Face Recognition System using PCA and Neural Networks

Authors : Harsh Patel , Harshil Shah, Bansi Gajera, Dhruv Satyapathi

Abstract— **This assignment is a step towards developing a face recognition system which can recognize static images. We have implemented face recognition framework using concepts of Machine learning which is able to detect colored images of humans quite rapidly. The strategy of face recognition involves the examination of facial features in a picture, recognizing those features and matching them to 1 of the many faces in the database. We have used Principal Component Analysis (PCA) method for face recognition. PCA is a statistical approach used for reducing the number of variables in face recognition. PCA extracts the absolute most relevant information within a face and then tries to construct a computational model that best describes it. We have used PCA as it is simple, quick and accurate. The system consists of a database of a set of facial patterns for each individual. The characteristic features called 'eigenfaces' are extracted from the stored images using which the system is trained for subsequent recognition of new images.**

Keywords— PCA, eigen values, eigen vectors, Neural network, Machine learning

I. INTRODUCTION

Face Recognition has been a matter of discussion and research since past few years. It is basically A set of two task 1) Face Identification: Given a face image that belongs to a person in a database, tell whose image it is. 2) Face Verification: Given a face image that might not belong to the database, verify whether it is from the person it is claimed to be in the database. The detection stage is the first stage, it includes identifying and locating a face in an image. The recognition stage is the second stage; it includes feature extraction, where important information for discrimination is saved, and the matching, where the recognition result is given with the aid of a face database. Face is a complex multidimensional structure and needs a good computing techniques for recognition.

Face recognition is an integral part of biometrics. In biometrics basic traits of human is matched to the existing data and depending on result of matching identification of a human being is traced. Facial features are extracted and implemented through algorithms which are efficient and some modifications are done to improve the existing algorithm models. Computers that detect and recognize faces could be applied to a wide variety of practical applications including criminal identification, security systems, identity verification etc. Face detection and recognition is used in many places nowadays, in websites hosting images and social networking sites. Face recognition and detection can be achieved using technologies related to computer science. Features extracted from a face are processed and compared with similarly processed faces present in the database. If a face is recognized it is known or the system may show a similar face existing in database else it is unknown.

This technology has been available for some years now and is being used all over the place. Now a days face recognition is in continuous demand in image processing system due to modernisation and automation. The goal of this project is to implement a face recognition system. Face detection is used to detect a face from an image while face recognition is used to find out whose face is it.

II. EXISTING APPROACHES IN FACE RECOGNITION\

Many face recognition methods have been proposed. In the vast literature on the topic, there are different classifications of the existing techniques. Many techniques have been proposed for face recognition such as PCA (principal component analysis), LBP (local binary patterns), ICA (independent component analysis) LDA (linear discriminant analysis), and many more.

There has been a rapid development of the reliable face recognition algorithms in the last decade. The traditional face recognition algorithms can be categorised into two categories: holistic features and local feature approaches. The holistic group can be additionally divided into linear and nonlinear projection methods. Many applications have shown good results of linear projection based appearance based methods such as Principal component analysis(PCA), Independent Component Analysis(ICA), Linear Discriminant Analysis(LDA), Linear Regression Classifier(LRC)
Here PCA and LDA focus on the global structure of Euclidean Space and LRC focuses on the Local Structure of the Manifold.

These methods project face onto a linear subspace spanned by eigenface images. The distance from face space is the orthogonal distance to the plane whereas the distance in face space is the distance along the plane from the mean image.

Also the the approach of PCA with artificial neural networks is being used where the the data dimensionality is reduced and then it is trained from the images of each individual such that when any external image is testes it tries to match the extractions of it with the given features of the given images in the database. PCA is a statistical approach used for reducing the number of variables in face recognition. In PCA, every image in the training set is represented as a linear combination of weighted eigenvectors called eigenfaces. These eigenvectors are obtained from covariance matrix of a training image set. The weights are found out after selecting a set of most relevant Eigenfaces. Recognition is performed by projecting a test image onto the subspace spanned by the eigenfaces and then classification is done by measuring minimum Euclidean distance.

