FACE RECOGNITION IN ARTIFICIAL INTELLIGENCE

Varun Chauhan(16BCP006), Jeanie Jessica(16BCP016), Nihit Parikh(16BCP027),
Parth Shah(16BCP028), Urja Thakkar(16BCP055), Yaagni Raolji(16BCP064)

ABSTRACT - Face recognition is one of the most prominent features that we find in smart devices, security systems, and social media applications, which form its basis from the concept of artificial intelligence. For the image dataset of a class, this system aspires to detect human face features, human face with expressions, etc. The preprocessing required for solving the face recognition problem comprises of steps such as color normalization(HDTV method), feature extraction using PCA (KLT), face detection using HAAR method, and labeling. The approach for 2D face recognition is to first find the eigen -faces using Principal Component Analysis, then separating the classes (each individual denoting a separate class) using Linear Discriminant Analysis, and finally using Probabilistic function (computing the class prior probability using Naive Bayes Theorem) for testing face to reach to a conclusive state of an output and label the testing data sample. This approach envisions to archive 93% accuracy for face recognition in artificial intelligence.

KEYWORDS - Artificial Intelligence, Face Detection, Face recognition, Feature Extraction, HAAR, Luminosity, Principal Component Analysis

INTRODUCTION

Today in this information era, data is secured by passwords, encryption keys, fingerprints and many other modes. The human face plays an important role in our social interaction and in conveying people's identity. Biometric face recognition technology has received significant attention in the past several years due to its potential for applications in both law enforcement and non-law enforcement agencies.
As compared with other biometrics systems using fingerprint, palm print and iris, face recognition has distinct advantages because of its non-contact process. Images can be captured from a distance without touching the person and face can be extracted from that image. The identification does not require interacting with the person. In addition, recognized face images can be recorded and archival can later help to identify the person(s).

FACE RECOGNITION PROCESS

1. DATA PREPROCESSING

1.1 Image Resize
The dataset contained irregularities in context of the size of the images as the images were taken from different devices. To ease the further operations, all images were converted to one general size - 3456 x 3456 pixels.

1.2 Color Normalization
The image dataset comprises of discrepancies due to lighting and shadow issues. The solution to this problem was to convert out RGB images to GRAY-SCALE. To handle this issue, four approaches were looked at as part of the research: (1) Average method (2) Lightness method (3) HDTV method(Luminosity method) (4) openCV python library..

Original image	
Lightness	

Average	
Luminosity	

FIGURE 1: LUMINOSITY METHOD FOR COLOR NORMALIZATION[1]

(**Luminosity formula:** 0.21 R + 0.72 G + 0.07 B)

Thus after going through all the approaches, the images were converted to GRAY-SCALE using the **openCV library.**

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

1.3 Dimensionality Reduction

Dimension reduction is a process of reducing the number of variables under observation. It deals with the "curse of dimensionality". The need for dimension reduction arises when there is a large number of univariate data points or when the data points themselves are observations of a high dimensional variable. The key observation is that although face images can be regarded as points in a high-dimensional space, they often lie on a manifold (i.e., subspace) of much lower dimensionality, embedded in the high-dimensional image space. The main issue is how to properly define and determine a low-dimensional subspace of face appearance in a high-dimensional image space.[11]

The approaches looked at for this phase of the research were:

(1) Histogram of Oriented Gradients(HOG) + Local Binary Pattern(LBP) + Support Vector Machines(SVM)

(2) Principal Component Analysis.

While researching about the first approach(HOG), it was observed that SVM increases the overall complexity of the program and also it results in lesser accuracy.

The second approach observed that when the training dataset is small, PCA can result in better accuracy, and also PCA is less sensitive to different training datasets. Therefore, this approach was chosen to reduce the overall dimensionality.

Principal Component Analysis

It is a linear transformation that chooses a new coordinate system for the data set such that

- greatest variance by any projection of the data set comes to

lie on the first axis (then called the first principal component),

- the second greatest variance on the second axis,
- and so on.

PCA can be used for reducing dimensionality by eliminating the later principal components.

By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset. This is the principal component.

Let A be an n by n matrix.

- Vectors x having same direction as Ax are

called eigenvectors of A.

