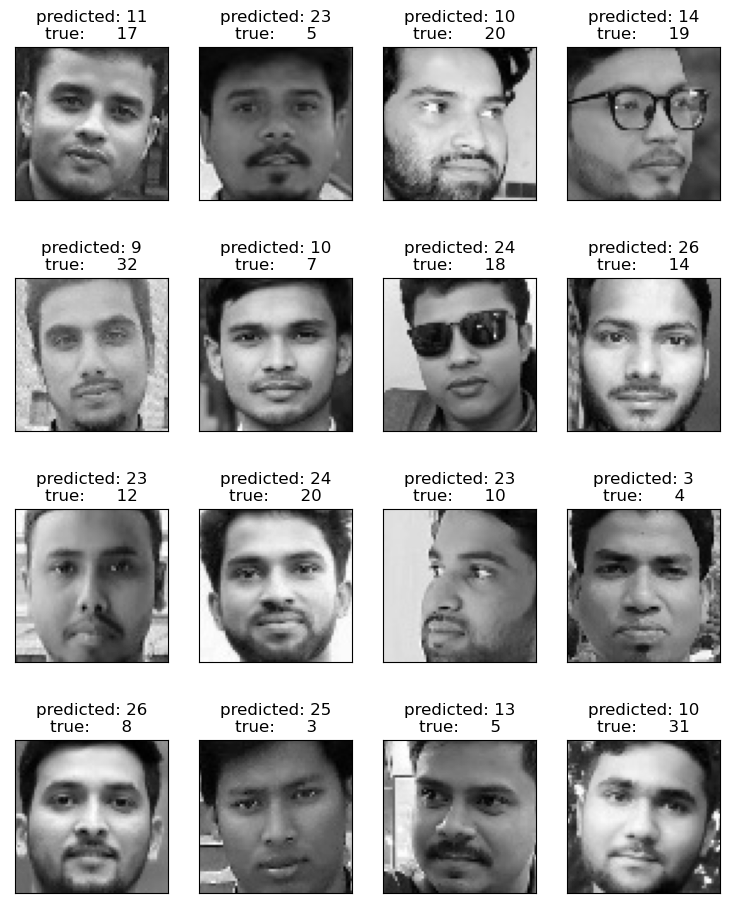
-2.99999862e+00 -4.47392589e-01]]

**KNN**

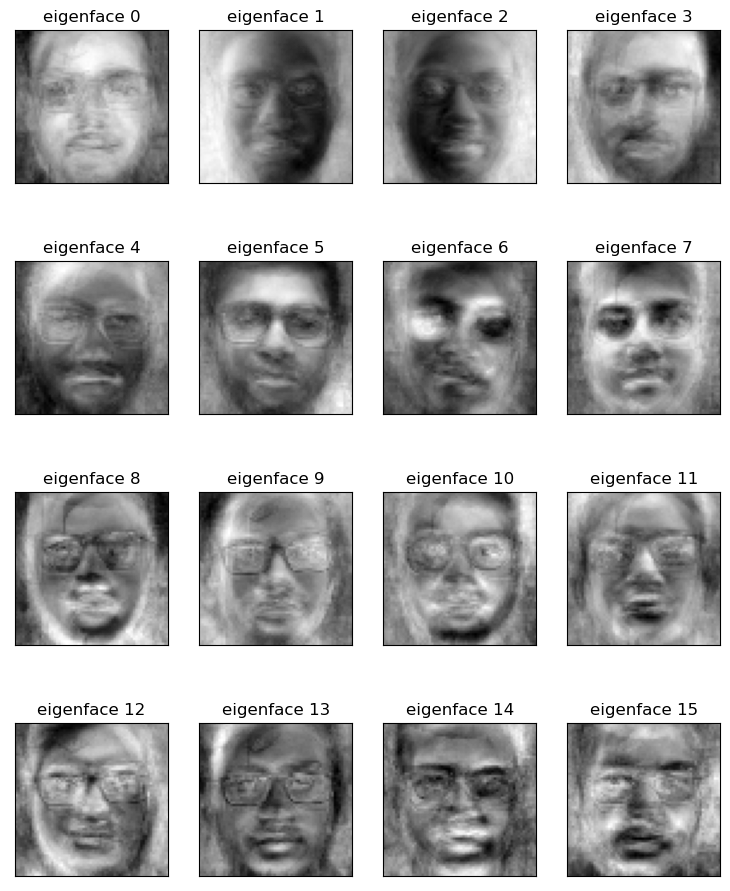
**(Incomplete.............)**

**SVM**

**(Incomplete...............)**

****

**Eigen Faces**

****

**CNN**

**(incomplete........)**

**Code For Face Recognition**

PCA:

from \_\_future\_\_ import print\_function

from time import time

import logging

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

#from sklearn.cross\_validation import train\_test\_split

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import GridSearchCV

from sklearn.decomposition import PCA

from sklearn import decomposition

from sklearn.svm import SVC

from PIL import Image

import os, numpy

from sklearn import metrics, svm, neighbors

from sklearn.metrics import classification\_report

import pandas as pd

os.chdir("C:/Users/R. K Bokul/Desktop/Project\_Face\_Recognition")

path="./datasets/"

