FACE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

A report submitted for the course named Project II (CS-300)

By

Shweta Kumari
Bachelor of Technology, VI Semester
Roll No. 15010103.

Department of Computer Science and Engineering
Indian Institute of Information Technology Manipur
April, 2017

Abstract

Facial recognition technology has emerged as an attractive solution to address many contemporary needs for identification and the verification of identity claims. The identification and verification are the two vast field where face recognition is used. The task of Identification is to identify the identity of the person based on database stored of that person. In this project the face identification part has been completed.

Keywords - Principal Component Analysis,Eigenfaces,Neural Network, Convolutional Neural Network

Declaration

I declare that this submission represents my idea in my own words and where others' idea or words have been included, I have adequately cited and referenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/sources in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from proper permission has not been taken when needed.

(Signature)

(Shweta Kumari)

Date: (15010103)

Department of Computer Science & Engineering
Indian Institute of Information Technology Manipur

Dr. Thoudam Doren Singh
Assistant Professor

Email: doren@iiitmanipur.ac.in

To Whom It May Concern

This is to certify that the report entitled **"Face Recognition Using Convolutional Neural Network"** submitted by "Shweta Kumari", has been carried out under my supervision and that this work has not been submitted elsewhere for a degree,diploma or a course.

Signature of Supervisor

(Dr. Thoudam Doren Singh)

Acknowledgement

I extend my sincere thanks to the institute who has provided me a chance to do the project. I would like to express deep debt to Dr.Thoudam Doren Singh, project guide for his vital suggestion, meticulous guidance and constant motivation which went a long way in succesful completion of the project. I cannot move on without thanking to Dr.Navnath Saharia who has always been an inspiration to me. I would like to thank to project coordinator who has been organising the project presentation time to time which helped me to finish the project on time. On a moral personal note my deepest appreciation and gratitude to my beloved friends, who has been an inspiration and have provided me a unrelenting encouragement and support.

- Shweta Kumari

Contents

List of Figures

List of abbreviations

		P
PCA	Principal Component Analysis	

		C
CNN	Convolutional Neural Network	

Chapter 1

Introduction

The proposed project is face recognition using convolutional neural network. Face plays a major role in identity of the person. Face recognition is an important embodiment of human-computer interaction, which has been widely used in access control system, monitoring system and identity verification. The concept of face recognition was proposed in 1960. Human being can recognize thousands of faces learned through our lifetime and can identify them even after years of separation. In the same way the proposed system will be able to recognize the faces.

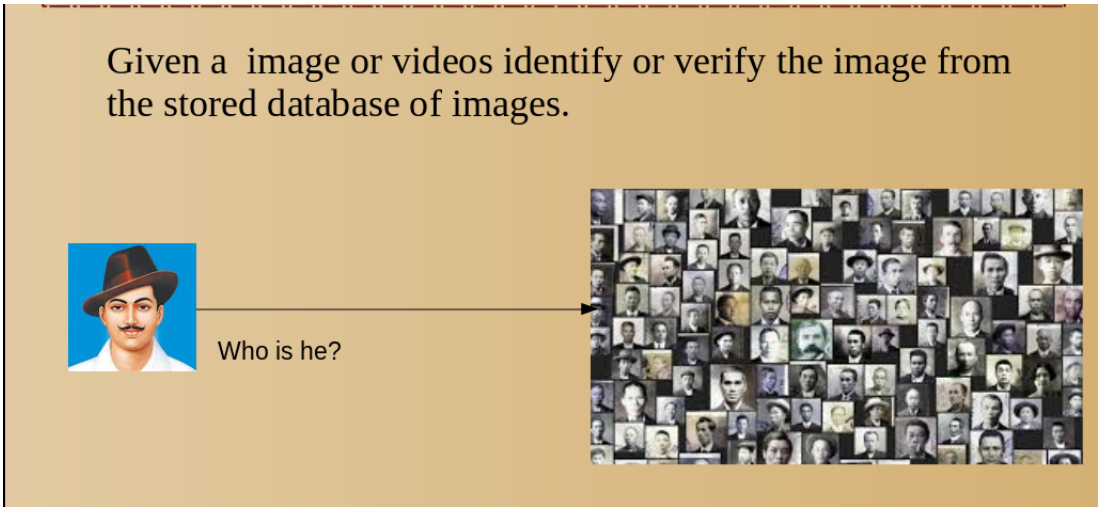

Figure 1.1: Problem Statement

The above figure describes the problem statement. Given a images the propesed system will identify the identity of a person from the images stored in the database.

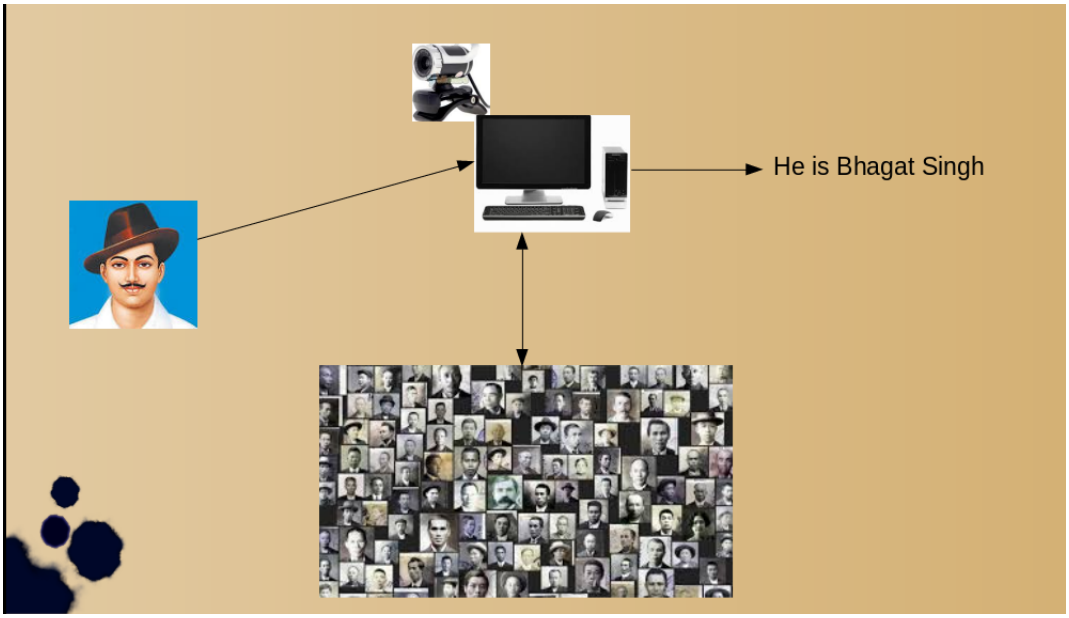

Figure 1.2: Image Identification

The above figure explains the identification process. The image will be taken by the camera and prediction will be performed by the model trained on the images stored in database and the identity i.e name of the person will be displayed.

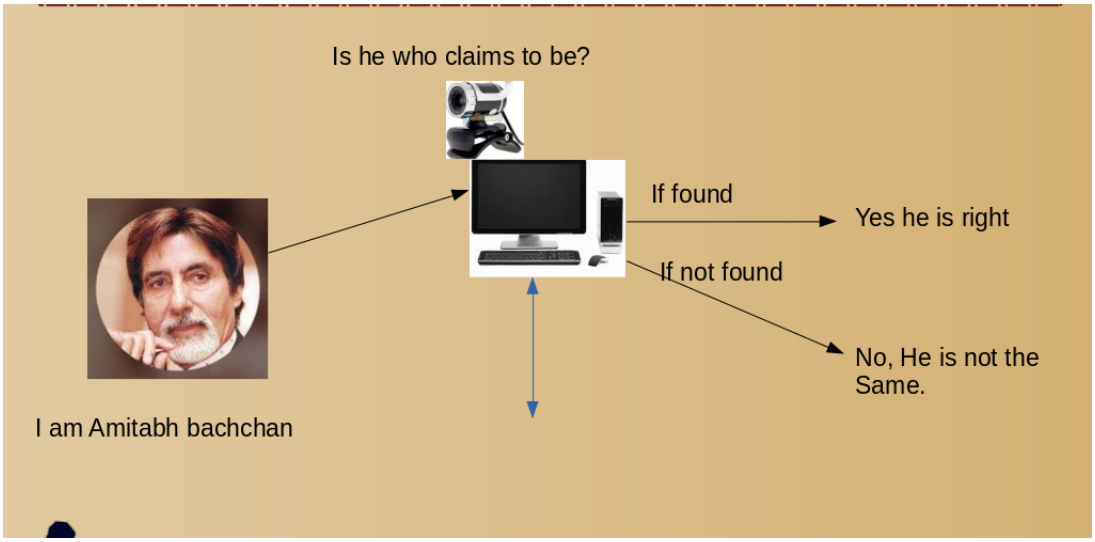

Figure 1.3: Image Verification

The above figure explains the verification process. If the person claims to the one then for the purpose verifying, the images of the person will taken by the camera and the verification will be done by the system. If the person who he claims to be is correct then he/she will be rightly verified.

