Minor Project Report

Face Detection and Recognition using Back Propagation Neural Network

Under the guidance of
Mr. Sanjay Sharma

Submitted By

Anoop Kumar Singh (14bcs008)
Chitranshu Yadav (14bcs015)
Sangam Kumar (14bcs045)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHRI MATA VAISHNO DEVI UNIVERSITY
KATRA-182320

ACKNOWLEDGEMENT

“The satisfaction and Euphoria that accompanies the successful completion of anyproject would be incomplete without the mentioning of the people who made it possible and whose constant guidance and encouragement served as a beacon of light and crowned the efforts with success”.

We wish to express our sincere gratitude to Dr. Baij Nath Kaushik, H.O.D of Computer Science & Engineering Department for providing us an opportunity to do our project work on this topic.

We express our deepest and most sincere gratitude to Mr. Sanjay Sharma, our Guide whose invaluable suggestions, comments and guidance during the course of this project, without which this project would not have been successful.

S.No.	Entry No.	Name	Signature
1.	14bcs008	Anoop Kumar Singh	
2.	14bcs015	Chitranshu Yadav	
3.	14bcs045	Sangam Kumar	

CERTIFICATE

This is to certify that we, Anoop Kumar Singh, Chitranshu Yadav, Sangam Kumar (14bcs008, 14bcs015, 14bcs045) have worked together under the guidance of Mr. Sanjay Sharma on the project titled as **"Face Detection and Recognition using Back Propagation Neural Network"** in the School of Computer Science & Engineering, College of Engineering, Shri Mata Vaishno Devi University, Kakryal, Jammu & Kashmir, dated from 1th August 2017 to 27th November 2017 for the award of **Bachelor of Technology** in Computer Science & Engineering.

The contents of this project, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Student's Signature

Student' name

This is to certify that the above student has worked for the project titled **"Face Detection and Recognition using Back Propagation Neural Network"** under my supervision.

Contents:

List of Figures:

Chapter 1

Abstract:

Detection and recognition of the facial images of people is an intricate problem which has garnered much attention during recent years due to its ever increasing applications in numerous fields. It continues to pose a challenge in finding a robust solution to it. Its scope extends to catering the security, commercial and law enforcement applications. Research for moreover a decade on this subject has brought about remarkable development with the modus operandi like human computer interaction, biometric analysis and content based coding of images, videos and surveillance. A trivial task for brain but cumbersome to be imitated artificially. The commonalities in faces does pose a problem on various grounds but features such as skin color, gender differentiate a person from the other.

1. Introduction:

Face recognition is very important for our daily life. It can be used for remote identification services for security in areas such as banking, transportation, law enforcement, and electrical industries, etc. For this security access project is aimed at demonstrating facial recognition techniques that could antiquate, substitute, or otherwise, supplement, conventional key, and can be used as an alternative to existing fingerprint biometrics method. A computerized system equipped with a digital camera can identify the face of a person and determine if the person is authorized to start the vehicle. This integrated system would be able to authorize a user before switching on the vehicle with a key. Whilst facial recognition systems are by now readily available in the market, the vast majority of them are installed at large open spaces, such as in airport halls. The focus of this project is, thus, to compare the extracted feature with face image database for the recognition analysis using Neural Network. Artificial neural network (ANN) is a complicated, able-to-learn, nonlinear and dynamic system. It is very useful for pattern recognition such as face recognition to increase identification accuracy and robust of the recognition system.

Facial recognition is one of classical applications of the Artificial Neural Network. This recognition system use neural network approach to recognize the image according to the neural network, this project use BackPropagation network. The BackPropagation learning is a technique discovered by Rumelhart, Hinto, and Williams in 1986 and it is a supervised learning that learns by propagating the signals through the network, computing the input and output using a feedforward network, then calculates the error values and propagates the error back through the network to adapt the weight during training.

2. Background work and related concepts:

2.1 Study of Existing system :

Face recognition is a very active area in computer vision and biometrics fields and it has been studied rigorously since last 25 years. The earlier known methods for face recognition are feature matching and template matching. Various algorithms have been developed for face recognition. Most of them capture the image using a capturing device of high quality like a webcam or digital cam. The Eigen weight matrix is extracted from each of the available images and stored in the database. The weight matrix is also similarly obtained from the image to be identified using the same mean image as obtained in the training phase. These weight matrixes are compared to get a match that is to check whether the image exist in the database or not. Thus using this project it provides a very friendly environment for both operator and eyewitnesses to easily design any face can identify criminals very easy. This project is intended to identify a person using the images previously taken, or taken at the very moment. The scope of the project is confined to store the image and store in the database. When a person has to be identified the images stored in the database are compared with the existing details.

Drawbacks in existing systems

1.) Time consuming

2.) Accuracy of identification is less

But in the case of proposed system we are extracting a pattern and create neural network files other than the complete image.