III. OUR APPROACH

We have used PCA and neural networks for face recognition. PCA is used for reducing dimensionality in a dataset while retaining those characteristics of the dataset that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. By means of PCA we can transform each original image of the training set into a corresponding eigenface.

A. **Dataset and Labeling of images :**
Preprocessing of the data we have resize all the images in (100,100) dimension. And Extracting the labels from the image in Dataset.

B. Eigenfaces approach :
The relevant information of the face image was extracted , encoded and stored in an array i.e the eigenvalue and eigenvector of covariance matrix of an image is computed. A single image can be visualised to be a single point in a n*n space. The (n*n) image is flattened into ($n^2 * 1$) dimension and stored in an array. Eigenvectors with highest value are principal component of the image set. Principal components are used to construct eigen faces.

C. **Finding eigen faces :**

- We created an array of dimension ($n^2 * M$) where M is the number of images in dataset.
- Computed the average image
- Calculated deviation of given image from average image and stored all deviation values in matrix A
- Computed covariance matrix $C = AA^T$
- We are only interested in eigen vectors with highest value. So we drop other vectors.

D. **Transform the dataset into lower dimension**
As PCA is used for lowering the dimension of the data. As each image in dataset are maps to the bases which are eigenfaces. Transformation of the image gives as weights of the dataset image which maps over the eigenvalues.

E. **Neural Networks**
Use of Neural Networks for determining labels from the weights of given images. An artificial neural network (ANN), also called as simulated neural network (SNN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on connectionist approach to computation.

IV. ALGORITHM AND BLOCK DIAGRAM

Algorithm we have used for face recognition is as follows:

Step 1 : Preparing the data
The first step is to obtain a set with all face images. Each image is transformed into a vector and placed into the set

Step 2 : Resizing the image
Image is converted into form RGB to Gray scale. Then it is reshaped into a 100x100 sized image. Labels are extracted from the image.

Step 3 : Feature extraction
Extract the relevant information of the face image, encode it and store in an array. Eigen faces and its corresponding eigen values are computed. From M eigen vectors, ones with highest eigen value are selected. Eigen faces with low eigen value can be omitted.

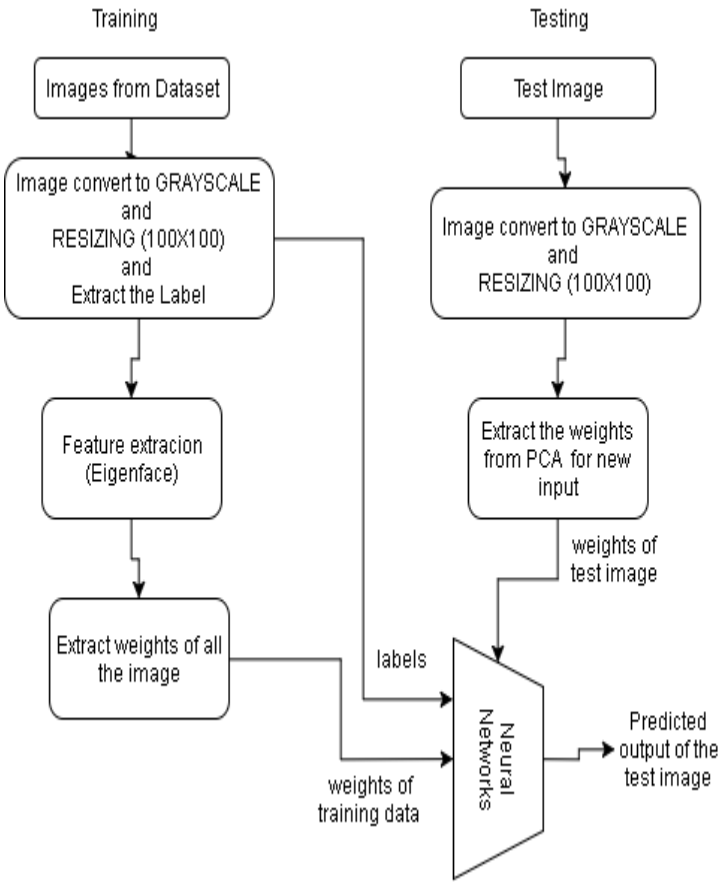
Step 4 : Eigen weights
The new face is transformed into its eigenface components and the resulting weights form the weight vectors

Step 5 : test result
Once the model has been trained, testing is done. Test image is resized and gray scaled. There after, weights are extracted from PCA for new inputs.