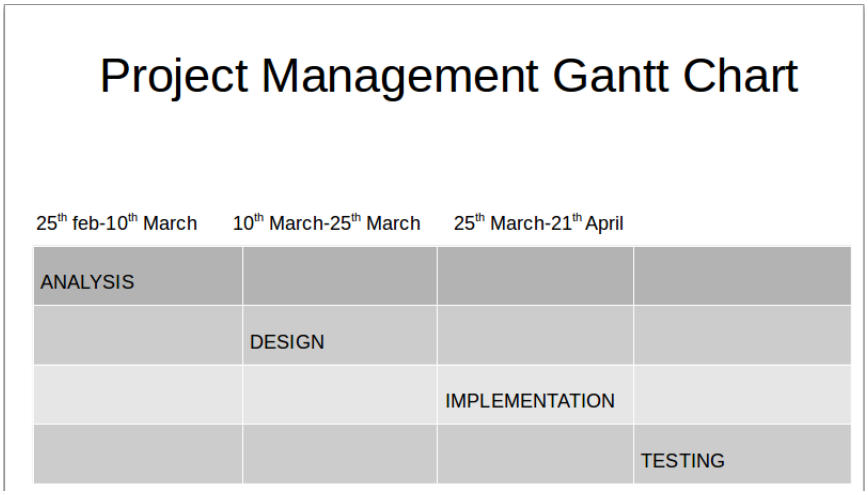

Figure 1.4: Gantt Chart

For completing the project on time the work has been carried in different phase. Different phases with the duration of time are mentioned above.

Chapter 2

Existing System Study

Outline: This chapter presents the study of various existing method:

1. Steps of Face Recognition

2. Eigenface approach

3. Neural Network approach

2.1 Introduction

System analysis is a detailed study of the various operation performed by a system and their relationship within and outside of the system. It is needed to study existing system to know the drawbacks of the present system. The success of the system depends largely on how clearly the problem is defined, throughly investigated and properly carried out through the choice of solution. During analysis, data is collected from various files handled by the present system. The face recognition is the emerging technology since 1964. The Process of identification and verication of images is Face Recognition. The input of a face recognition system is always an image or video stream. The output is an identification or verification of the subject or subjects that appear in the image or video.

2.2 Face Recognition System Analysis

Most of the approaches of existing face recognition system comprises of three steps.

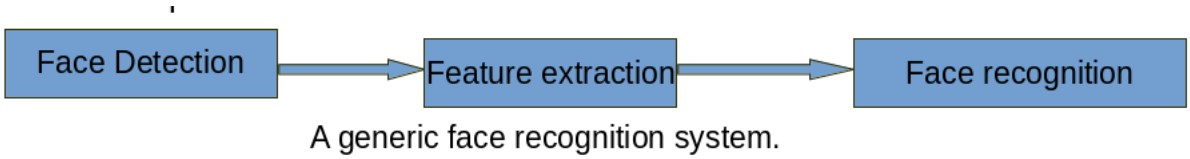

Figure 2.1:

1. Face Detection: Face detection is defined as the process of extracting faces from scenes. So, the system identifies a certain image region as a face.

2. Feature extraction: The next step is feature extraction that involves obtaining relevant facial features from the data. These features could be certain face regions, variations, angles or measures, which can be human relevant (e.g. eyes spacing).

3. Face Recognition: The Process of identification of identity of a person whose images are already stored in database is Face Recognition.

2.2.1 Face Detection

The process of detection of faces from scenes. This process is required in face recognition if the images or scene is having images of things other than face. In this case first it is needed to detect the faces then perform the face recognition. This process is needed where we want to develop automated face traking system. Eg: video surveillance systems try to include face detection, tracking and recognizing. Some applications of face recognition don't require the step of face detection. These system already store the images in standard format.
Eg: Criminal data base. There, the law enforcement agency stores faces of people with a criminal report. If there is new subject and the police has his or her passport photograph, face detection is not necessary.Video survillence system. Face tracking is other problem which sometimes is a consequence of face detection. Many system's goal is not only to detect a face, but to be able to locate this face in real time.
There are different approaches for face detection. Some of the approaches are mentioned below:

Detection depending on the scenario:

1. Controlled environment: Simple edge detection techniques can be used to detect faces.

2. Color images: The typical skin colors can be used to find faces.

3. Images in motion: Real time video gives the chance to use motion detection to localize faces. Another approach based on motion is eye blink detection.

Detection methods divided into categories:

1. Knowledge based methods : Ruled based methods that encode our knowledge of human faces.

2. Feature invariant methods: Algorithms that try to find invariant features of a face despite it's angle or position.

3. Template matching methods : These algorithms compare input images with stored patterns of faces or features.

4. Appearance-based methods: A template matching method whose pattern database is learnt from a set of training images.

Knowledge based methods
It capture our knowledge of faces,and translate them into a set of rules.
The method is divided in several steps::–

1. Find eye-analogue pixels, so it removes unwanted pixels from the image.

2. Consider each eye-analogue segment as a candidate of one of the eyes. Then, a set of rule is executed to determinate the potential pair of eyes.

3. Once the eyes are selected, the algorithms calculates the face area as a rectangle. The four vertexes of the face are determined by a set of functions. So, the potential faces are normalized to a fixed size and orientation.

4. The face regions are verificated using a back propagation neural network.

Appearance-based methods[1] Appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images.

1. Eigenface-based

2. Distribution-based.

3. Neural Networks

4. Support Vector Machines

5. Naive Bayes Classifiers

6. Hidden Markov Model

7. Information-Theoretical Approach

2.2.2 Feature Extraction

Feature extraction process can be defined as the procedure of extracting relevant information from a face image. It involves several steps - dimensionality reduction, feature extraction and feature selection.

Some of the feature extraction algorithm used for face recognition are follows:

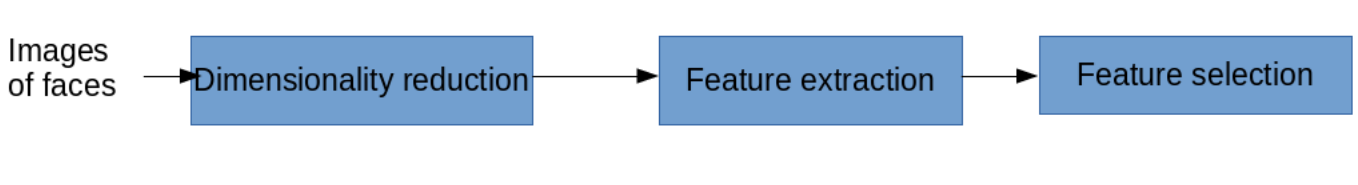

Figure 2.2: Process of feature extraction

1. Principal Component Analysis (PCA) :–Eigenvector-based, linear map.
2. Kernel PCA:–Eigenvector-based, non-linear map, uses
3. kernel methods.
4. Linear Discriminant Analysis (LDA):–Eigenvector-based, supervised linear map
5. Kernel LDA:–LDA-based, uses kernel methods

2.2.3 Face Recognition

2.2.4 The Eigenface approach:

It is also known as Karhunens loeve expansion, eigenvector, eigenpicture and principal component. This algorithm considers the fact that not all parts of a face are equally important and equally useful. When we look at some one we recognize him/her by his distinct features like eyes, nose, cheeks, forehead and how they vary with respect to each other. So we are actually focusing on the areas of maximum change (mathematically speaking, this change is variance) of the face. For example, from eyes to nose there is a significant change and same is the case from nose to mouth. When you look at multiple faces we compare them by looking at these parts of the faces because these parts are

the most useful and important components of a face. Important because they catch the maximum change among faces, change the helps we differentiate one face from the other. This method is based on the Eigenfaces technique in which the Principal Component Analysis is used. This method is successfully used to perform dimensionality reduction. Eigenfaces are the principal components divide the face into feature vectors. The feature vector information can be obtained from covariance matrix. These Eigenvectors are used to quantify the variation between multiple faces. The faces are characterized by the linear combination of highest Eigenvalues. Each face can be considered as a linear combination of the eigenfaces. The face can be approximated by using the eigenvectors having the largest eigenvalue [2].

Algorithmic Description Of Eigenfaces

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a random vector with observations $x_i \in R^d$.

1. Compute the mean μ
   $\mu = 1/n \sum_{i=1}^{n} xi$
2. Compute the the Covariance Matrix S
   $S = \sum_{i=1}^{n} (xi - \mu)(xi - \mu)^T$
3. Compute the eigenvalues λ_i and eigenvectors v_i of S
   $Sv_i = \lambda_i v_i, i = 1, 2, \ldots, n$
4. Order the eigenvectors descending by their eigenvalue. The k principal components are the eigenvectors corresponding to the k largest eigenvalues.
   The k principal components of the observed vector x are then given by:
   $y = W^T(x - \mu)$
   where $W = (v_1, v_2, \ldots, v_k)$
5. The Eigenfaces method then performs face recognition by:
   Projecting all training samples into the PCA subspace.
   Projecting the query image into the PCA subspace.
   Finding the nearest neighbor between the projected training images and the projected query image.

LIMITATIONS OF EIGENFACE APPROACH

1. The accuracy of eigenface depends on many things. As it takes the pixel value as comparison for the projection, the accuracy would decrease with varying light intensity.

2. The method is very sensitive to scale, therefore, a low-level preprocessing is still necessary for scale normalization

3. Since the eigenface representation is, faithful to the original images, its recognition rate decreases for recognition under varying pose and illumination.