2.2 Viola Jones Method:

The Viola–Jones object detection framework is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. This algorithm is implemented in OpenCV as cvHaarDetectObjects().

This approach to detecting objects in images combines four key concepts:

i) The features that Viola and Jones used are based on Haar wavelets. Haar wavelets are single wavelength square waves (one high interval and one low interval). In two dimensions, a square wave is a pair of adjacent rectangles - one light and one dark. The actual rectangle combinations used for visual object detection are not true Haar wavlets. Instead, they contain rectangle combinations better suited to visual recognition tasks.Because of that difference, these features are called Haar features, or Haarlike features,rather than Haar wavelets. The presence of a Haar feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present.

ii) To determine the presence or absence of hundreds of Haar features at every image location and at several scales efficiently, Viola and Jones used a technique called an Integral Image. In general, "integrating" means adding small units together. In this case, the small units are pixel values. The integral value for each pixel is the sum of all the pixels above it and to its left. Starting at the top left and traversing to the right and down,the entire image can be integrated with a few integer operations per pixel.

iii) To select the specific Haar features to use, and to set threshold levels, Viola and Jones use a machine-learning method called AdaBoost. AdaBoost combines many "weak" classifiers to create one "strong" classifier. "Weak" here means the classifier only gets the right answer a little more often than random guessing would. That's not very good. But if you had a whole lot of these weak classifiers, and each one "pushed" the final answer a little bit in the right direction, you'd have a strong, combined force for arriving at the correct solution. AdaBoost selects a set of weak classifiers to combine and assigns a weight to each. This weighted combination is the strong classifier.

iv) Viola and Jones combined a series of AdaBoost classifiers as a filter chain that's especially efficient for classifying image regions. Each filter is a separate AdaBoost classifier with a fairly small number of weak classifiers. The acceptance threshold at each level is set low enough to pass all, or nearly all, face examples in the training set. The filters at each level are trained to classify training images that passed all previous stages. (The training set is a large database of faces, maybe a thousand or so.) During use, if any one of these filters fails to pass an image region, that region is immediately classified as "Not Face." When a filter passes an image region, it goes to the next filter in the chain. Image regions that pass through all filters in the chain are classified as "Face." Viola and Jones dubbed this filtering chain a cascade.

2.3 Local Binary Patterns Histograms:

Local Binary Patterns Histograms,or LBPH in short,also needs to be trained on hundreds of images LBPH is a visual/texture descriptor, and thankfully, our faces are also composed of micro visual patterns.So, LBPH features are extracted to form a feature vector that classifies a face from a non-face.

"But how are LBPH features found?"

Each training image is divided into some blocks as shown in the picture below.

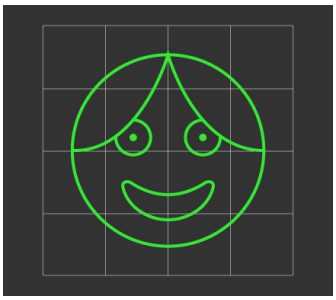
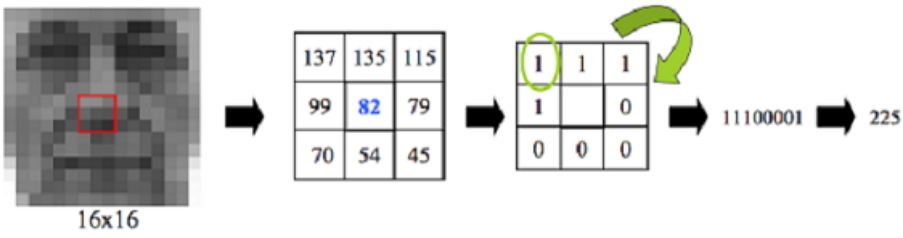


fig:1

For each block,**LBPH looks at 9 pixels** (3×3 window) at a time, and with a particular interest in the pixel located in the center of the window.

Then, it compares the central pixel value with every neighbor's pixel value under the 3×3 window. For each neighbor pixel that is greater than or equal to the center pixel, it sets its value to 1, and for the others, it sets them to 0.

After that, it reads the updated pixel values (which can be either 0 or 1) in a clockwise order and forms a binary number. Next, it converts the binary number into a decimal number, and that decimal number is the new value of the center pixel. We do this for every pixel in a block.