Step 6 : Neural network
Test image is fed into Neural network to predict the outcome. Neural Networks is used for determining labels from the weights for given test images.

Step 7 : Result
For a given test image, if corresponding image of that person lies in the database than return the label of that person else return that person is not found.

Block diagram

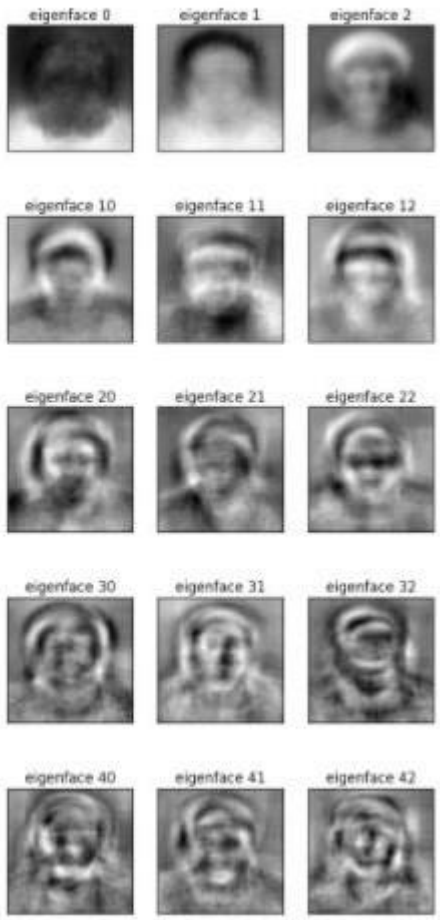


V. EXPERIMENTAL RESULTS

a) Test images

Here the backpropagation method uses the epoch value as 100 and the accuracy to identify an image within the database is almost 95%.
And the data preprocessing is carried out by down sampling the data size using PCA.

```
In [24]: eigenfaces  = tempPCA.components_.res
         eigenface_titles = ["eigenface %d" %
         plot_gallery(eigenfaces, eigenface_ti
         plt.show()
```



VI. EVALUATION METRICS

Training Parameters of the Neural Networks.
Optimizer : adam,
Loss function : categorical_crossentropy
Learning rate : 0.001

VII. DISCUSSIONS

During this face recognition assignment we faced a few challenges.
The key ones are :

1. Improper data : The data we had was not proper as the image size of all samples were not equal. Some consisted of only face while some images were beyond face .
2. Data training : We had a problem while training the data due to improper data. Then we normalized data (divided eigen values by 100) and were able to train it.
3. Few images weren't labelled properly which at times created trouble.

VIII. CONCLUSIONS

The face recognition algorithm using PCA and neural networks is implemented here which has a very non uniform database in the beginning which is made uniform using the PCA thereby reducing the size of each image ,after the facial data is made uniform. It accurately identifies input face images from database of an individual which differ from the set of images of that person already stored in the database thus serving as an effective method of recognizing face images. We are happy to have done this project.

