ORCHID INTERNATIONAL COLLEGE

Tribhuvan University

Institute of Science and Technology

A FINAL REPORT ON

ATTENDANCE SYSTEM USING FACE RECOGNITION

In Partial Fulfillment of Requirements for the Bachelor Degree in Computer Science and Information Technology

Submitted To:

Department of Computer Science and Information Technology,

Orchid International College

Submitted By:

Akash Shrestha (T.U. Roll No. 4962/071)

Dagina Basnet (T.U. Roll No. 4973/071)

September, 2018

ACKNOWLEDGEMENTS

The completion of this project would not have been possible without the support and guidance of many individuals.

We would like to express our gratitude to our project supervisors Mr. Nipun Thapa who took an interest in our project and guided us through the project by providing necessary ideas, information and knowledge for developing system that attend the student automatically. This system is based on face detection and recognition algorithms, automatically detects the student and marks the attendance by recognizing them. We would like to thank Mr. Utsav Koirala and Mr. Dhiraj Jha for their encouragement and guidance towards the making of this report as per the standard.

We are thankful and fortunate enough to get constant support from our colleagues and teaching staff of B.Sc.CSIT department, which helped us, complete our project. We would also like to extend our regards to all the non-teaching staff of B.Sc.CSIT department for their timely support.

Akash Shrestha
Roll No. 4962/071

Dagina Basnet
Roll No. 4973/071

ABSTRACT

The traditional attendance systems in school and colleges have been so time consuming, old fashioned and no so efficient to keep track of the students. Automated Attendance System is the advancement that has taken place in the field of automation replacing traditional attendance marking activity. This project is developed using Image Classification Algorithm Convolutional Neural Network and Python coding.Face recognition is an important application of Image processing owing to its use in many fields.The proposed system aims to overcome the pitfalls of the existing systems and provides features such as detection of faces, extraction of the features, detection of extracted features, and analysis of students' attendance. The technique behind the face recognition the training images are feed into the model and model learn the parameters. After learning parameters, the haarcascade classifier and CNN is used to classify the images and detect the faces in the real time.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
CNN	Convolutional Neural Network
GB	Gigabyte
GHz	Gigahertz
GUI	Graphic User Interface
IP cam	Internet Protocol Camera
JPG/JPEG	Joint Photographic Experts Group
MNIST	Modified National Institute of Standards and Technology
RFID	Radio Frequency Identification
RGB	Red-Green-Blue

Chapter 1 - Introduction

1.1. Overview

The automated attendance management system is based on face detection and recognition algorithms, automatically detects the student and marks the attendance by recognizing them. The “Attendance system using Face recognition” is a hardware prototype of a face recognition attendance system. This project is developed using Image Classification Algorithm Convolutional Neural Network and Python coding. Face recognition is an important application of Image processing owing to its use in many fields. Identification of individuals in an organization for the purpose of attendance is one such application of face recognition. Maintenance and monitoring of attendance records plays a vital role in the analysis of performance of any organization. The purpose of developing attendance management system is to digitize the traditional way of taking attendance. Automated Attendance Management System performs the daily activities of attendance marking and analysis with reduced human intervention. The prevalent techniques and methodologies for detecting and recognizing face fail to overcome issues such as scaling, pose, illumination, variations, rotation, and occlusions. The proposed system aims to overcome the pitfalls of the existing systems and provides features such as detection of faces, extraction of the features, detection of extracted features, and analysis of students' attendance. The technique behind the face recognition the training images are feed into the model and model learn the parameters. After learning parameters, the haarcascade and CNN is used to classify the images and detect the faces in the real time. The system is tested for various use cases. We consider a specific area such as classroom attendance for the purpose of testing the accuracy of the system.

1.2. Statement of Problem

In this modern era of automation many scientific advancements and inventions have taken place to save labor, increase the accuracy and to ameliorate our lives. The traditional attendance systems in school and colleges have been so time consuming, old fashioned and no so efficient to keep track of the students. Automated Attendance System is the advancement that has taken place in the field of automation replacing traditional attendance marking activity. Automated Attendance Systems are generally biometric based, smart-card based and web based. These systems are widely used in different organizations. Traditional method of attendance marking is very time consuming and becomes complicated when the strength is more. Automation of Attendance System has edge over traditional method as it saves time and also can be used for security purposes. This also helps to prevent fake attendance. An Attendance Management System which is developed using biometrics, in our case face, generally consists of Image Acquisition, Database development, Face detection, Preprocessing, Feature extraction, and Classification stages followed by Post-processing stage

1.3. Objectives

The main objectives of the project are:

- To detect unique face image amidst the other natural components such as walls, backgrounds etc.
- To perform effective recognition of unique faces in a crowd (individual recognition in crowd).

1.4. Report Organization

Our report is organized into 6 chapters:

Chapter 1: Introduction

In this section the brief introduction of our project, statement of the problem and its objectives are discussed.

Chapter 2: Literature Review

The previous work related to our projects were studied and are briefly summarized in this section.

Chapter 3: Requirement Analysis and Feasibility Study

The functional and nonfunctional requirement of the system along with different feasibility studies are discussed in this section.

Chapter 4: System Design

In this section we have designed the structuring system requirement like activity diagram, sequence diagram, database design etc.

Chapter 5: Implementation and testing

The various implementation method and tools are described in this section. This part also contains the description of various testing and results we got after performing them.

Chapter 6: Conclusion and Recommendations

This section contains the conclusion and recommendation based on our project.