2.2.5 Neural Network

A neural network is a system of interconnected artificial "neurons" that exchange messages between each other. The connections have numeric weights that are tuned during the training process, so that a properly trained network will respond correctly when presented with an image or pattern to recognize. [3]The network consists of multiple layers of feature-detecting "neurons". Each layer has many neurons that respond to different combinations of inputs from the previous layer.

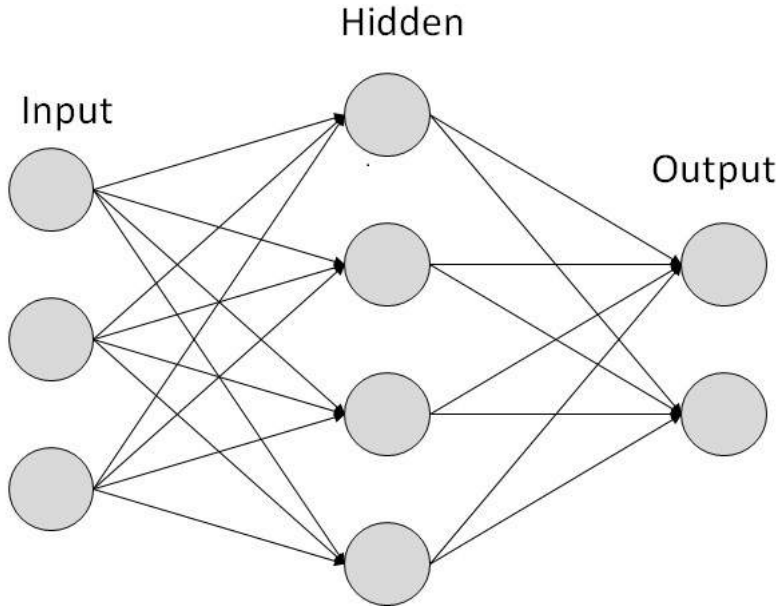

Figure 2.3: A typical artificial Neural Network

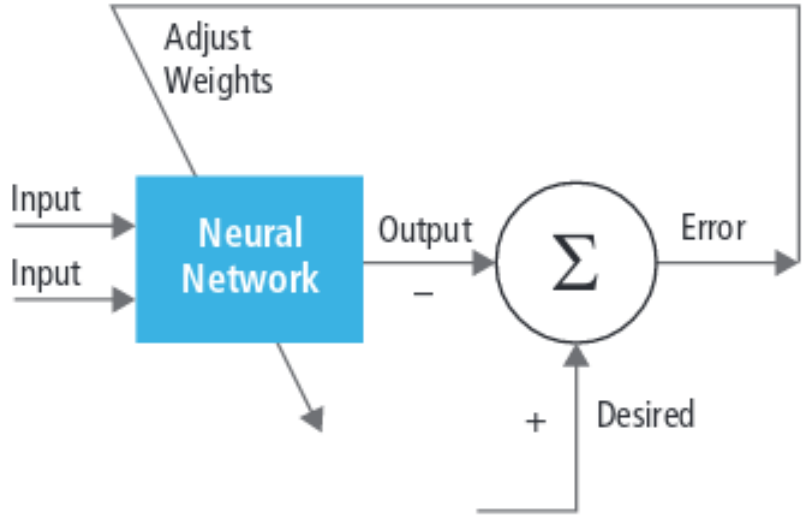

Figure 2.4: Training of neural network Source: [3]

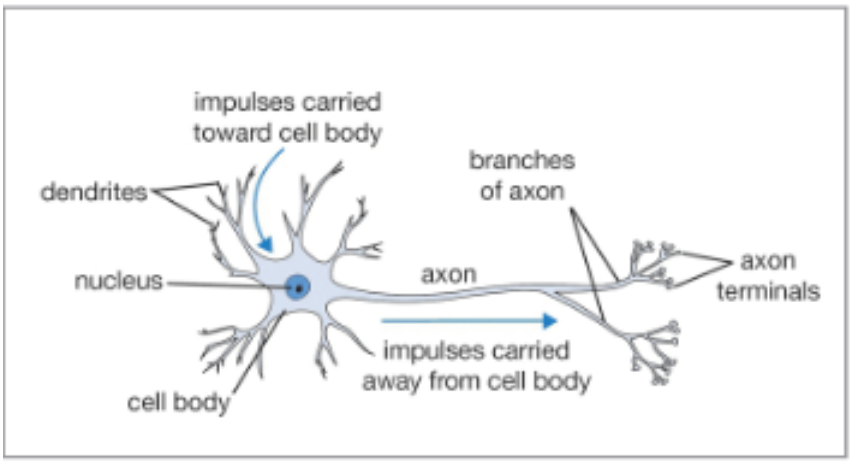

Figure 2.5: Biological Neuron. Source: [3]

Neural networks are inspired by biological neural systems. In a real animal neural system, a neuron is perceived to be receiving input signals from its dendrites and producing output signals along its axon. The axon branches out and connects via synapses to dendrites of other neurons. When the combination of input signals reaches some threshold condition among its input dendrites, the neuron is triggered and its activation is communicated to successor neurons.

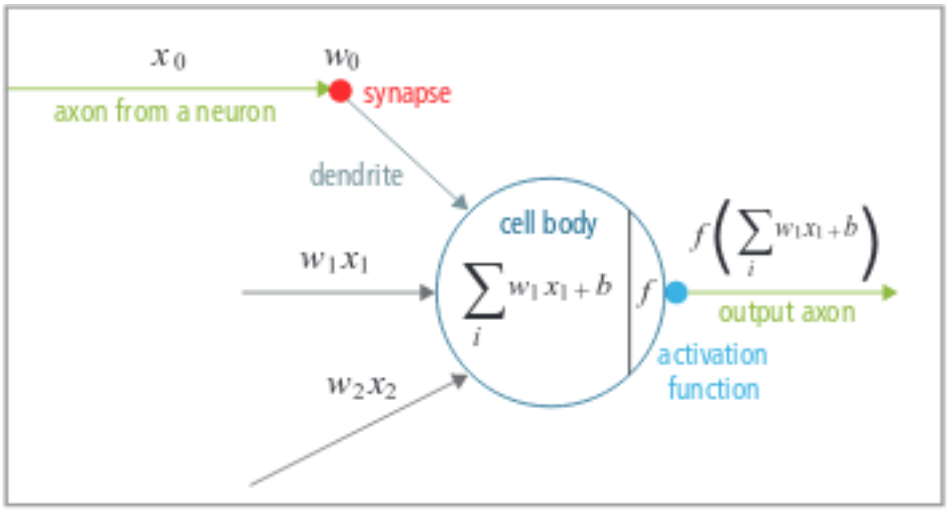

Figure 2.6: Mathematical Model of neurons. Source: [3]

In the neural network computational model, the signals that travel along the axons (e.g.,$x0$) interact multiplicatively (e.g., w0x0) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g., w0). Synaptic weights are learnable and control the influence of one neuron or another. The dendrites carry the signal to the cell body, where they all are summed. If the final sum is above a specified threshold, the neuron fires, sending a spike along its axon. In the computational model, it is assumed that the precise timings of the firing do not matter and only the frequency of the firing communicates information. Based on the rate code interpretation, the firing rate of the neuron is modeled with an activation function f that represents the frequency of the spikes along the axon. A common choice of activation function is sigmoid. In summary, each neuron calculates the dot product of inputs and weights, adds the bias, and applies non-linearity as a trigger function (for example, following a sigmoid response function).

2.2.6 Conclusion from Existing System Analysis

From the Analysis Of Existing System It has been concluded to use CNN for the implementation part of the project. The advantage of using CNN over other pattern detection method are as follows:

1. **Ruggedness to shifts and distortion in the image**
   Detection using CNN is rugged to distortions such as change in shape due to camera lens, different lighting conditions, different poses, presence of partial occlusions,

horizontal and vertical shifts, etc. CNNs are shift invariant since the same weight configuration is used across space.

2. **Fewer memory requirements**
   In this same hypothetical case where we use a fully connected layer to extract the features, the input image of size 32x32 and a hidden layer having 1000 features will require an order of 10^6 coefficients, a huge memory requirement. In the convolutional layer, the same coefficients are used across different locations in the space, so the memory requirement is drastically reduced.

3. **Easier and better training**
   Again using the standard neural network that would be equivalent to a CNN, because the number of parameters would be much higher, the training time would also increase proportionately. In a CNN, since the number of parameters is drastically reduced, training time is proportionately reduced. A standard neural network equivalent to CNN would have more parameters, which would lead to more noise addition during the training process. Hence, the performance of a standard neural network equivalent to a CNN will always be poorer.