- In the equation $Ax=\lambda x$, λ is called an eigenvalue of A.
- $Ax=\lambda x \Leftrightarrow (A-\lambda I)x=0$

Method to calculate x and λ:

- Calculate $\det(A-\lambda I)$, yields a polynomial (degree n)
- Determine roots to $\det(A-\lambda I)=0$, roots are eigenvalues λ
- Solve $(A- \lambda I)\ x=0$ for each λ to obtain eigenvectors x

Steps of PCA

1. First we read all the training images
2. Convert each [m x n] image matrix into [(m x n) x 1] image vector.(Flatten image matrix to vector).
3. Find average_face_vector, sum(all image vectors)/number(images).
4. Subtract average_face_vector from every image vector.
5. Stackall (image_vectors—average_face_vector) in matrix forming A = [(m x n) x i] matrix (where i = number of images).
6. Stack all (image_vectors—average_face_vector) in matrix forming A = [(m x n) x i] matrix (where i = number of images).
7. Calculate covariance matrix of above matrix -> C = A*transpose(A).
8. Find eigenvectors and eigenvalues of above covariance matrix.
9. As size of above matrix is huge its very expensive to compute eigenvectors of such a huge matrix,so we will calculate the eigenvectors of transpose(A)*A as the matrix is small of i x i dimension.
10. Then we will pre-multiply the eigenvectors of transpose(A)*A with A to get eigenvectors of A*transpose(A).

11. Choose best k eigenvectors such that k << i.
12. Find weights of each image and store it.

Face detection using HAAR

After feature extraction of the human face is completed, face detection is done to detect the face from the dataset and label it. The available approaches for face detection are: (1) Skin detection (2) Viola Jones with HAAR (3) Viola Jones with LBP.
Skin detection is not the most viable method as the image dataset is converted to gray scale mode in the data preprocessing stages. Both LBP and the HAAR based approaches use an underlying Viola-Jones framework. The LBP approach approximates features (pixel quantization) and this results in the HAAR approach producing more accurate results. The Viola-Jones approach offers real-time performance and scale/location invariance, but it still has a few disadvantages - intolerance to object rotations, sensitivity to illumination variations, etc. The comparison of the approaches showed the **HAAR method** was the most appropriate to use for detecting each individual's face.

Current approach of PCA that we have implemented, uses HAAR method for face detection.

IMPLEMENTATION

RESULTS

1. All the images are converted to square shaped images (NxN matrix).

2. RGB image(3456x3456) converted to grayscale image(3456x3456).

3. Code for feature extraction is implemented in Python.

4. This approach aspires to gain an accuracy of approximately 93%.

CONCLUSIONS

1. Gray-scale images are preferred over RGB color images in the application of face recognition.
2. HAAR is the most suitable method for face detection for a non Neural Network based approach.
3. As the current dataset(2.5 GB) contains much noise and outliers, the **combination of Probabilistic PCA(with Bayesian inference) and LDA approach** aims to result in the maximum expected accuracy.

REFERENCES

[1]https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

[2]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.146.5125&rep=rep1&type=pdf

Faisal R. Al-Osaimi, Mohammed Bennamoun, and Ajmal Mian, "Illumination Normalization for Color Face Images", The University of Western Australia 35 Stirling Highway, Crawley, WA 6009, Australia

[3]https://www.researchgate.net/publication/224173583_Illumination_normalization_preprocessing_for_face_recognition

Muhammad Sharif, Sajjad Mohsin, Muhammad Jawad Jamal, Mudassar Raz, "Illumination Normalization Preprocessing for face recognition", 2nd Conference on Environmental Science Information Application technology, 2010

[4]https://www.researchgate.net/publication/303851281_Image_preprocessing_for_appearance-based_face_recognition

César Fernández, M. Asunción Vicente, Ramón P. Ñeco, Rafael Puerto, "Image Preprocessing for Appearance-based Face Recognition", Miguel Hernández University, Av. de la Universidad s/n, 03202 Elche, Spain

[5]https://www.researchgate.net/publication/281838833_FACE_RECOGNITION_TECHNIQUES_AND_APPROACHES_A_SURVEY

Muhammad Naeem, Imran Qureshi, and Faisal Azam, “FACE RECOGNITION TECHNIQUES AND APPROACHES: A SURVEY”, Department of Computer Science, COMSATS Institute of Information Technology, Wah Cant, Pakistan, Sci.Int.(Lahore),27(1),301-305,2015 ISSN 1013-5316; CODEN: SINTE 8

[6]https://www.giassa.net/?page_id=472

Prata, Stephen. *C Primer Plus (5th Edition)*. Indianapolis: Sams, 2004. Print.