Chapter 2 - Literature Review

In a RFID-Based Students Attendance Management System[1] in recent years, there have been rise in the number of applications based on Radio Frequency Identification (RFID) systems and have been successfully applied to different areas as diverse as transportation, health-care, agriculture, and hospitality industry to name a few. RFID technology facilitates automatic wireless identification using electronic passive and active tags with suitable readers. In this paper, an attempt is made to solve recurrent lecture attendance monitoring problem in developing countries using RFID technology. The application of RFID to student attendance monitoring as developed and deployed in this study is capable of eliminating time wasted during manual collection of attendance and an opportunity for the educational administrators to capture face-to-face classroom statistics for allocation of appropriate attendance scores and for further managerial decisions. In Biometric time and attendance system[2] is one of the most successful applications of biometric technology. One of the main advantage of a biometric time and attendance system is it avoids "buddy-punching". Buddy punching was a major loophole which will be exploiting in the traditional time attendance systems. Fingerprint recognition[3] is an established field today, but still identifying individual from a set of enrolled fingerprints is a time taking process. Most fingerprint-based biometric systems store the minutiae template of a user in the database. It has been traditionally assumed that the minutiae template of a user does not reveal any information about the original fingerprint. This belief has now been shown to be false; several algorithms have been proposed that can reconstruct fingerprint images from minutiae templates. In this paper, a novel fingerprint reconstruction algorithm is proposed to reconstruct the phase image, which is then converted into the grayscale image. The proposed reconstruction algorithm reconstructs the phase image from minutiae. The proposed reconstruction algorithm is used to automate the whole process of taking attendance, manually which is a laborious and troublesome work and waste a lot of time, with its managing and maintaining the records for a period of time is also a burdensome task. The proposed reconstruction algorithm has been evaluated with respect to the success rates of Type-I attack (match the reconstructed fingerprint against the original fingerprint) and type-II attack (match the reconstructed fingerprint against different impressions of the original fingerprint) using a commercial fingerprint recognition system. Given the reconstructed image from our algorithm, we show that both types of attacks can be effectively launched against a fingerprint recognition system. In the authors have proposed Daugman's algorithm based Iris recognition system. This system uses iris recognition management system that does capturing the image of iris recognition, extraction, storing and matching. But the difficulty occurs to lay the transmission lines in the places where the topography is bad. In authors have proposed a system based on real time face recognition which is reliable, secure and fast which needs improvement in different lighting conditions.

Chapter 3 - Requirement Analysis and Feasibility Study

3.1. Requirement Analysis

Requirement analysis holds the process of reviewing and determining the system needs, functional requirements and non-functional requirements that a system must meet. For requirement analysis, following approaches are followed:

3.1.1 Preliminary Analysis

- Problem analysis has been done and titled as problem statement
- Performed literature review to know about algorithm used in existing system.
- Analysis of system planning and design.

3.1.2. Functional Requirement

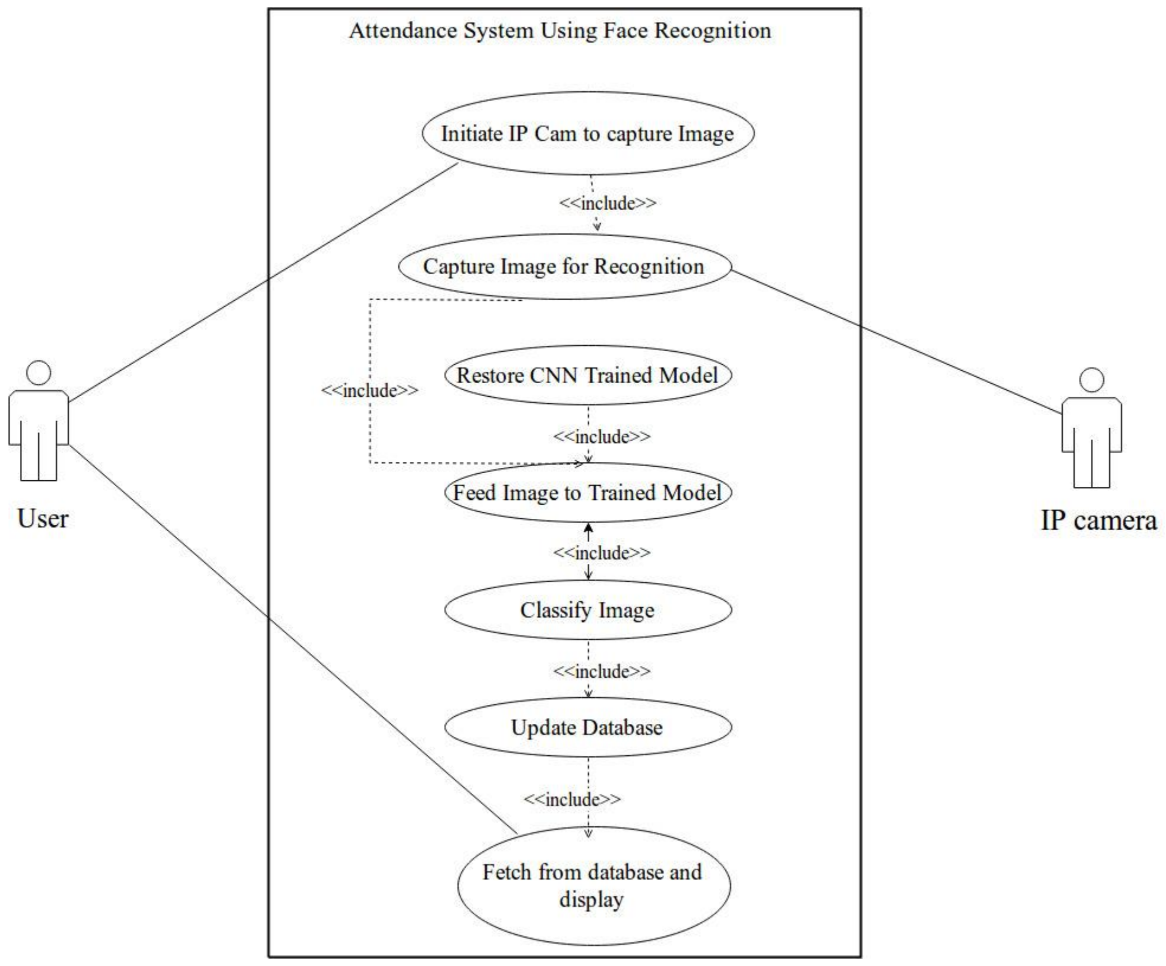
Functional requirement identifies the provision of the system and the system's reaction to certain input and how the system should behave in day to day basis. This attendance system is focused on capturing the image of an individual, classify and detect them and know the presence or absence of any individual.