Chapter 3

System Analysis, Design & Implementation

Outline: This chapter presents the following:

1. Requirement analysis and specification
2. Feasibility analysis
3. Design
4. Coding and testing
5. Maintainence

3.1 Introduction

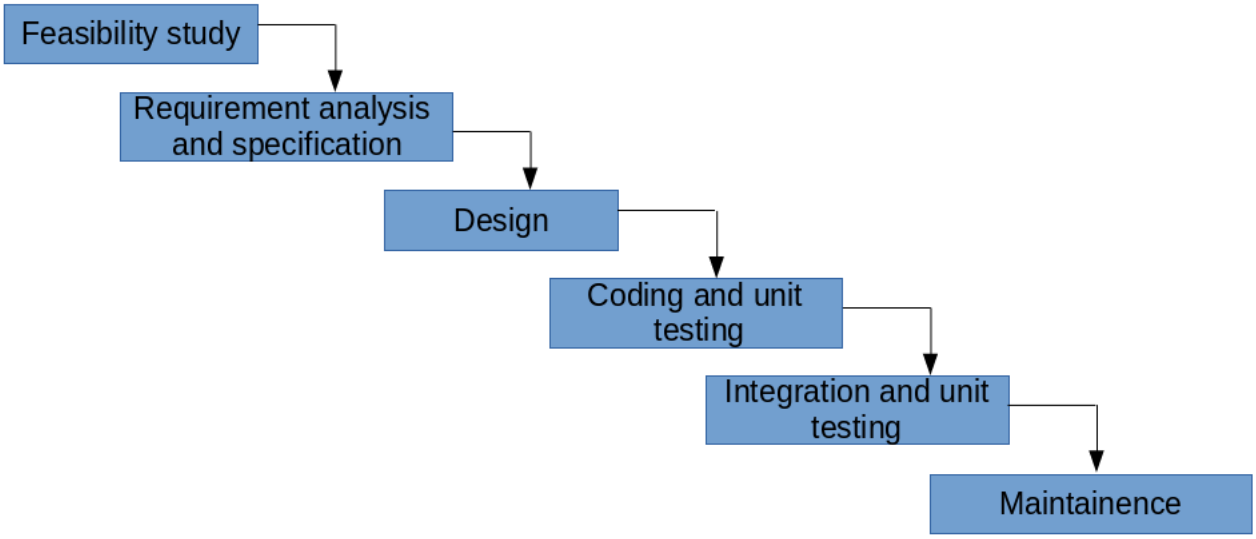

Figure 3.1: Different Steps involved in project formation

In order to complete the project in a systematic way waterfall model as SDLC software development life cycle model is used. A SDLC describes the the various steps that need to be carried out for the software to evolve in its life cycle.

3.1.1 FEASIBILITY ANALYSIS

Feasibility is the study of impact which happens in the organisation by the development of the system. It involves carrying out several activities such as collection of basic information relating to software such as the different data items that would be input to the system, the processing required to be carried out on these data. The impact can either be positive or negative. when the positive impact nominate the negative the system is considered as feasible. Feasibility study can be performed in two ways
1.Technical feasibility
2.Economic feasibilty

3.1.2 Technical feasibility

It can be strongly said that the system is technically feasible, since there is not much difficulty in getting the required resources for development and maintaining the System as well. All the resources that is being used is available in the organisation.

3.1.3 Economic feasibility

Development of the System is highly economic feasible. Organisation needed not to spend much on the development of the system. The only thing needed is effective supervision.

3.2 Requirement analysis and specification

The requirement analysis and specification phase is to understand the exact requirement of the customer and to document them properly. It consist of the distinct activities namely:
1.Requirement Gathering and Analysis
2.Requirement Specification

3.2.1 Requirement Gathering and Analysis

The aim of requirement gathering is to collect all relevant information regarding software to be developed from the customer with a view to clearly understand the requirement. Face recognition system are intended for use in biometric sysytem. Facial feature authentication system is unique with its operation as it does not require contact between individual and authentication device.

3.2.2 Requirement Specification

.

After requirement gathering and analysis activities are complete the identified requirements are documented. This document is called software requirement specification(SRS document). In order to develop Face Recognition System the requirement of tools are as follows:

1.Language : Python3

2.Technique used : Convolutional Nueral Network

3.Package used : Tensorflow, tqdm, matplotlib, tflearn, cv2

4.Operating system : Ubuntu 16.04

5.Image data : Image data is prepared by taking photo from Webcam

6.GUI : tkintter package is used for GUI

3.3 Design

The goal of design approach is to transform the requirement specified in the SRS document into a structure that is suitable for implementation in some programming language. In this phase the approach to solve the problem statement i.e face identification is explained. The system architecture are shown below. For implemeting the system CNN algorithm are used.

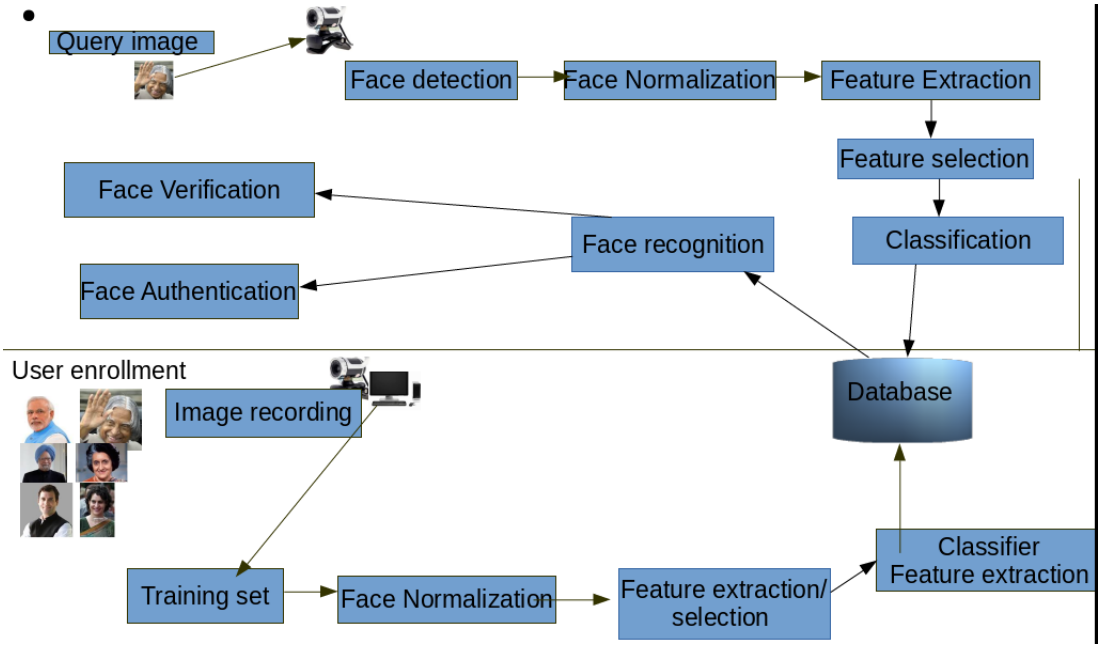

Figure 3.2: System architecture

3.3.1 CONVOLUTIONAL NEURAL NETWORK

A CNN is a special case of the neural network. A CNN consists of one or more convolutional layers[4], often with a subsampling layer, which are followed by one or more fully connected layers as in a standard neural network explained in explained in existing system study. The design of a CNN is motivated by the discovery of a visual mechanism, the visual cortex, in the brain. The visual cortex contains a lot of cells that are responsible for detecting light in small, overlapping sub-regions of the visual field, which are called receptive fields. These cells act as local filters over the input space, and the more complex cells have larger receptive fields. The convolution layer in a CNN performs the function that is performed by the cells in the visual cortex.

1. CNN is most popular deep learning. Due to effectiveness of convnets it became popular technique. The interest in CNN started with AlexNet in 2012 and it has grown exponentially ever since.

2. The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures two or more people it learns distinctive features for each person by itself.

3. CNN is also computationally efficient. It uses special convolution and pooling

operations and performs parameter sharing.

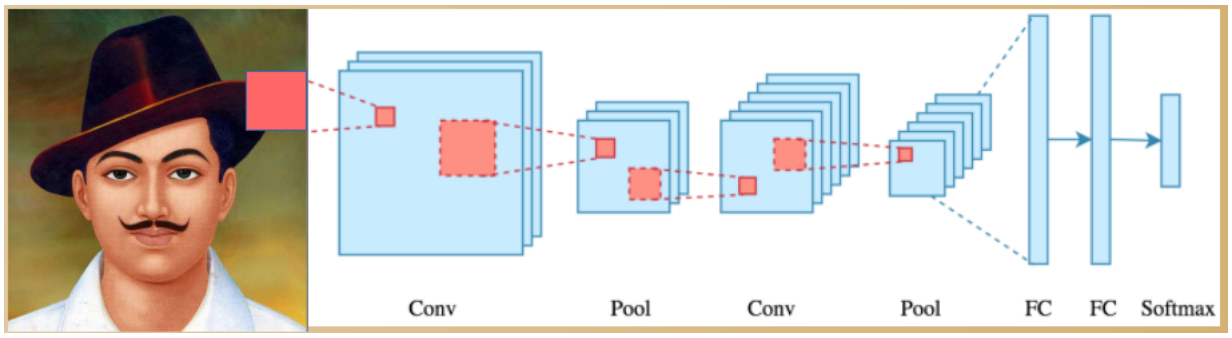

Figure 3.3: Architecture Of CNN

Convolution:: Convolution layer is the main building block of CNN. It is mathematical operation to merge two sets of information. Convolution is applied on the input image using convolution filter to produce a feature map. We continue sliding the filter to the right and perform the same operation for each window and add it to the feature map. We perform multiple convolutions on an input, each using a different filter and resulting in a distinct feature map. We then stack all these feature maps together and that becomes the final output of the convolution layer.