LBPH conversion to binary (Fig:2)

Then, it converts each block values into a **histogram,** so now we have gotten one histogram for each block in an image, like this:-

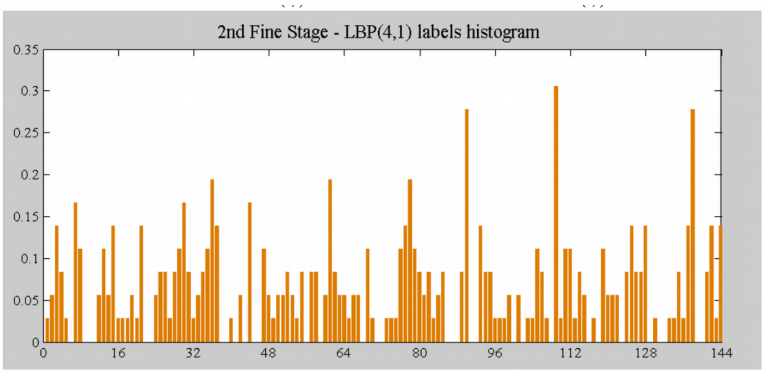


Fig:3

Finally, it concatenates these block histograms to form a one feature vector for one image, which contains all the features we are interested. So, this is how we extract LBPH features from a picture.

2.4 Haar Cascades:

The Haar classifier is a machine learning based approach, an algorithm created by Paul Viola and Michael Jones; which (as mentioned before) are trained from many many positive images (with faces) and negatives images (without faces).

It starts by extracting Haar features from each image as shown by the windows below:

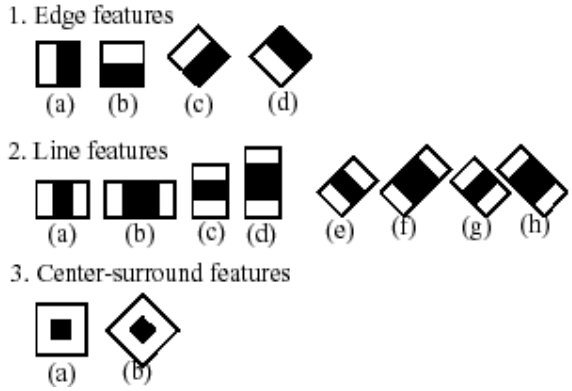
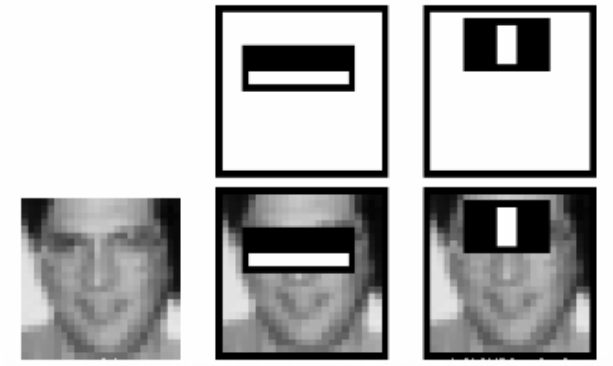


Fig:4

Each window is placed on the picture to calculate a single feature. This feature is a single value obtained by subtracting the sum of pixels under the white part of the window from the sum of the pixels under the black part of the window.

Now, all possible sizes of each window are placed on all possible locations of each image to calculate plenty of features.



Different stages in visualization **(Fig:5)**

For example, in above image, we are extracting two features. The first one focuses on the property that the region of the eyes is often darker than the area of the nose and cheeks. The second feature relies on the property that the eyes are darker than the bridge of the nose.

But among all these features calculated, most of them are irrelevant. For example, when used on the cheek, the windows become irrelevant because none of these areas are darker or lighter than other regions on the cheeks, all sectors here are the same.

So we promptly discard irrelevant features and keep only those relevant with a fancy technique called **Adaboost.**AdaBoost is a training process for face detection, which selects only those features known to improve the classification (face/non-face) accuracy of our classifier.

In the end, the algorithm considers the fact that generally: most of the region in an image is a non-face region. Considering this, it’s a better idea to have a simple method to check if a window is a non-face region, and if it's not, discard it right away and don’t process it again. So we can focus mostly on the area where a face is.

2.5 Artificial Neural Network:

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

2.6 Back Propagation:

Backpropagation is a common method of training artificial neural networks so as to minimize the objective function. It is a supervised learning method, and is a generalization of the delta rule. It requires a dataset of the desired output for many inputs, making up the training set. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backward propagation of errors".Backpropagation requires that the activation function used by the artificial neurons (or "nodes")be differentiable.

2.7 OpenCV:

OpenCV means Intel® Open Source Computer Vision Library. It is a collection of C functions and a few C++ classes that implement some popular Image Processing and Computer Vision algorithms. The OpenCV is an open source computer vision library.

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel and now supported by Willow Garage.The library is OS/hardware/window manager independent.

The library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Matlab, and other languages. OpenCV was designed for computational efficiency and with a strong focus on real time applications. Open CV is written in optimized C and can take advantage of multi core processors. If you desire further automatic optimization on Intel architectures you can buy Intel's Integrated Performance Primitives (IPP) libraries, which consist of low-level optimized routines in many different algorithmic areas. OpenCV automatically uses the appropriate IPP library at runtime if that library is installed.