The functional requirement for the 'Attendance System Using Face Recognition' includes following tasks:

- System should capture the image of the person in real time and feeds them to the model for classification in no time.
- System shouldn't take so long to classify and test the image.
- System should detect the face correctly.
- The database must be updated after the detection.

Use Case Diagram

Our system has following use case diagram



Attendance System Using Face Recognition
Initiate IP Cam to capture Image
<<include>>
Capture Image for Recognition
<<include>>
Restore CNN Trained Model
<<include>>
Feed Image to Trained Model
<<include>>
Classify Image
<<include>>
Update Database
<<include>>
Fetch from database and display
User
IP camera

Figure 3.1. Use case diagram for Attendance System Using Face Recognition

Use-case Description

Table 3.1. Initiate IP camera to capture image

Use-case 1	Initiate IP camera to capture image
Primary Actor	IP cam
Description	The IP camera gets ready to capture the image of the person in real time.
Pre-condition	The camera should be in proper working condition
Post-condition	The camera should capture the image of a person
Failure Scenario	IP camera fails to start

Table 3.2. Capture image for Recognition

Use-case 2	Capture image for Recognition
Primary Actor	IP cam
Secondary Actor	User
Description	The image of a person is captured for further processing and recognition using the IP camera
Pre-condition	The camera and person whose photo should be taken should be ready
Post-condition	A recognizable image of the person
Failure Scenario	Blur image or unclear image, sudden shutdown of camera

Table 3.3. Restore CNN Model

Use-case 3	Restore CNN Model
Primary Actor	System
Description	The CNN model which has been trained and saved earlier should be restored for the classification of taken images.
Pre-condition	The well trained CNN model should be saved properly
Post-condition	The restored CNN model
Failure Scenario	Model may not be trained well, difficulty in restoring model, inconsistency in accuracy of model after restoration.

Table 3.4. Feed image to trained model

Use-case 4	Feed image to trained model
Primary Actor	System
Description	The image captured previously are fed to the restored CNN model for further classification and detection
Pre-condition	The model and images to be fed should be ready
Post-condition	A model with image fed to it.
Failure Scenario	Model fails to take image database.

Table 3.5. Classify image

Use-case 5	Classify image
Primary Actor	System
Description	The image fed into the CNN model are now classified on the basis of the training data provided to it.
Pre-condition	A model with image fed to it
Post-condition	Classified and detected image
Failure Scenario	Model fails to classify image accurately, wrong detection.

Table 3.6. Update database

Use-case 6	Update database
Primary Actor	System
Description	The result after classification are now stored in a database.
Pre-condition	A database to store result
Post-condition	An updated database
Failure Scenario	Difficulty in updating database, server problems.

Table 3.7. Fetch from database and display

Use-case 7	Fetch from database and display
Primary Actor	User
Secondary Actor	System
Description	The updated results of database are fetched and displayed to the user using a user interface.
Pre-condition	User fetches information from database
Post-condition	Updated data shown in an interface
Failure Scenario	Error while fetching data, database server problems.

3.1.3. Non-functional Requirement

Non-functional requirement deals with the quality and performance of the system. It focuses on security, advancements and performance mechanism of the system. The non-functional requirements of the 'Attendance System Using Face Recognition' includes following points:

- The image should be captured using IP cam.
- There should be consistency in an accuracy of the system.
- The system should be usable, reliable and effective to the user.
- The system should only perform in terms of valid input provided by the user.

3.1.4 Software Requirement

- MySQL database
- Apache Server
- Python and Java programing language
- -IP cam

3.1.5. Hardware Requirement

- PC or laptop with the minimum system requirement of:

    - Operating system Windows XP or higher

    -Processor: 2.50 GHz

    - Memory: 6 GB (minimum)

    -Network Connection: yes

    -Portable Router (optional)

3.2. Feasibility Analysis

3.2.1. Technical Feasibility

The system is easier to use and there is no higher technical resource demand for the system. Once a user gets proper guidelines they can use it without any difficulties. It also doesn’t require any third party software.

3.2.2. Operational Feasibility

The system is operationally feasible if it can clearly classify and detect the person. The well planned design of the system will ensure the optimal utilization of the computer resources (such as storage, memory processing etc.). The system will have simple user interface, so that any non-technical user can operate with the system.

3.2.3 Schedule Feasibility

The team members will be responsible for different aspect of the system development. The project is intended to be completed within 50-60 days, it is feasible with respect to time also. The schedule to develop the product is presented in the Gantt chart below which describes the time specified for different task performed during system development.