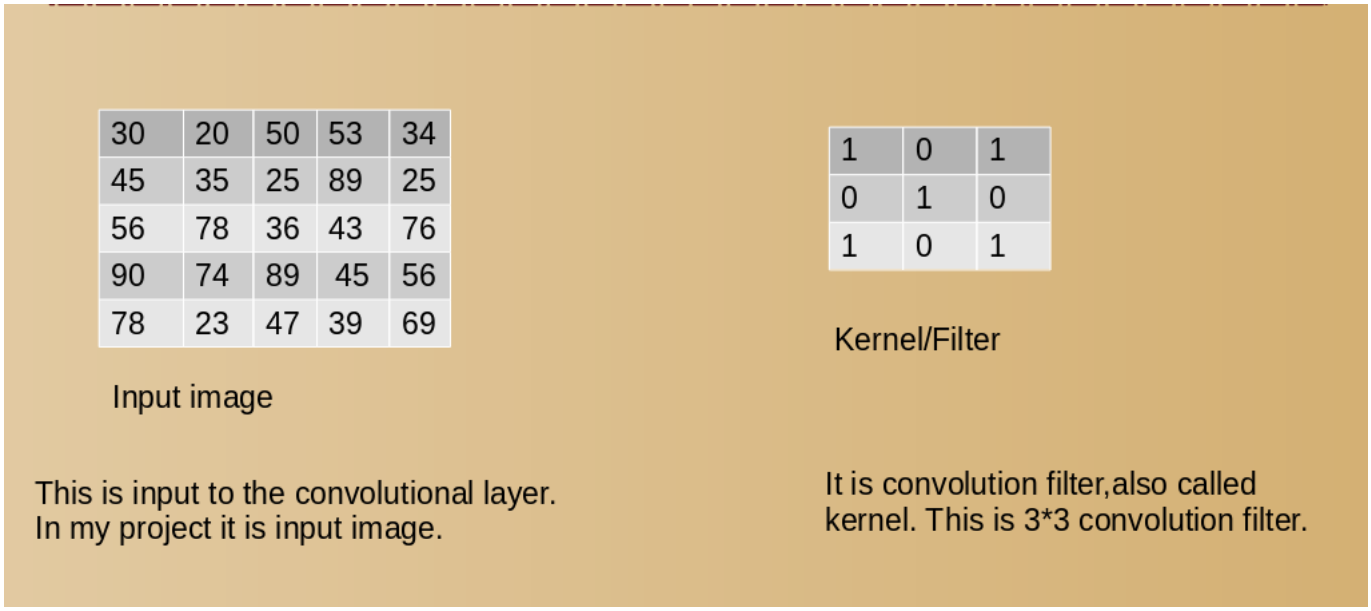

Figure 3.4: Process of CNN

Figure 3.5: Process of CNN

Stride and Padding

1. Stride specifies how much we move the convolution filter at each step. By default the value is 1.
2. We can have bigger strides if we want less overlap between the receptive fields.
3. This also makes the resulting feature map smaller since we are skipping over potential location.
4. To maintain the same dimensionality, we can use padding to surround the input with zeros.
5. Padding is commonly used in CNN to preserve the size of the feature maps, otherwise they would shrink at each layer, which is not desirable.

Pooling

1. After a convolution operation we usually perform pooling to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact.
2. The most common type of pooling is max pooling which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the max value in the window. Similar to a convolution, we specify the window size and stride.

3. The main use case of pooling, downsampling the feature map while keeping the important information. If the input to the pooling layer has the dimensionality 32x32x10, using the same pooling parameters described above, the result will be a 16x16x10 feature map. Both the height and width of the feature map are halved, but the depth doesn't change because pooling works independently on each depth slice the input.

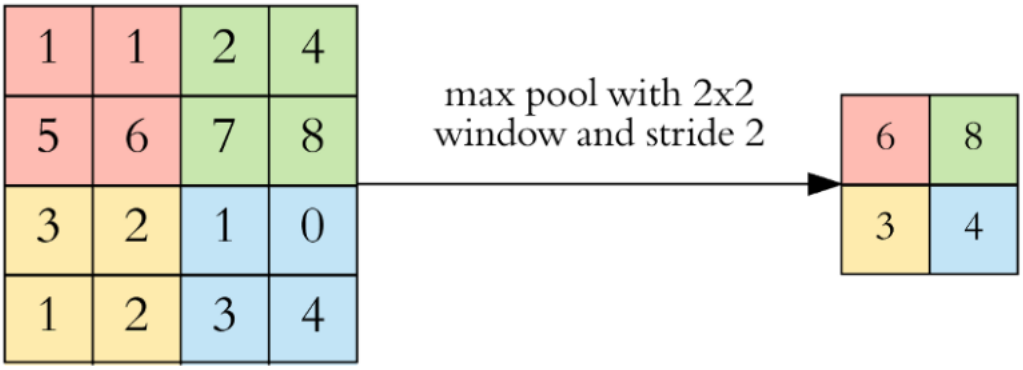

Figure 3.6: Pooling

Fully Connected

1. After the convolution + pooling layers we add a couple of fully connected layers to wrap up the CNN architecture.
2. The output of both convolution and pooling layers are 3D volumes, but a fully connected layer expects a 1D vector of numbers.
3. So we flatten the output of the final pooling layer to a vector and that becomes the input to the fully connected layer.
4. Flattening is simply arranging the 3D volume of numbers into a 1D vector

Softmax Activation function

Activation functions—The main purpose is to convert a input signal of a node in a ANN to an output signal. That output signal now is used as a input in the next layer in the stack.
Specifically in A-NN we do the sum of products of inputs(X) and their corresponding Weights(W) and apply a Activation function f(x) to it to get the output of that layer and feed it as an input to the next layer. Some of the activation function used in CNN are as

follows:

1. Sigmoid or Logistic
2. Tanh—Hyperbolic tangent
3. ReLu -Rectified linear units

1. Sigmoid or Logistic It is a activation function of form f(x) = 1 / 1 + exp(-x). Its Range is between 0 and 1. It is a S—shaped curve. It is easy to understand and apply but it has major reasons which have made it fall out of popularity Vanishing gradient problem. Secondly, its output isn't zero centered. It makes the gradient updates go too far in different directions. $0 <$ output < 1, and it makes optimization harder. Sigmoids saturate and kill gradients. Sigmoids have slow convergence.

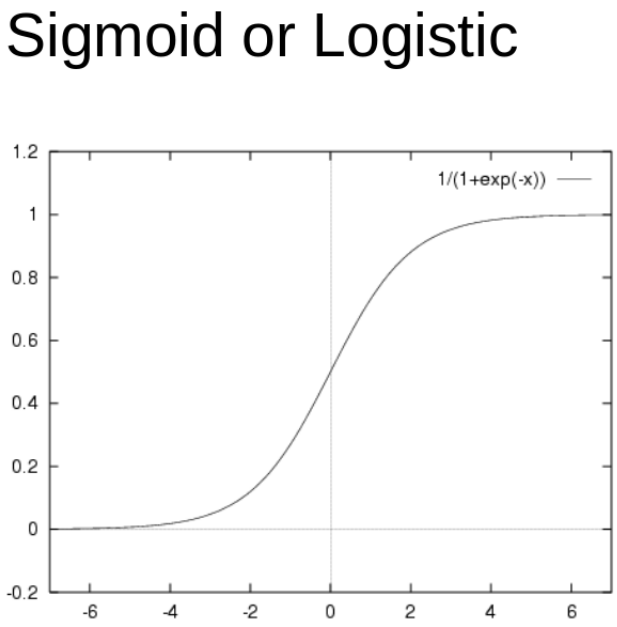

Figure 3.7: Sigmoid Function

2. Hyperbolic tangent It's mathamatical formula is f(x) = 1—exp(-2x) / 1 + exp(-2x) Now it's output is zero centered because its range in between -1 to 1 i.e -1 < output < 1. Optimization is easier in this method hence in practice it is always preferred over Sigmoid function . But still it suffers from Vanishing gradient problem.