Features of OpenCV

- Image data manipulation (allocation, release, copying, setting, conversion).
- Image and video I/O (file and camera based input, image/video file output).
- Matrix and vector manipulation and linear algebra routines (products, solvers,eigenvalues, SVD).
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
- Motion analysis (optical flow, motion segmentation, tracking).
- Object recognition (Eigen-methods, HMM).
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
- Image labeling (line, conic, polygon, text drawing)

3. Design Principles:

Modules

A module is a small part of our project. This plays a very important role in the project and in coding concepts. Importance of modules in any software development side is we can easily understand what the system we are developing and what its main uses are.

The complete Face Recognition system can be divided into the following modules:

- Face Detection
- Data acquisition
- Image preprocessing
- Face Recognition

Module Description

3.1 Face Detection:

Face detection is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies.

Early face-detection algorithms focused on the detection of frontal human faces, whereas newer algorithms attempt to solve the more general and difficult problem of multi-view face detection.That is, the detection of faces that are either rotated along the axis from the face to the observer (in-plane rotation), or rotated along the vertical or left-right axis (out-of-plane rotation), or both.The newer algorithms take into account variations in the image or video by factors such as face appearance, lighting, and pose.

The sub Modules are:

- Input of frame from the Camera or Images
- Detection of Face within the given frame

3.1.1 Input of frame from Camera or Images:

It is very easy to use a webcam stream as input to the face recognition system instead of a file list. Basically it just needs to input frames from a camera or image from a file.

Understanding the code

```
# Get user supplied value
 imagePath = sys.argv[1]
 cascPath = sys.argv[2]
```

You first pass in the image and cascade names as command-line arguments. We'll use the image as well as the default cascade for detecting faces provided by OpenCV.

```
# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)
```

Now we create the cascade and initialize it with our face cascade. This loads the face cascade into memory so it's ready for use. Remember, the cascade is just an XML file that contains the data to detect faces.

```
# Read the image
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Here we read the image and convert it to grayscale. Many operations in OpenCv are done in grayscale.

```
# input faces via your webcam.
cap = cv2.VideoCapture(0)
```

OpenCv provides the cv2VideoCapture() also known as cvCaptureFromCAM() for this. It finds any camera or cam on the system, accesses it and then uses it to capture a stream of frames.

3.1.2 Detection of face within a given frame:

OpenCV uses a type of face detector called a Haar Cascade classifier. Given an image, which can come from a file or from live video, the face detector examines each image location and classifies it as "Face" or "Not Face." Classification assumes a fixed scale for the face, say 50x50 pixels. Since faces in an image might be smaller or larger than this, the classifier runs over the image several times, to search for faces across a range of scales.

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
 gray,
scaleFactor=1.1,
minNeighbors=5,
minSize=(30, 30),
```

```
flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

This function detects the actual face – and is the key part of our code, so let's go over the options.

1.) The detectMultiScale function is a general function that detects objects. Since we are calling it on the face cascade, that's what it detects. The first option is the grayscale image.

2.) The second is the scaleFactor. Since some faces may be closer to the camera, they would appear bigger than those faces in the back. The scale factor compensates for this.

3.) The detection algorithm uses a moving window to detect objects.minNeighbors defines how many objects are detected near the current one before it declares the face found.minSize, meanwhile, gives the size of each window.

For testing use below script:

```
$ python face_detect.py abc.png haarcascade_frontalface_default.xml
```

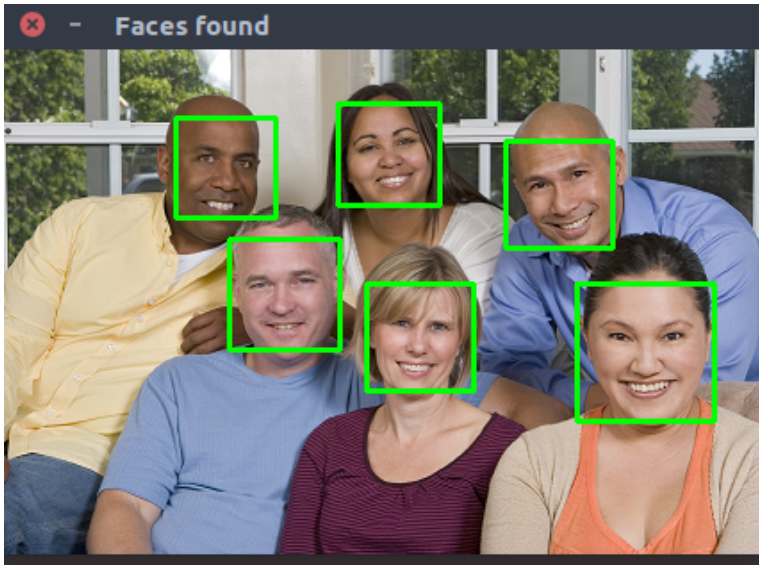


Fig:6

3.2 Data Acquisition:

Once a face has been detected in an image we need to capture the face region so as to do further processing of image for recognition purposes. This is setting of region of interest. A Region of Interest (ROI) is a portion of an image that we want to filter or perform some other operation on. We take the image consisting of the face as input image and set a rectangle ROI on the face area.Function used for this in OpenCv is:

cvSetImageROI(img, cv2Rectangle(r->x,r->y,r->width,r->height));

3.2.1 Capturing multiple frames:

Since we want our system to be illumination, rotation invariant and the face may not be upright frontal, so we need a number of different images in various conditions for better training. For this purpose we need to capture multiple continuous frames in different poses. Capturing a single frame and saving it to disk is a bit easier task but capturing multiple involves naming each image to a unique name. This is done by adding an identifier i.e frame number or image id to an image inside a loop.

3.3 Image preprocessing:

In this module, by means of early vision techniques, face images are normalized and if desired, they are enhanced to improve the recognition performance of the system. Image preprocessing is a significant step before applying any other technique on images. Image preprocessing can be of different kinds, for noise removal, smoothing thresholding, background removal etc., it can be rightly said that the kind of preprocessing required is greatly dependent on the application under consideration. The data that has been acquired may not be suitable for applying further processing. It consists of many noisy and redundant data. So this data needs to be preprocessed.Preprocessing is done to improve the efficiency, accuracy and robustness of the proposed system. It also reduces the complexity of the system by removing redundant data.

Some or all of the following pre-processing steps may be implemented in a face recognition system:

3.3.1 Grayscale conversion:

In many applications involving face recognition, the inputs are first binarized to form a two level image based on the threshold value. It is usual to preprocess the scanned images to do the image enhancement in the presence of noise and other types of distortions that occur during the scanning process. In this preprocessing stage, the image is enhanced, resized and binarized to make the image clear and accurate. It is necessary to employ the non linear technique for processing the face images prior to binarization.

There are several methods available for thresholding image to produce binary image. An experimental performance evaluation of several such techniques may found in. These methods include fixed global threshold, Otsu threshold and other techniques. The input grayscale pixels are denoted by Xi € {0, 1}. The corresponding output binarization pixels are denoted by bi € {0,1} where 0 refers to „black' and 1 refers to 'white’.

Image enhancement is necessary to remove noise by filtering the image, adjust the contrast of the image and enhance the edge of the face image before binarization process and training the facial recognition. Edge enhancement can be achieved by noise presents in the input images but tend to smooth the edge details.