[7]https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6077989

SHU Chang (舒 畅), DING Xiaoqing (丁晓青)**, FANG Chi (方 驰), “Histogram of the Oriented Gradient for Face Recognition”, TSINGHUA SCIENCE AND TECHNOLOGY, ISSNll1007-0214ll15/15llpp216-224 Volume 16, Number 2, April 2011

[8]http://sklin93.github.io/hog.html

[9]https://pdfs.semanticscholar.org/7fc5/ab3743e6e9a2f4fe70152440e13a673e239b.pdf

Harihara Santosh Dadi, Gopala Krishna Mohan Pillutla , “Improved Face Recognition Rate Using HOG Features and SVM Classifier”, IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 11, Issue 4, Ver. I (Jul.-Aug .2016), PP 34-44

[10]https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html

[11]https://www.intechopen.com/books/reviews-refinements-and-new-ideas-in-face-recognition/dimensionality-reduction-techniques-for-face-recognition

[12] M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience,vol. 3, no. 1, pp. 71-86, 1991, hard copy

Rough work/ Extra:

***(Remove this whole part)//1.2 Feature Extraction using PCA and KLT

Eye features are identified relative to the position of the mouth, by searching for regions which satisfy some statistical, geometrical, and structural properties of the eyes in frontal face images. On detecting a feature set containing a mouth and two eyes, PCA analysis is performed over a normalized search space relative to the distance between the two eyes. In test image, one image is acquired as input and mean, covariance and Eigen values are calculated. It is been normalized as feature and it is been compared to the features in training section. If features are matched, equivalent image is been found. If not matched, no image is detected.

Starting with preprocessing, HDTV method is used to convert images from RGB to GRAYSCALE.

```
from PIL import Image
im = Image.open('16BCP064_NEUTRAL_RBG.png')
pix = im.load()
print pix[x,y]  # Get the RGBA Value of the a pixel of an image
```

Taking individual rgb values of each pixel and applying the HDTV formula for efficient conversion to gray-scale.

Feature Extraction using PCA uses **Karhunen–Loève transform (KLT)** to reduce the dimensionality of the dataset.

Consider X as the dataset with M dimensions which are required to be reduced to L,

Y= KLT(X).

Next step is to rearrange the 2D image matrix of dimensions m x n to the form 1D matrix.

Write X1...Xn as column vectors, each of which has M rows.

Place the column vectors into a single matrix X of dimensions M × N.

Calculate the empirical mean,

$$u[m] = \frac{1}{N}\sum_{n=1}^{N} X[m,n]$$

Calculate the deviations of input data from the mean. This helps towards finding a principal component basis that minimizes the mean square error of approximating the data.
Subtract the empirical mean vector from each column of the data matrix X.
Calculate the covariance matrix and its eigenvalues and eigenvectors.

The image detection phase is executed using the HAAR method, which uses **openCV - HAAR cascade detection** in openCV.
OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc. Those XML files are stored in the opencv/data/haarcascades/ folder. Let's create a face and eye detector with OpenCV.
First the required XML classifiers are loaded. Then load the input image (or video) in grayscale mode as done using the HDTV method.

```
import numpy as np
import cv2 as cv
face_cascade=
        cv.CascadeClassifier('haarcascade_frontalface_d
        efault.xml')
eye_cascade                                          =
        cv.CascadeClassifier('haarcascade_eye.xml')
img = cv.imread("16BCP064_NEUTRAL_RBG.png")
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

Find the faces in the image, if the faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once these locations are extracted, a ROI(Region of Interest) is created for the face and eye detection is applied on this ROI (since eyes are always on the face !!!).

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
        cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in eyes:

        cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,
        255,0),2)
cv.imshow('img',img)
cv.waitKey(0)
cv.destroyAllWindows()
```

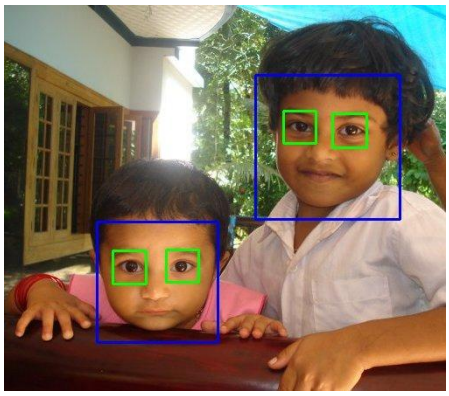

FIGURE 2: RESULT OF A EXAMPLE DATA SAMPLE