X=[]

# h=150

# w=125

h=64

w=64

target\_names=dict()

target\_names={'0':0, '1':1, '2':2, '3':3, '4':4, '5':5, '6':6, '7':7, '8':8, '9':9, '10':10, '11':11, '12':12, '13':13, '14':14, '15':15, '16':16, '17':17, '18':18, '19':19, '20':20, '21':21, '22':22, '23':23, '24':24, '25':25, '26':26, '27':27, '28':28, '29':29, '30':30, '31':31, '32':32, '33':33, '34':34, '35':35, '36':36, '37':37, '38':38, '39':39 }

#######Converting the images into matrix

for filePath in sorted(os.listdir(path)):

    imagePath = os.path.join(path, filePath)

    img=Image.open(imagePath)

    featurevector=numpy.array(img).flatten()

    X.append(featurevector)

X=numpy.asarray(X)

print("Image Matrix of all images:\n")

print(X)

#######Compute Labels

Y = pd.read\_csv('./training\_data/label.csv')

Y=Y['Class\_label'].replace({'amy':1, 'andrew':2, 'andy':3, 'erin':4, 'gabe':5, 'hill':6, 'jack':7, 'zach':8})

Y=Y.values

print(X.shape)

print(Y.shape)

# Y=numpy.asarray(Y).reshape(27,1)

# split into a training and testing set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, Y, test\_size=0.2, random\_state=42)

###############Compute PCA and Eigen faces################

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fit on training set only.

scaler.fit(X\_train)

# Apply transform to both the training set and the test set.

train\_img = scaler.transform(X\_train)

test\_img = scaler.transform(X\_test)

pca = PCA(0.95).fit(train\_img)

print("\nExplained Variance\n")

print(pca.explained\_variance\_)

print("\nPCA Components\n")

print(pca.components\_)

eigenfaces = pca.components\_.reshape((pca.n\_components\_, h, w))

print("\nTransformed Train Matrix\n")

X\_train\_pca = pca.transform(train\_img)

print(X\_train\_pca)

print("\nTransformed Test Matrix\n")

X\_test\_pca = pca.transform(test\_img)

print(X\_test\_pca)

############################# SVM ############################

knn = neighbors.KNeighborsClassifier(n\_neighbors=5, weights='distance', algorithm='auto')

knn.fit(X\_train\_pca, y\_train)

y\_trainPredKNN = knn.predict(X\_train\_pca)

acc\_knnTrain = metrics.accuracy\_score(y\_train, y\_trainPredKNN)

print("Train Set accuracy KNN: {0}".format(acc\_knnTrain))

y\_testPredKNN = knn.predict(X\_test\_pca)

acc\_knnTest = metrics.accuracy\_score(y\_test, y\_testPredKNN)

print("Test Set accuracy for KNN: {0}".format(acc\_knnTest))

cm\_knn = metrics.confusion\_matrix(y\_test, y\_testPredKNN)

print("===========Report for Test Set KNN=============")

print(cm\_knn)

print(classification\_report(y\_test, y\_testPredKNN))

############################# SVM ############################

svm\_classifier=svm.SVC(kernel='linear')

svm\_classifier.fit(X\_train\_pca, y\_train)

preds\_trainSVM = svm\_classifier.predict(X\_train\_pca)

acc\_trainSVM = metrics.accuracy\_score(y\_train, preds\_trainSVM)

print("Train Set accuracy for SVM: {} for {}".format(acc\_trainSVM, 'linear'))

preds\_testSVM = svm\_classifier.predict(X\_test\_pca)

acc\_testSVM = metrics.accuracy\_score(y\_test, preds\_testSVM)

print("Test Set accuracy in SVM: {} for {}".format(acc\_testSVM, 'linear'))

print("=========Report for Test Set SVM in {}".format('linear'))

cm\_svm = metrics.confusion\_matrix(y\_test, preds\_testSVM)

print(cm\_svm)

print(classification\_report(y\_test, preds\_testSVM))

def plot\_gallery(images, titles, h, w, n\_row=4, n\_col=4):

    """Helper function to plot a gallery of portraits"""

    plt.figure(figsize=(1.8 \* n\_col, 2.4 \* n\_row))

    plt.subplots\_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)

    #for i in range(len(images)):

    for i in range(min(n\_row \* n\_col, len(images))):

        plt.subplot(n\_row, n\_col, i + 1)

        plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)

        plt.title(titles[i], size=12)

        plt.xticks(())

        plt.yticks(())