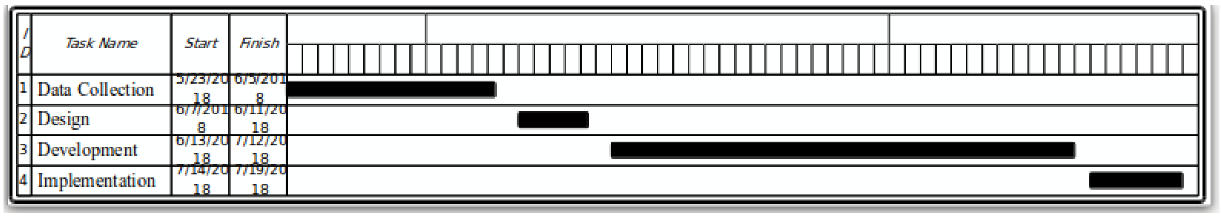


Figure 3.2. Gantt chart

3.2.4. Economic Feasibility

This system doesn’t require extra costs for the development and implementation. User doesn't even require the Internet connection in order to use this system. Once the system is ready and user gets guidelines to use it, it is the easiest and very low cost system to implement

Chapter 4 - System Design

System design is basically a process of defining the components, modules, interfaces and data for a system so as to satisfy specified requirements. The main objective of system design is to prepare a blueprint of a system that meets the goals of the proposed system. It can also be for altering systems along with the processes, models and methodologies that can be used to develop them. The system designs used for building this project include activity diagram, module diagram, sequence diagram, database diagram and class diagram.

4.1. System Design

4.1.1. Module Diagram

Our system contains six modules, whose relationships are illustrated in figure below:

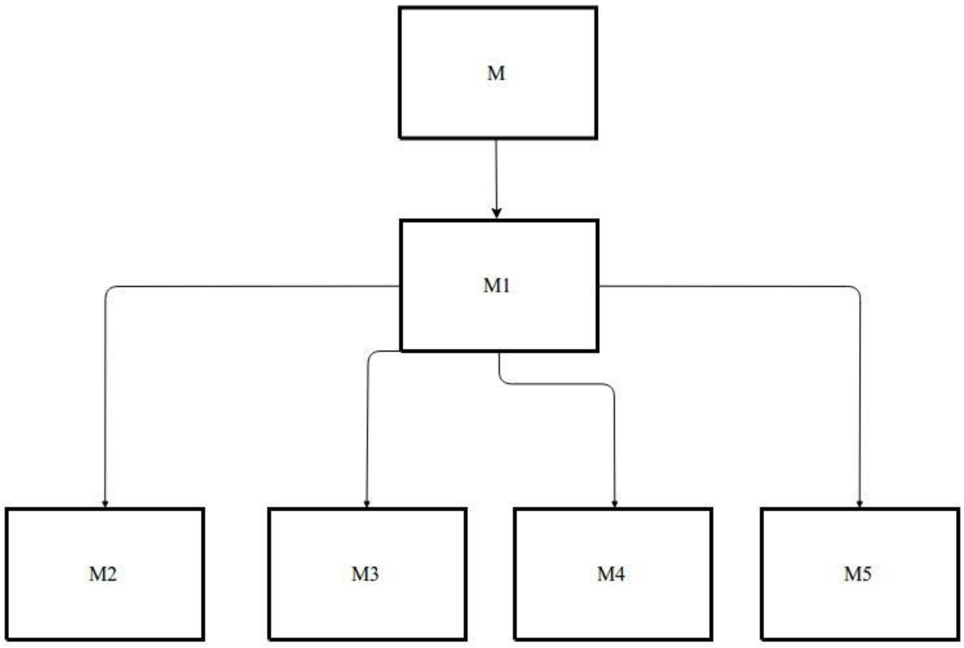


Figure 4.1. Module diagram for Attendance System Using Face Recognition

Table 4.1. Module Description

M	main_system()
M1	load_model_and_predict1(middle(int))
M2	taking_input1()
M3	extract_face()
M4	predict(X(numpy array),i(int),middle(int))
M5	insert_data(Nam(String), A1(int), A2(int), A3(int), A4(int), A5(int))

4.1.2. Activity Diagram

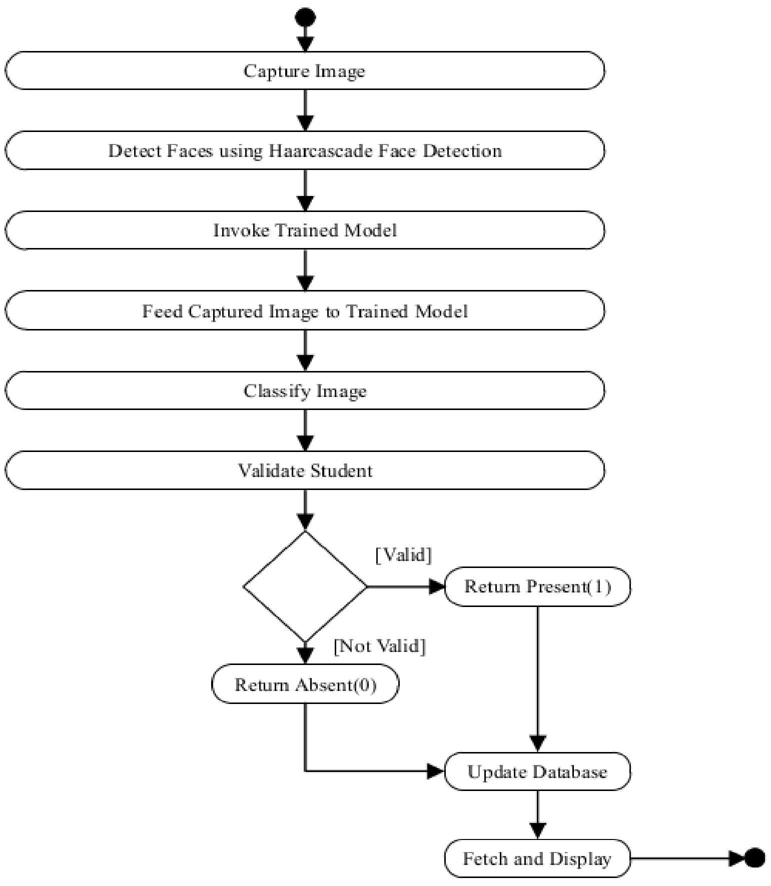


Figure 4.2. Activity diagram for Attendance System Using Face Recognition

4.1.3. Sequence Diagram

The sequence diagram for detection of face is illustrated below:

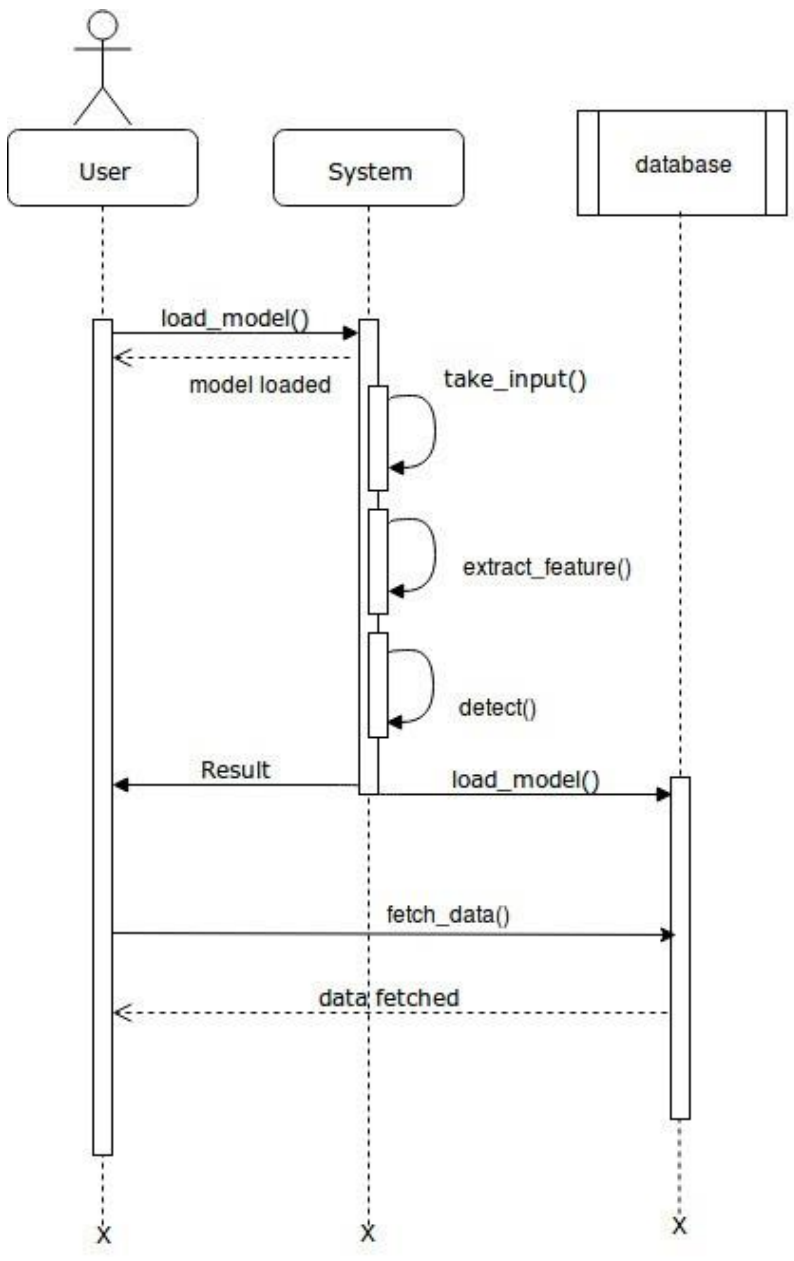


Figure 4.3. Sequence diagram for Detection of image

4.1.4. Class Diagram

The classes are created for creating the interface for our system. Our system has two classes and their relationship is illustrated by following class diagram.