In this project, the global image threshold using Otsu's method will be chosen as the technique to binarize the face images. This method finds the global threshold t that minimizes the infraclass variance of the resulting black and white pixels of the image.
The threshold of this method can be shown as:

$$b_i = 1 \text{ if } x_i >= t \text{ or } b_i = 0 \text{ if } x_i < t$$

where

x_i = input grayscale pixel

b_i = output binarization pixel

OpenCv function: cv2Color(img, g_img, CV_BGR2GRAY);

3.3.2 Resizing:

It is usually done to change the acquired image size to a default image size such as 128 x 128, on which the face recognition system operates. This is mostly encountered in systems where face images are treated as a whole like the one proposed in this thesis.

3.3.3 Histogram equalization:

It is usually done on too dark or too bright images in order to enhance image quality and to improve face recognition performance. It modifies the dynamic range (contrast range) of the image and as a result, some important facial features become more apparent.

3.3.4 Median filtering:

For noisy images especially obtained from a camera or from a frame grabber, median filtering can clean the image without losing information.

3.3.5 High-pass filtering:

Feature extractors that are based on facial outlines, may benefit the results that are obtained from an edge detection scheme. High-pass filtering emphasizes the details of an image such as contours which can dramatically improve edge detection performance.

3.3.6 Background removal:

In order to deal primarily with facial information itself, face background can be removed. This is especially important for face recognition systems where entire information contained in the image is used. It is obvious that, for background removal, the preprocessing module should be capable of determining the face outline.

3.3.7 Illumination normalization:

Face images taken under different illuminations can degrade recognition performance especially for face recognition systems based on the principal component analysis in which entire face information is used for recognition. A picture can be equivalently viewed as an array of reflectivity's r(x). Thus, under a uniform illumination I, the corresponding picture is given by:

$$\Phi(x)=Ir(x)$$

The normalization comes in imposing a fixed level of illumination I_0 at a reference point x_0 on a picture. The normalized picture is given by:

$$\Phi(x) = I_0 * \Phi(x_0)/I(x_0)$$

In actual practice, the average of two reference points, such as one under each eye, each consisting of 2 x 2 arrays of pixels can be used.

3.4 Face recognition:

After the preprocessing and feature extraction phase the most important phase of this project is the face recognition. Face recognition aims at giving a name to the faces which has been obtained till now. It is the actual part of access control. We have implemented face recognition using neural network classification. Neural network classification is better than other methods of classification because it gives a non linear classification, able to learn and incorporate changes easily.

3.4.1 Network Topology:

The network topology is defined as: how many layers and how many neurons per layer are used

Number of Layers

The number of layers include 1 input layer, 1 output layer and one more than one hidden layers.There is actually no rule for choosing the number of layers but In MLPs with any of a wide variety of continuous nonlinear hidden-layer activation functions, one hidden layer with an arbitrarily large number of units suffices for the "universal approximation"

Number of Nodes

The number of nodes in the input layer depends on the number of eigenvectors produced after the feature extraction. Each eigenvector is an independent attribute and is fed into one of the nodes of the input layer.

3.5.2 Training parameters:

Learning Rate

It is a control parameter of some training algorithms, which controls the step size when weights are iteratively adjusted. It affects the speed of learning. It will apply a smaller or larger proportion of current adjustment to the previous weight. The higher the rate is set, the faster the network will learn, but if there is large variability in the input the network will not learn very well if at all.

Initial weights

This initial weight will influence whether the net reaches a global or local minima of the error and if so how rapidly it converges. To get the best results the initial weights are set to random numbers between 0 and 1.

Training Set

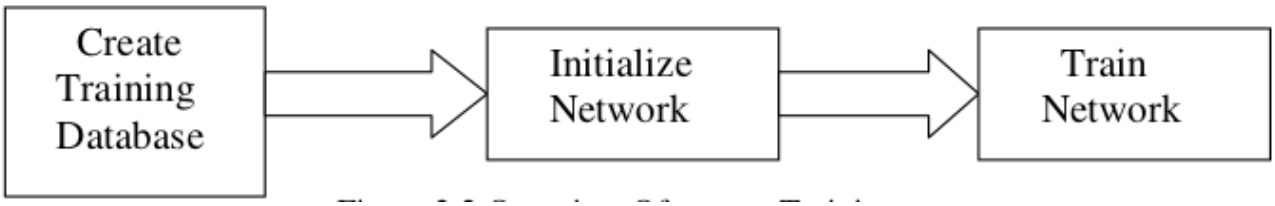
It is the set of data which is used for the training of the Multi Layer Perceptron model. A good training set should have

- Samples must represent the general population.
- Samples must contain members of each class.
- Samples in each class must contain a wide range of variations or noise effect.

3.5.3 Classifier Training :

3.5.3.1 Training and Learning:

The decision functions can be generated in a variety of ways. When a complete prior knowledge about the patterns to be recognized is available, the decision function may be determined with precision on the basis of this information. When only qualitative knowledge about the patterns is available, reasonable guesses of the forms of the decision functions can be made. In this case the decision boundaries may be far from correct, and it is necessary to design the machine to achieve satisfactory performance through a sequence of adjustments.



Overview Of system Training

fig:6

3.4.3.2 Training Algorithm:

BackPropagation Algorithm

BackPropagation algorithm is a neural-network learning algorithm. It is a supervised learning technique. A Back Propagation network learns by example. We give the algorithm examples of what we want the network to do and it changes the network's weights so that, when training is finished, it will gives the required output for a particular input. Back Propagation networks are ideal for simple Pattern Recognition and Mapping tasks.

The network is first initialised by setting up all its weights to be small random numbers – say between –1 and +1. Next, the input pattern is applied and the output calculated (this is called the

forward pass). The calculation gives an output which is completely different to what user want (the Target), since all the weights are random. We then calculate the error of each neuron, which is essentially: Target - Actual Output (i.e. what we want – What we actually get). This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the Output of each neuron will get closer to its Target (this part is called the reverse pass). The process is repeated again and again until the error is minimal.