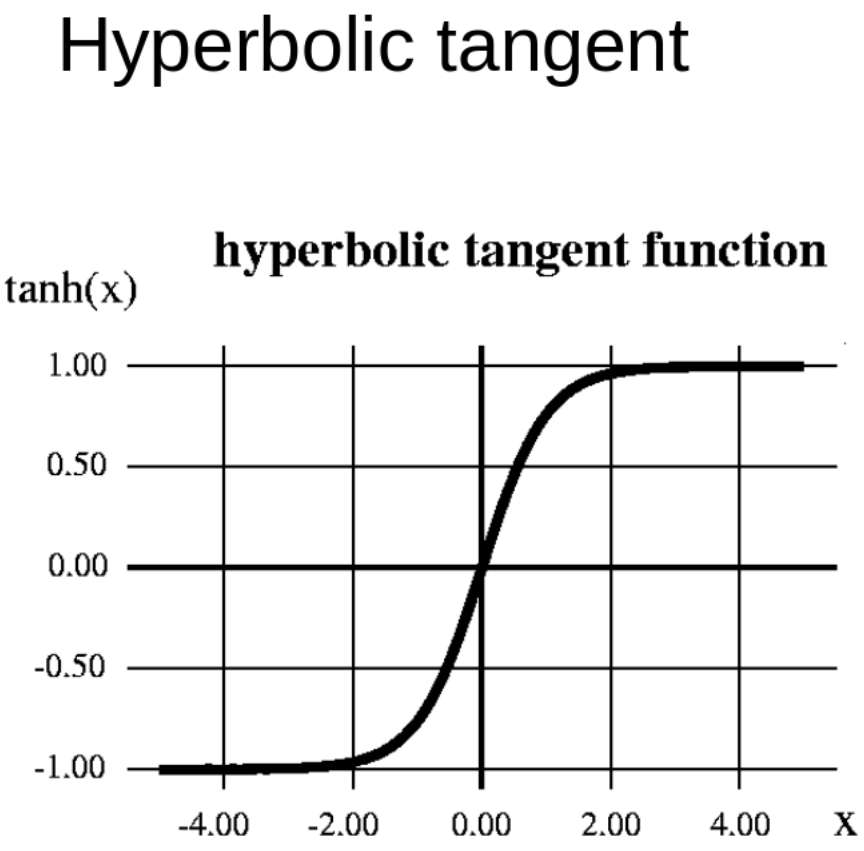

Figure 3.8: Hyperbolic Function

3. ReLu Rectified Linear Units It's just R(x) = max(0,x) i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x. It was recently proved that it had 6 times improvement in convergence from Tanh function. It avoids and rectifies vanishing gradient problem. It's limitation is that it should only be used within Hidden layers of a Neural Network Model. Hence for output layers we should use a Softmax function for a Classification problem to compute the probabilites for the classes, and for a regression problem it should simply use a linear function.

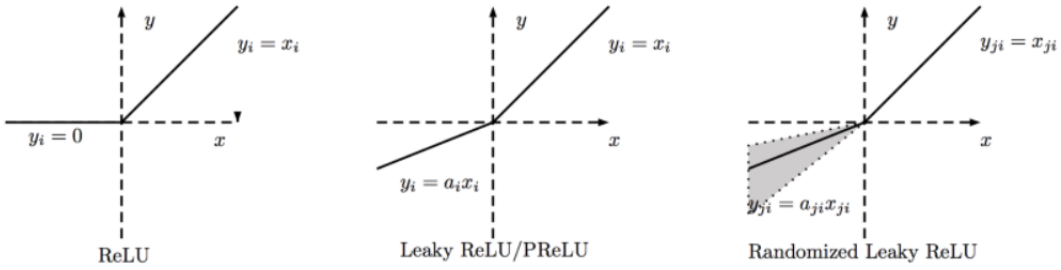

Figure 3.9: Relu Activation Function

3.4 Coding and Unit Testing

This phase of the project will be about implementation of project. For implementing the system dataset prepared containing 62 images of 7 People.

3.4.1 Training Data

For having image data for training it have been stored in folder with format Name.i.jpg where ”Name” is the person Name and ”i” is a variable which count the number of images for each person.

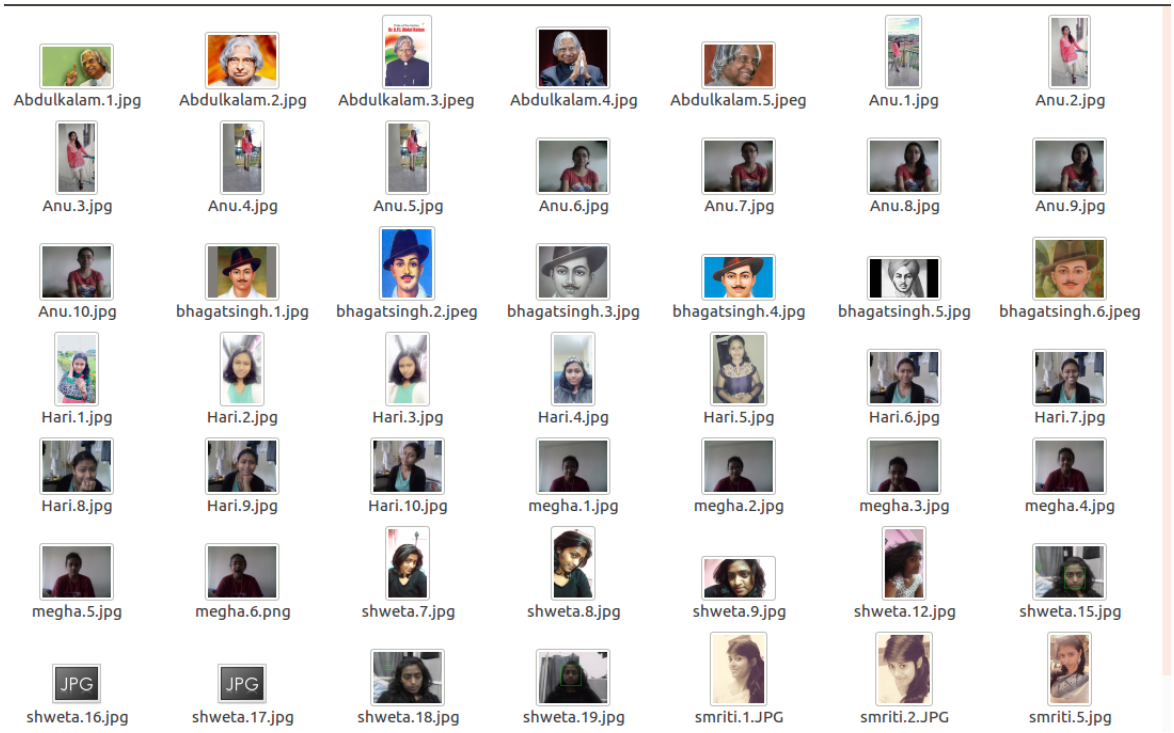

Figure 3.10: Training Images

The above figure shows the training images for the system. To train the model images are saved in some format i.e specified in the program. The format of saving train image is personname.imagenoofperson.jpg.

3.4.2 Label the training data

```
def label_img(img):
        word_label=img.split('.')[-3]
        if word_label=='shweta':return np.array([1,0,0,0,0,0,0])
        elif word_label=='smriti':return np.array([0,1,0,0,0,0,0])
        elif word_label=='Anu':return np.array([0,0,1,0,0,0,0])
        elif word_label=='Hari':return np.array([0,0,0,1,0,0,0])
        elif word_label=='bhagatsingh':return np.array([0,0,0,0,1,0,0])
        elif word_label=='Abdulkalam':return np.array([0,0,0,0,0,1,0])
        elif word_label=='megha':return np.array([0,0,0,0,0,0,1])
```

Figure 3.11: code for labeling the data

1. The code above explains the function that is used for labeling of training images.

2. The split function is used to split the images before image no of person and assign the name of person to the variable word label.

3. For each label a unique array is returned which will be used further in prediction part.

4. When it is need to retrain the model with the images of the person which are not stored in the database, then it is needed to write elif statement and increase the size of array.

5. The size of array will always be equal to the number of person in the database.

3.4.3 Converting training image into array containing the label and pixel value of images

```
def create_train_data():
        training_data=[]
        for img in tqdm(os.listdir(TRAIN_DIR)):
                label=label_img(img)
                path=os.path.join(TRAIN_DIR,img)
                img=cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                img=cv2.resize(img,(IMG_SIZE,IMG_SIZE))
                training_data.append([np.array(img),np.array(label)])
        shuffle(training_data)
        np.save('train_data2.npy',training_data)
        return training_data
```

Figure 3.12: code for training data

1. The code explains the function that is used for converting the image into an list(data structure in python) array of pixel value of image and label of the image.
2. All the images of training folder are read using the cv2 library. Images are converted into Grayscale.
3. Images are resized in $50 * 50$.
4. label of the image is extracted by the function label defined above.
5. Label of the Image and array of pixel values of image is saved into list.
6. After all the image is stored in the list. It is shuffled.
7. After shuffling of the list it is stored in binary file for further use in the program. So we need not to read the image every-time for training.