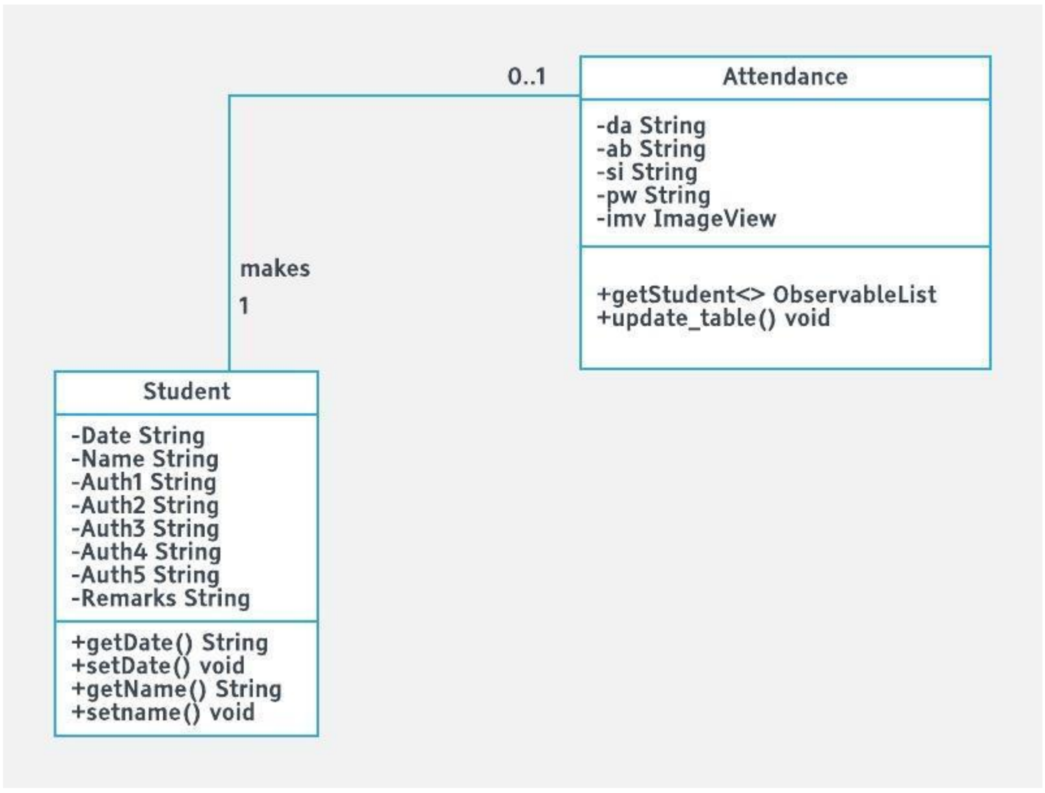


Figure 4.4. Class Diagram for User Interface

4.1.5. Database Design

Our system possess the following database design since it has only one table:

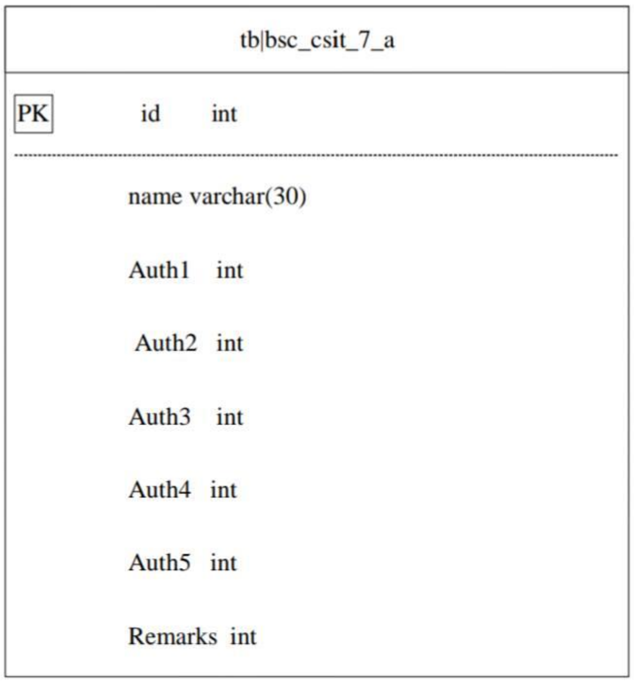


Figure 4.5. Database design

Chapter 5 - Implementation and Testing

5.1. Implementation

The main purpose of implementation of this system is to detect the person and perform the attendance. This project intends to implement a convolutional neural network which is trained using large number of data, used for testing, classifying and detecting the face of the person.

5.1.1. Analysis and Design Tools

- There is various presence of designing tools to create figures and diagrams like entity-relationship diagram, flow chart, use case diagram and other desired diagram. In this project Microsoft Visio Professional software was used for diagrammatic design of the proposed system.
- Use case diagram and activity diagram of the system is used to analyze system design. The figures and diagrams help to analyze the application and release the information required to maintain the interaction with the application. It also makes the development process easier and faster.

5.1.2. Implementation Tools

- The tool implemented for the programming logic of the system is Python. Front end of the system is developed by using Java Swing package. The front end or the interface is designed with GUI Builder’s design features of Java.
- Back end of the system is developed with python using the concept of Convolutional Neural Network (CNN).

5.1.3. Other Tools and Platforms

- GitHub

  GitHub is an online platform for code sharing and version controlling. In this project all the team member was form different residential location so, there was a necessity for distributed working environment and for that GitHub was used for sharing the code. Each team member would pull the project, add some functionality and push those changes to the central repository to the GitHub.

- NetBeans

  NetBeans is software development platform used as a code editor tool. The entire application was written in Java using NetBeans. NetBeans was also used for debugging, testing, compiling, and creating executable files of the source code.

- Sublime Text

  Sublime Text is a versatile and fun text editor for code and prose that automates repetitive tasks so you can focus the important stuff. It works on OS X, Windows and Linux. Sublime Text is a proprietary cross-platform source code editor with a Python API. It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses.

5.2. Methodology

In this proposed system, the system is instantiated automatically after the classes starts. After it triggers then the system starts processing the image for which we want to mark the attendance. Image capturing phase is one in which we capture the image using Android software i.e. Ip cam. This is basic phase from which we start initializing our system. First the image is captured after that haarcascade face classifier is used to detect student faces and save the faces in jpg format. From the saved image folder each and every images are feed into the trained model that is built in Convolutional Neural Network architecture and classify faces. In which first feature is extracted using convolutional and pooling layer after that the 3-layer depth image is flatten into two dimension. The flatten images is then feed into the multiple layer's neural network and uses the softmax Activation function to classify the image. According to the label of images the stored name of student is extracted from database and update the database as per the classified images. The detailed methodology of our system development is described as follows:

5.2.1. Data Collection and Processing

The data used in this project are the primary data ie. Images of an individual among which some of them were fed into the model for training the model and some were used for testing purpose. 150 images of each individual were collected using the android camera and after using filtering and data augmentation technique, 500 images for each person is obtained. The detailing of the data collection and modification is described below:

1. Manually capture the image of shape 3264x2448x3 by Android Camera

This are the sample images that are taken form Samsung Galaxy Note 3.

We captured images for 10 Days so that we can get different images of same people and collected 150 image of every student after filtering those images.

2. Second Step: Apply Data Augmentation Techniques

A. Rotation

The rotation is a geometric transformation that maps the position (x,y) of an image to position (‘x’,’y’) through certain angle θ about the origin.