Output and code :

In [24]:
```
from keras.models import Sequential
from keras.layers import Dense
from PIL import Image
import cv2
import numpy as np
import os
from sklearn.decomposition import PCA
```

In [25]:
```
#Data preprocessing
data = []
label = []
unqiue_label = []
for files in os.listdir('F:\MY_DRIVE_F\STUDY_ICT_BTech\ICT Sem 6\Machine_Learning\Face data set\ML face images'):
    f = files.split('_')[0].split('O')[0].split('o')[0]
    if(f!='.DS'):
        tempimage =  cv2.imread('F:\MY_DRIVE_F\STUDY_ICT_BTech\ICT Sem 6\Machine_Learning\Face data set\ML face images/'+files)
        tempgray = cv2.cvtColor(tempimage, cv2.COLOR_BGR2GRAY)
        tempresize = cv2.resize(tempgray,(100,100),interpolation = cv2.INTER_AREA)
        tempNP = np.array(tempresize)
        tempNP = tempNP.flatten()

        data.append(tempNP);
        label.append(f);
        if( f not in unqiue_label ):
            print(f)
            unqiue_label.append(f)

data = np.array(data)
data = data/255
print("data")
print(data[0])
```

```
17440004
1744001
1744002
1744003
1849004
201501004
201501007
201501008
201501009
201501011
201501012
201501021
201501025
201501028
201501031
201501032
201501034
201501038
201501039
201501051
201501053
201501054
201501055
201501060
201501067
201501070
201501071
201501077
201501079
201501086
201501088
201501091
201501095
201501097
201501101
201501104
201501109
```

In [26]:
```
print(data.shape)

tempPCA = PCA(n_components=100)
tempPCA.fit(data)


x = tempPCA.transform(data)
print(x.shape)
print(x[0])
label = np.array(label)

#np.save('X.npy',tempPCA);
#np.save('label.npy', np.array(label));
```

```
(588, 10000)
(588, 100)
[-30.06102539  -6.92679527 -11.8593719    8.95178941  -3.0840769
   6.9723534   -7.33970169   3.79414711  -0.30904446  -1.42346843
   1.87760667  -0.62279725   4.61169305  -3.60660533   2.06225699
  -2.53103852   1.89555088  -4.41489891   1.26214037  -3.5143218
   3.10498945  -1.57437949   2.98765167  -1.72493223  -0.33904555
  -0.80857419   0.12901867  -0.28015839   1.82885659  -0.94237501
   1.62336574  -0.88419734   0.5131754   -0.81889225   0.39450659
  -1.87750219   1.05622556  -0.19150387   3.09918556  -1.40711066
   0.768917    -1.46735929   1.52497384  -1.14020552  -0.08001977
   0.61837497   1.48513006  -0.57901689   0.20475237   0.23777481
   0.88080302  -2.0928183   -0.21130841  -1.06480922   0.54584342
   0.09437264  -0.48886442  -0.41926585  -0.19982913  -1.36920543
   1.41401482   1.73829708  -0.25670303  -0.36285409   1.00444725
  -0.19529363   1.16961983  -0.13122332   0.52330922  -0.05355059
   0.3588198   -0.13227915   0.14094547  -0.6064889    0.41671519
  -0.55573914   0.16554361  -0.23536858   0.14838305  -0.33833714
   0.12319526  -0.80333708   1.07598089  -0.83800866   0.84770247
   0.10345238  -0.41537482   0.52085506   0.21573232  -1.22869266
   0.22142986   0.6314891   -1.28008954  -0.34301546  -0.29556437
```

In [27]:
```
len(unqiue_label)
```

Out[27]:
```
40
```

In [28]:
```
y = []
for value in label:
    oneencode = np.zeros(40, dtype=int)
    np.put(oneencode,unqiue_label.index(value),1)
    y.append(oneencode)

y = np.array(y)
```

In [29]:
```
from keras.layers import Dropout,Input
from keras.models import Model

input_feature_vector = Input(shape=(100,),name='input_layer')
X = Dense(512, activation='relu',name= 'layer_1')(input_feature_vector)
hidden1 = Dense(256,activation='relu',name= 'layer_2')(X)
Y = Dense(40,activation= 'softmax',name= 'output_layer')(hidden1)

model = Model(input_feature_vector, Y)
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 100)               0
_________________________________________________________________
layer_1 (Dense)              (None, 512)               51712
_________________________________________________________________
layer_2 (Dense)              (None, 256)               131328
_________________________________________________________________
output_layer (Dense)         (None, 40)                10280
=================================================================
Total params: 193,320
Trainable params: 193,320
```