# plot the result of the prediction on a portion of the test set

def title(y\_pred, y\_test, target\_names, i):

    idx\_true=y\_test[i]

    idx\_pred=y\_pred[i]

    pred\_name = list(target\_names.keys())[list(target\_names.values()).index(idx\_pred)]

    true\_name = list(target\_names.keys())[list(target\_names.values()).index(idx\_true)]

    return 'predicted: %s\ntrue:      %s' % (pred\_name, true\_name)

prediction\_titles = [title(preds\_testSVM, y\_test, target\_names, i)

                     for i in range(preds\_testSVM.shape[0])]

plot\_gallery(X\_test, prediction\_titles, h, w)

eigenface\_titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]

plot\_gallery(eigenfaces, eigenface\_titles, h, w)

plt.show()

**CNN**

import os

os.getcwd()

os.chdir("C:/Users/R. K Bokul/Desktop/Project\_Face\_Recognition")

from tensorflow import keras

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense, Dropout, BatchNormalization

from keras.optimizers import SGD, Adam

from keras.constraints import maxnorm

from keras import regularizers

from sklearn.metrics import classification\_report

from sklearn import metrics

import numpy as np

# Initialising the CNN

Model = Sequential()

# Step 1 - Convolution

Model.add(Conv2D(32, (3, 3), padding="same", input\_shape = (64, 64, 3), activation = 'relu'))

Model.add(BatchNormalization())

# Step 2 - Pooling

Model.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a second convolutional layer

Model.add(Conv2D(64, (3, 3), padding="same", activation = 'relu'))

Model.add(BatchNormalization())

Model.add(Dropout(0.1))

Model.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a third convolutional layer

Model.add(Conv2D(128, (3, 3), padding="same", activation = 'relu'))

Model.add(BatchNormalization())

Model.add(Dropout(0.2))

Model.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

Model.add(Flatten())

# Step 4 - Full connection

Model.add(Dense(units = 50, activation = 'relu', kernel\_regularizer=regularizers.l2(0.0001)))

Model.add(Dense(units = 9, activation = 'softmax'))

# Compiling the CNN

optimizer = keras.optimizers.Adam(learning\_rate=0.001)

Model.compile(optimizer = optimizer, loss = 'categorical\_crossentropy', metrics = ['accuracy'])

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255,

                                   shear\_range = 0.2,

                                   zoom\_range = 0.2,

                                   horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255)

training\_set = train\_datagen.flow\_from\_directory('C:/Users/R. K Bokul/Desktop/Project\_Face\_Recognition/CNN\_dataset/ORL\_Database/',

                                                 target\_size = (64, 64),

                                                 batch\_size = 16,

                                                 class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('C:/Users/R. K Bokul/Desktop/Project\_Face\_Recognition/CNN\_dataset/Test\_data/',

                                            target\_size = (64, 64),

                                            batch\_size = 8,

                                            class\_mode = 'categorical')

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define the model architecture

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(5, activation='softmax'))

# Compile the model

model.compile(optimizer='Adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.compile(optimizer='Adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Train the model

history = Model.fit(training\_set, steps\_per\_epoch=151, epochs=5, validation\_data=test\_set, validation\_steps=52)

filenames=test\_set.filenames

nb\_samples=len(filenames)

Y\_pred = Model.predict(test\_set, np.ceil(nb\_samples/8))

y\_pred = np.argmax(Y\_pred, axis=1)

#print(Y\_pred)

print('Confusion Matrix')

print(metrics.confusion\_matrix(test\_set.classes, y\_pred))

print(metrics.accuracy\_score(test\_set.classes, y\_pred))

print('Classification Report')

target\_names = ['Liton', 'Siddique', 'Alomgir Ridoy', 'Toufique', 'Harun Badsha', 'Jakaria', 'Jamiul Islam', 'Juel', 'kanto']

print(classification\_report(test\_set.classes, y\_pred, target\_names=target\_names))

#from skimage.io import imread

#from skimage.transform import resize

#import numpy as np

##

### Class labels

#class\_labels = {v: k for k, v in training\_set.class\_indices.items()}

#class\_labels

##

## reading the input image

#img = imread('data\\single\_prediction\\1.jpg')

#img = resize(img,(64,64))

#img = np.expand\_dims(img,axis=0)

#prediction = classifier.predict\_classes(img)

#prediction

import numpy as np

from keras.preprocessing import image

from tensorflow.keras.preprocessing import image

test\_image = image.load\_img('C:/Users/R. K Bokul/Desktop/Project\_Face\_Recognition/CNN\_dataset/1.jpg', target\_size = (64, 64))

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = Model.predict(test\_image)

# result\_1 = Model.predict\_classes(test\_image)

# result\_1

training\_set.class\_indices

if result[0][0] == 1:

    prediction='Fahad Liton'

elif result[0][0] == 2:

    prediction='Siddique'

elif result[0][0] == 3:

    prediction='Alomgir Ridoy'

elif result[0][0] == 4:

    prediction='Toufique'

elif result[0][0] == 5:

    prediction='Harun Badsha'

elif result[0][0] == 6:

    prediction='Jakaria Rahman'

elif result[0][0] == 7:

    prediction='Jamiul'

elif result[0][0] == 8:

    prediction='Juel'

else:

    prediction='None'

print("The input image is: {0}".format(prediction))