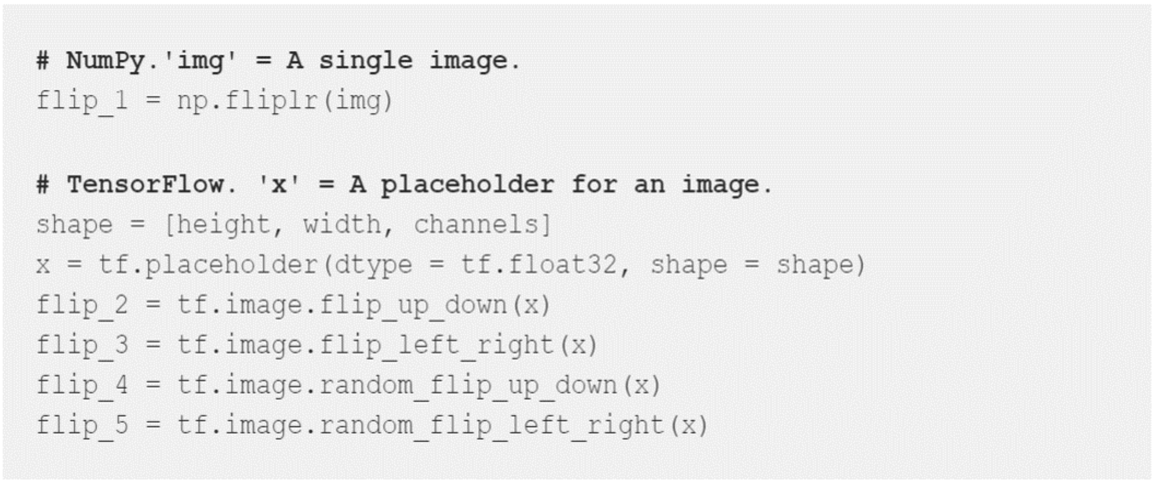
Code to rotate the image is

```
# Placeholders: 'x' = A single image, 'y' = A batch of images
# 'k' denotes the number of 90 degree anticlockwise rotations
shape = [height, width, channels]
x = tf.placeholder(dtype = tf.float32, shape = shape)
rot_90 = tf.image.rot90(img, k=1)
rot_180 = tf.image.rot90(img, k=2)

# To rotate in any angle. In the example below, 'angles' is in
radians
shape = [batch, height, width, 3]
y = tf.placeholder(dtype = tf.float32, shape = shape)
rot_tf_180 = tf.contrib.image.rotate(y, angles=3.1415)

# Scikit-Image. 'angle' = Degrees. 'img' = Input Image
# For details about 'mode', checkout the interpolation section
below.
rot = skimage.transform.rotate(img, angle=45, mode='reflect')
```

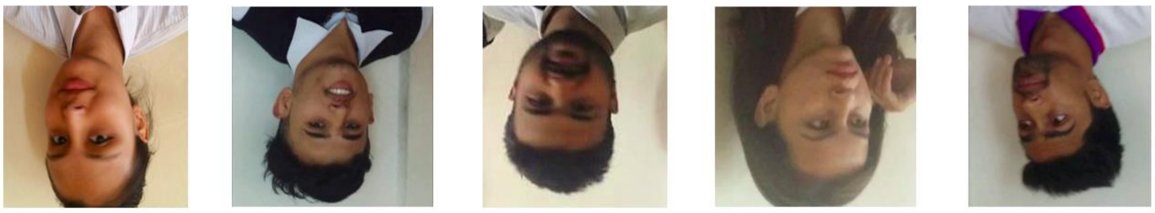
The result obtained after the rotation is

B. Flip

Flipping of images provides the mirror or reversed image of the original image so that we can view the image from different perspective. The code for flipping the images and the result obtained after flipping is:

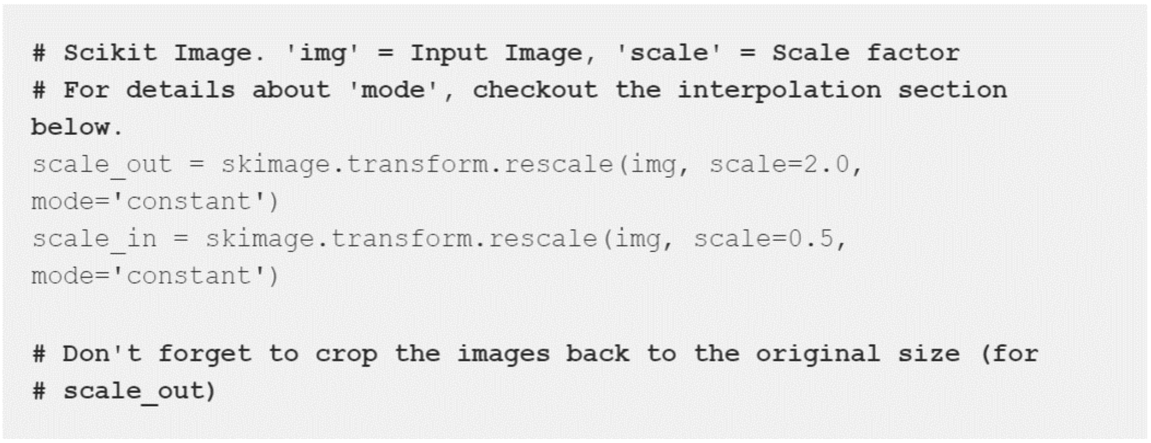


```
# NumPy.'img' = A single image.
flip_1 = np.fliplr(img)

# TensorFlow. 'x' = A placeholder for an image.
shape = [height, width, channels]
x = tf.placeholder(dtype = tf.float32, shape = shape)
flip_2 = tf.image.flip_up_down(x)
flip_3 = tf.image.flip_left_right(x)
flip_4 = tf.image.random_flip_up_down(x)
flip_5 = tf.image.random_flip_left_right(x)
```