3.4.4 Building Convolutional Layer

```
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)
convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)
convnet = fully_connected(convnet, 7, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name=

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Figure 3.13: code for building CNN layer

1. The above code is used to build CNN

2. **Conv2d:**This method creates a convolutional layer. The first parameter is the filter count, and the second one is the filter size. For example in the first convolution layer we create 32 filters of size 3x3. We use relu non-linearity as activation.

3. **MaxPooling2D:**This method creates a maxpooling layer, the second argument is the window size.

4. **Dropout:**Dropout is by far the most popular regularization technique for deep neural networks.Dropout is used to prevent overfitting . During training time, at each iteration, a neuron is temporarily "dropped" or disabled with probability p. This means all the inputs and outputs to this neuron will be disabled at the current iteration. The dropped-out neurons are resampled with probability p at every training step, so a dropped out neuron at one step can be active at the next one. The hyperparameter p is called the dropout-rate and it's typically a number. In the above code dropout is 0.8.

3.4.5 Converting Test image into vector

```
def process_test_data():
        testing_data=[]
        for img in tqdm(os.listdir(TEST_DIR)):
                path=os.path.join(TEST_DIR,img)
                img_num=img.split('.')[0]
                img=cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                img=cv2.resize(img,(IMG_SIZE,IMG_SIZE))
                testing_data.append([np.array(img),img_num])

        shuffle(testing_data)
        np.save('test_data3.npy',testing_data)
        return testing_data
```

Figure 3.14: code for prediction

3.4.6 Code For Prediction Of Test Image

```
if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')
import matplotlib.pyplot as plt
fig=plt.figure()
for num,data in enumerate(test_data[:32]):

    img_num = data[1]
    img_data = data[0]
    y = fig.add_subplot(8,4,num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)
    model_out = model.predict([data])
    if np.argmax(model_out) == 1: str_label='Smriti'
    elif np.argmax(model_out) == 2: str_label='Anu'
    elif np.argmax(model_out) == 3: str_label='Hari'
    elif np.argmax(model_out) == 4: str_label='BhagatSingh'
    elif np.argmax(model_out) == 5: str_label='AbdulKalam'
    elif np.argmax(model_out) == 6: str_label='megha'
    else: str_label='Shweta'
    print("identified as",str_label)
    y.imshow(orig,cmap='gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

Figure 3.15: code for prediction

3.4.7 Library Used in Program

```
import cv2
import numpy as np
import os
from random import shuffle
import tensorflow as tf
from tqdm import tqdm
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
```

Figure 3.16: Library used

1. cv2 : In the implemented code cv2 is used to read the images . It is used for plotting the predicted image.

2. Tflearn : TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

3. Tensorflow : TensorFlow is a Python library for fast numerical computing created and released by Google.

Chapter 4

Result Analysis and Testing

Outline: This chapter presents the following:

1. The testing of model on verious optimiser
2. The testing of model on different kind of images

4.1 Introduction

The result and analysis of the implemented system is needed to identify the pros and cons of the system. The reason of the cons of the system are also identified so that the measures can be taken to improve the system further.

4.2 Analysis of Different Optimiser

Optimiser	Accuracy(Training step 1-25)	Accuracy(Training step 25-50)	Accuracy(Training step 50-75)
FTRL	21.43	33.33	21.43
AdaGrad	21.43	28.57	19.05
momentum	30.95	28.57	33.33
RmsProp	35.71	21.43	28.57
SGD	30.95	19.05	26.19
ADAM	54.76	90.48	97.68

The above table is analysed at epoch =25 and learning rate=1e-3

Figure 4.1: Analysis of different Optimiser

The conclusion from the above Table:

1. From above table it has been concluded that the training accuracy of ADAM optimiser is more than the other optimiser.
2. Increasing the epoch not always increases the accuracy in case of all optimiser. From the table it can be observed that the accuracy of optimiser likeFTRL, AdaGrad, RMsProp decrease with increase in epoch.
3. From the above table the theory of visual mechanism, the visual cortex the base of CNN can be proved. In case of recognition by human the more it see the object the more accuratly it identifies the object. The same theory works for CNN as well.

4.3 Analysis of different kind of image

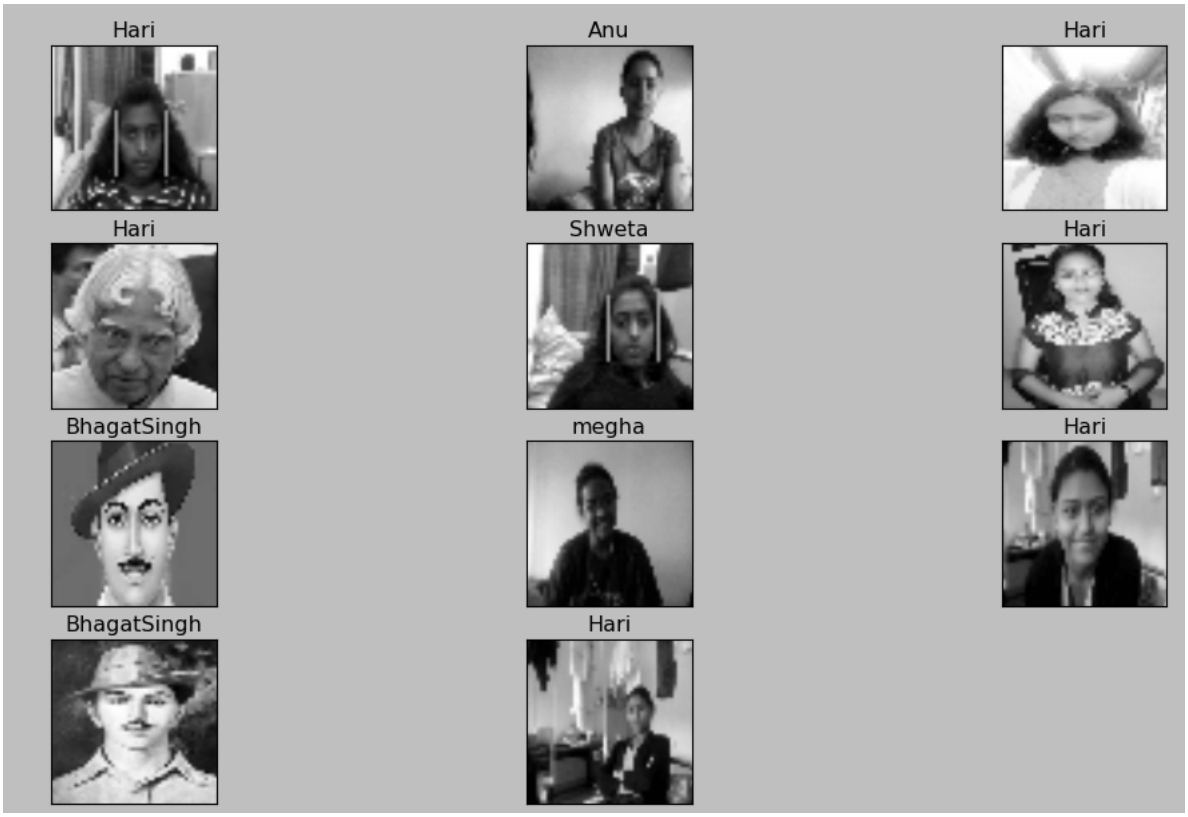

Figure 4.2: Predicted Image

The conclusion from the above Images:

1. The 2 images out of 12 is wrongly predicted. The images which are wrongly predicted is due to the quality of of images given to the test data and train data.
2. The system works well for the images that are taken by webcam for training and for prediction as well.
3. For getting more accuracy the system should be trained with atleast 10 images per person.

4.4 Analysis of different approaches

	Test image=12	Train Images =62
EIGENFACES	Correctly predicted= 7 Wrong prediction=5	Both have same train as well as test images
CONVOLUTION NEURAL NETWORK	Correctly predicted=10	

Figure 4.3: Comaparison of different approaches

For the test data which was used in the project the accuracy of the result is shown above: For Eigenface accuracy prediction is- 0.58. For CNN accuracy of prediction is – 0.83

Chapter 5

Conclusion

The Face Recognition System is working well for the dataset which is prepared by taking image by webcam with enough brightness. While training the model accuracy is good enough around 97 percent. For each person atleast the system need 5-10 images to train the model. The model needs to be train twice or thrice until we get the satisfactory accuracy of training. Retrain the model with same data increase the accuracy because CNN is based on the appearance based the more it see the images the more accurately it identifies the images just like the human being the more number of time we see a person the more we know about them.

5.1 Limitation

It is not working well for training dataset which is edited by the any beauty app.
It is also not working well for the image that are portrait.
It takes long time to train the model
It is not real time i.e every time for predicting it is needed to save the target images into correct folder intended for prediction.
When a person is new to the system then for training the model with their images it is needed to do 3-4 changes in the code for adding it to the model which is later used for prediction.

5.2 Future direction

The future direction of the project are as follows:
1.To make the system Real Time
2.Make it feasible for large dataset
3.To identify Gender of the person
4.The implemented system only perform identification of the person , the verification part of the system will be carried out later