Algorithm:

- First apply the inputs to the network and work out the output – remember this initial output could be anything, as the initial weights were random numbers.
- Next work out the error for neuron B. The error is what we want to get and what we actually get, in other words:
- Error = $\text{Output}_B (1 - \text{Output}_B) (\text{Target}_B - \text{Output}_B)$
- The “Output (1 - Output)” term is necessary in the equation because of the Sigmoid Function – if we were only using a threshold neuron it would just be (Target - Output).
- Change the weight. Let $W+_{AB}$ be the new (trained) weight and W_{AB} be the initial weight.

  $W+_{AB} = W_{AB} + (\text{Error}_B * \text{Output}_A)$
- Notice that it is the output of the connecting neuron (neuron A) we use (not B). We update all the weights in the output layer in this way.
- Calculate the Errors for the hidden layer neurons. Unlike the output layer we can’t calculate these directly (because we don’t have a Target), so we Back Propagate them from the output layer (hence the name of the algorithm). This is done by taking the errors from the output neurons and running them back through the weights to get the hidden layer errors. For example if neuron A is connected as shown to B and C then we take the errors from B and C to generate an error for A.
- $\text{Error}_A = \text{Output}_A(1 - \text{Output}_A) (\text{Error}_B W_{AB} + \text{Error}_c W_{AC})$
- Again, the factor “Output (1 - Output)” is present because of the sigmoid squashing function.
- Having obtained the Error for the hidden layer neurons now proceed as in stage 3 to change the hidden layer weights. By repeating this method we can train a network of any number of layers

4. Result:

4.1 Snapshots:

4.1.1 Face detection: Using Image Set

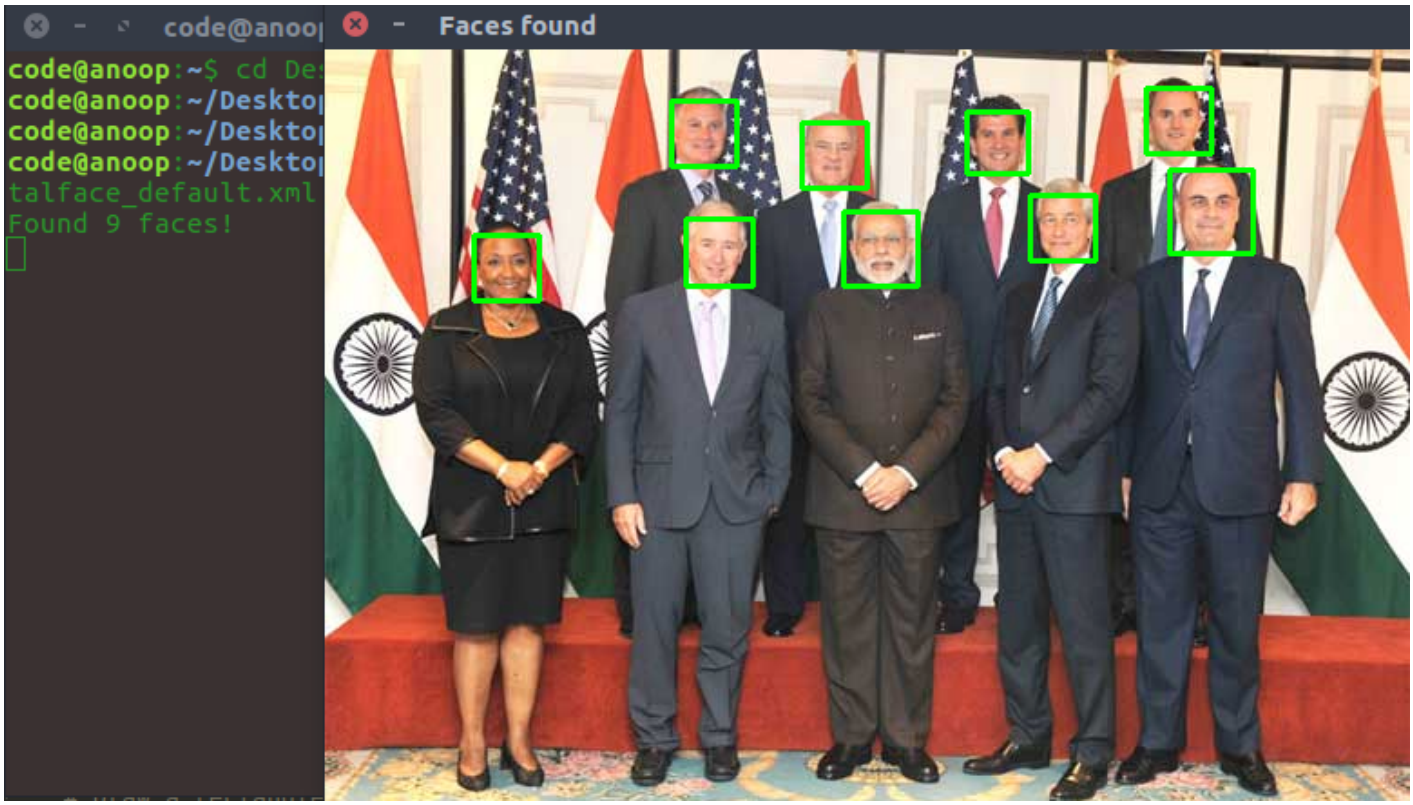


Fig:7

Face detection: Through Cam

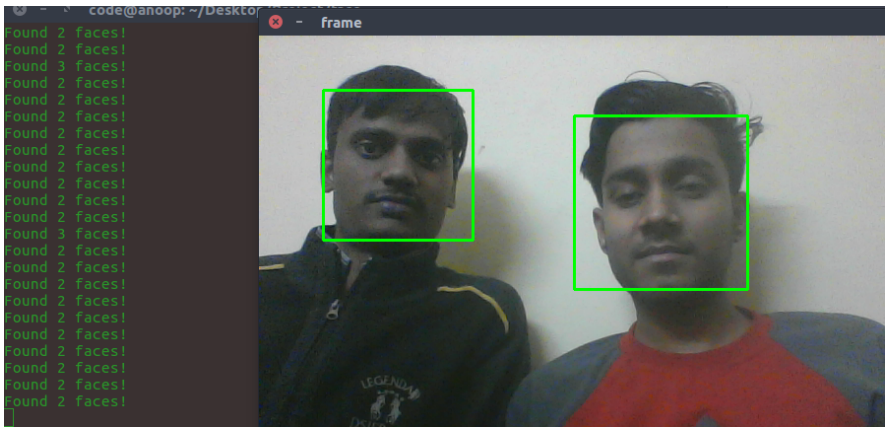


Fig:8

4.1.2 Face recognition:

Fig:9

Back Propagation : For a data Set

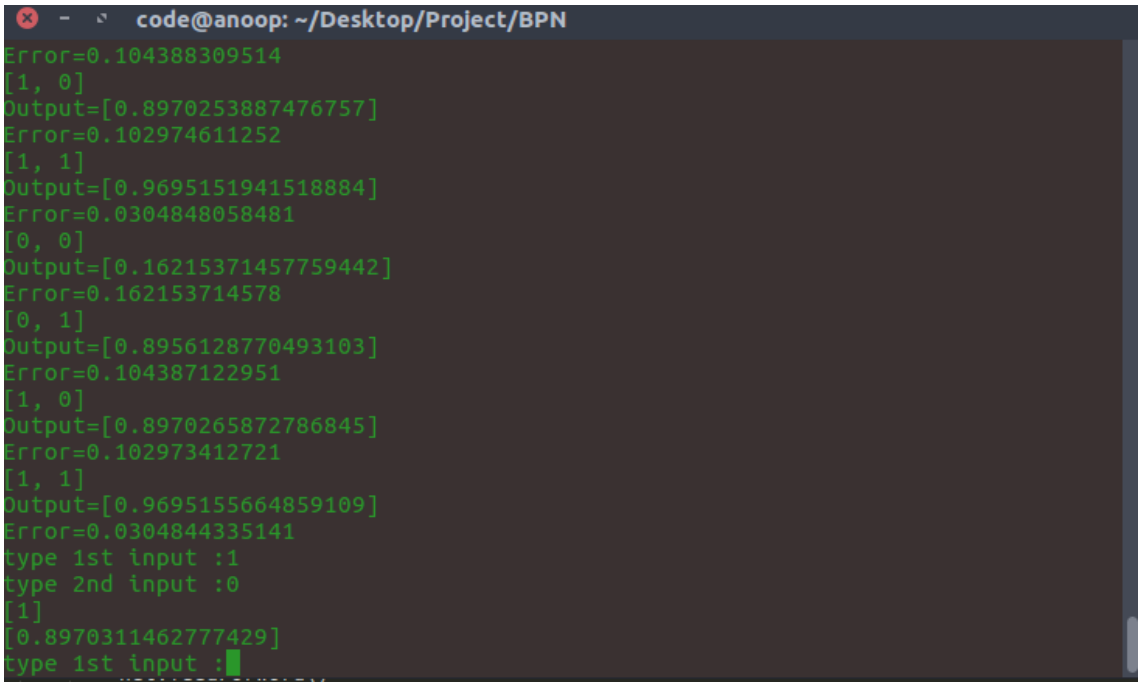


Fig:10

5. Conclusions and Future Work:

In this research project we have used “Haar-cascade” Detection in OpenCV through python and also have implemented a face recognition system in OpenCV through “Local Binary Patterns Histograms”, also implemented a simple neural network through Back Propagation method for a data set.

Future Work:

1.) Recognition through back propagation neural network.

2.) Implementing face detection technique to facilitate real time feature extraction

3.) Using different classifier like Support Vector Machines (SVM).

5.1 Applications:

- Security/Counter terrorism: Access control, comparing surveillance images to Known
- Correctional institutions/prisons: Inmate tracking, employee access.
- Day Care: Verify identity of individuals picking up the children.
- Gaming Industry: Find card counters and thieves.
- Internet, E-commerce: Verify identity for Internet purchases

References:

- https://docs.opencv.org/
- https://www.superdatascience.com/
- https://www.ijcsi.org/papers/IJCSI-9-6-1-169-172.pdf
- https://github.com/