Appendix A

Screenshot and Description of the Implemented System

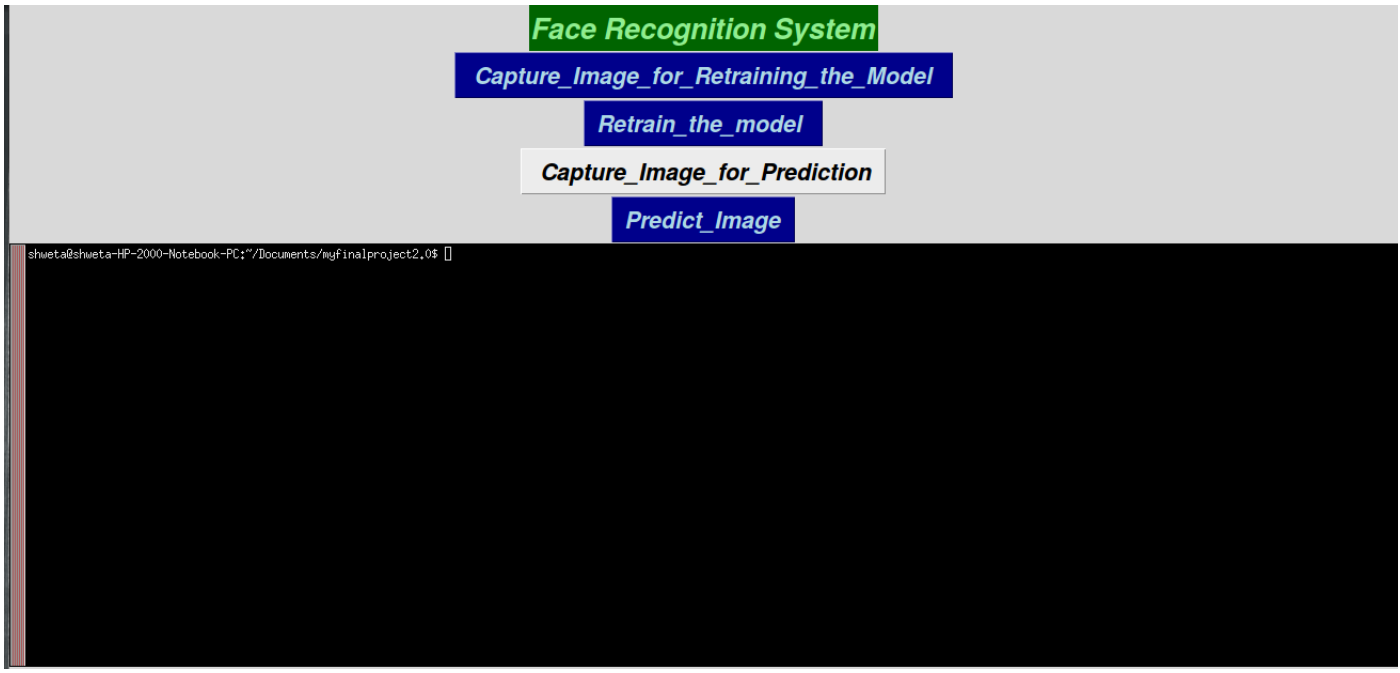

Figure A.1: Interface of the system

The above shown figure is the GUI of implemented system. The interface contains four buttons clicking on which performs different function. Clicking on the button Capture images for Retraining the model will open the webcam to take the images and save it to the training folder. Clicking on the second button i.e Retrain the model will Retrain the model with the new images. Clicking on the third button i.e Capture image for prediction will open the webcam and will direct to save image in testdata. The fourth button i.e Predict image will predict the image stored in testdat.

```
Training Step: 134  | total loss: 1.53024 | time: 1.172s
| Adam | epoch: 034 | loss: 1.53024 - acc: 0.5863 | val_loss: 0.53653 - val_acc: 1.0000 -- iter: 52/52
--
Training Step: 135  | total loss: 1.43308 | time: 1.191s
| Adam | epoch: 035 | loss: 1.43308 - acc: 0.6276 | val_loss: 0.59500 - val_acc: 1.0000 -- iter: 52/52
--
```

Figure A.2: Output while training

The above shown figure shows the metric displayed during training which contains the information of Number of iteration, loss, Accuracy, and the optimiser used.

Figure A.3: Predicted image

The above figure shows the images with label identified by model which are in testdata folder.

Appendix B

User manual

B.1 Introduction

The Face recognition Sytem is used for identifying the identity of the person. Just like the Recognition works in case of human being it works to the system as well. When we meet a person we identify some features of him/her that differentiate it from other. In the same way while training the model it read the images and identify the features and stores it and while identifying it uses the the stored features to identify. The implemeted system works well in case of the training data preapred by webcam with enough brightness. It is mainly intended to use for Biometric system where the images will be taken by the camera without any editing of the image. The Biometric part is not completed in this version of the software. That part will be the continuation of my project.

B.2 Step to install your implemented system

To use the implemented Face Recognition System user need to have following prerequisite installed in their system:
1.Python3
2.Tensorflow
3.Tqdm
4.Tflearn
5.Opencv
6.Numpy
7.Tkintter

The steps below are for installation on ubuntu 16.04. To install all the packages follow the steps below:

1. Install python3 on ubuntu 16.04. Open Terminal and type the following command:
   sudo apt-get update
   sudo apt-get install python3
   To check of the version of Python type:
   python -V
   To manage software package of python install pip
   sudo apt-get install -y python3-pip

2. Install tensorflow
   One need to install tensorflow with CPU support.User can install it by typing
   pip3 Install tensorflow
   if pip3 is not working then install it following command
   sudo apt-get install python3-pip python3-dev

3. Install tflearn by typing the command
   pip3 install tflearn

4. Install Numpy
   sudo pip3 install numpy scipy

5. Install package python-opencv with following command in terminal
   sudo apt-get install python-opencv

6. Install tqdm with the following command
   pip3 install tqdm

7. Install tkinter for the GUI
   sudo apt-get install python3-tk

After the complete installation of the above package one can run system by typing the command in terminal
python3 Mainprog.py
After running it successfully user will be directed to a interface where they can do
1.Capture New Image for Retraining the model
2.Retrain the model
3.Capture New image for prediction
4.Predict the captured image

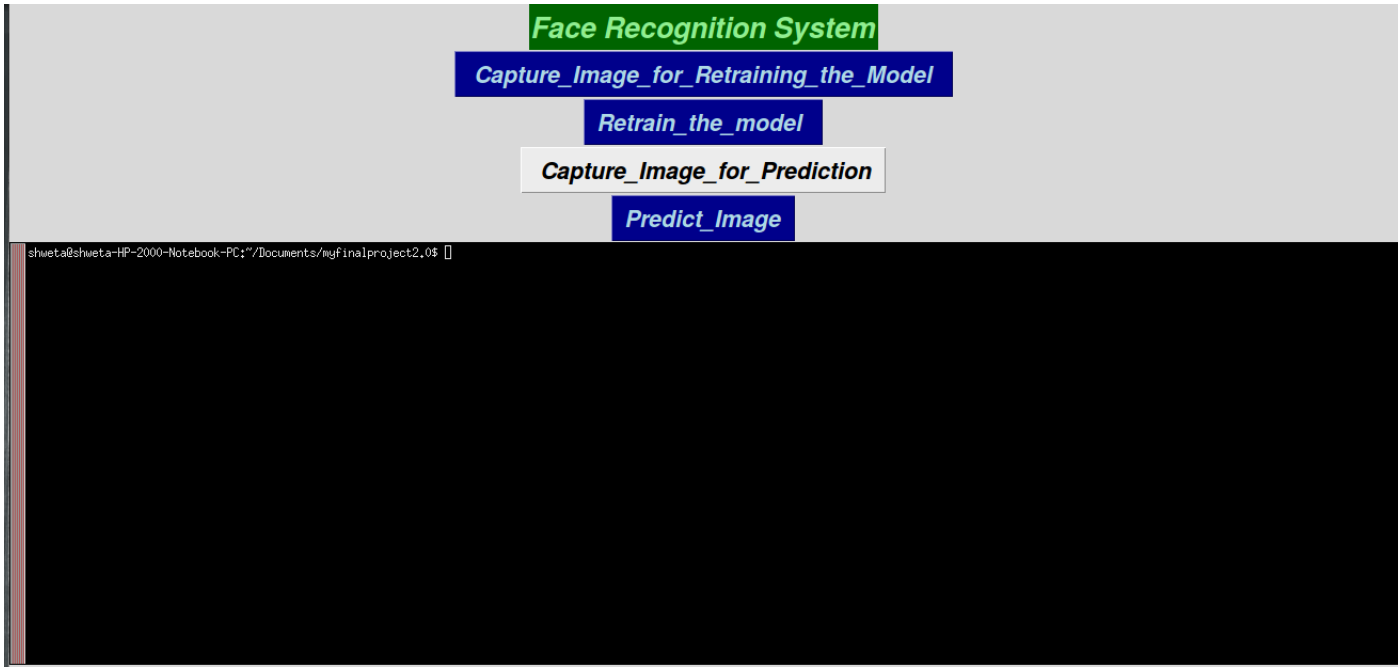

Figure B.1: Interface

The above shown interface of the system has four buttons clicking on which perform different function:
First button is for Capturing Image for Retraining the Model. Clicking on this will open webcam where you can capture images of person you wish to and save it the folder named traindata. For saving the images the proper format is specified i.e Nameoftheperson.serialnotheimagesoftheperson.jpg. Second button is for Retraining the model with the new Images. For Retraining the model with new images of the person who is new to the system we need to do the following

1. Delete the checkpoint and the existing model model from the software folder.
2.Because the number of classes is increasing so we need to do some changes in program.Add the name of the person through program train.py and increase the size of array.
3. Add the name of person in predict.py.
4.Now run Retrain the model
Third button is for caputing the image for identification. Once the image is capture save to the folder named testdata. Fourth button is for identifying the image. Cilcking it will give the image with label identified by model. Below the Buttons System terminal is added where all the ongoing process after clicking any of the button above will be shown.

Bibliography

[1] P. F. De Carrera and I. Marques, "Face recognition algorithms," *Master's thesis in Computer Science, Universidad Euskal Herriko*, 2010.

[2] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[3] P. Latha, L. Ganesan, and S. Annadurai, "Face recognition using neural networks," *Signal Processing: An International Journal (SPIJ)*, vol. 3, no. 5, pp. 153–160, 2009.

[4] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," *Cadence Design Systems Inc.: San Jose, CA, USA*, 2015.