C. Scale

Scaling helps in resizing of images in the desired size. The code for scaling the images is



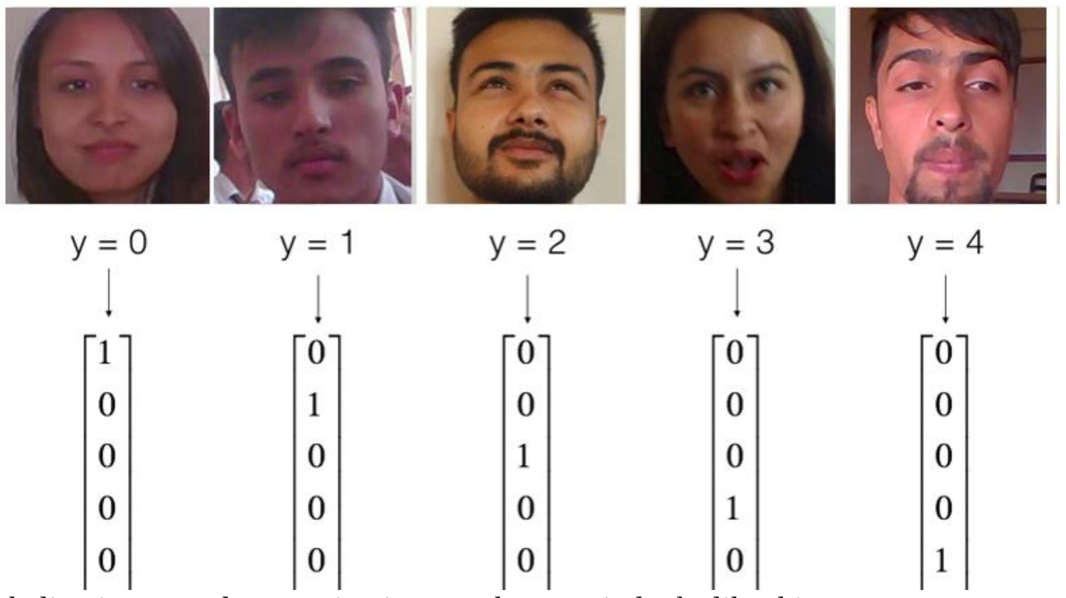
```
# Scikit Image. 'img' = Input Image, 'scale' = Scale factor
# For details about 'mode', checkout the interpolation section
below.
scale_out = skimage.transform.rescale(img, scale=2.0,
mode='constant')
scale_in = skimage.transform.rescale(img, scale=0.5,
mode='constant')

# Don't forget to crop the images back to the original size (for
# scale_out)
```

The result generated after scaling the images

Finally collected 1000 images of each student after applying lots of filtering process. Filtering process contains eliminating blurry image, image that contains lots of noise, darker image, and coloring effected images.

3. Save detected faces into specific folder

4. Finally selected 1000 image to feed into model so that model correctly learns every features. Since our system can't be more accurate because of lots of constraints. Deep Learning require lots of images to learn every possible features in the image. we have created model and feed 1000 images of each student. Since this is supervised learning first we label every image and convert it into one hot matrix.



Labeling image and converting into one hot matrix looks like this

5.3. Algorithm Implemented

5.3.1. Convolutional Neural Network

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a ‘m x m x r’ image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has ‘r=3’. The convolutional layer will have k filters (or kernels) of size ‘n x n x q’ where n is smaller than the dimension of the image and q can either be the same as the number of channels ‘r’ or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size ‘m−n+1’. Each map is then subsampled typically with mean or max pooling over ‘p x p’ contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs. Either before or after the sub sampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map.

The main motivation behind the emergence of CNNs in deep learning scenarios has been to address many of the limitations that traditional neural networks faced when applied to those problems. When used in areas like image classification, traditional fully-connected neural networks simply don’t scale well due to their dis-proportionally large number of connections. CNNs bring a few new ideas that contribute to improve the efficiency of deep neural networks.There exist many open source libraries like numpy, scipy, matplotlib, tensorflow etc. making it easy to implement CNNs without knowing the underlying theory.
The architecture of our CNN model can be illustrated as:

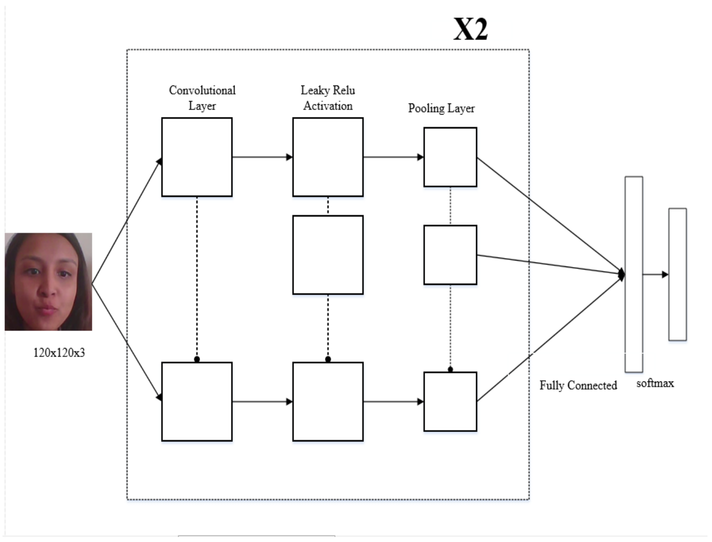


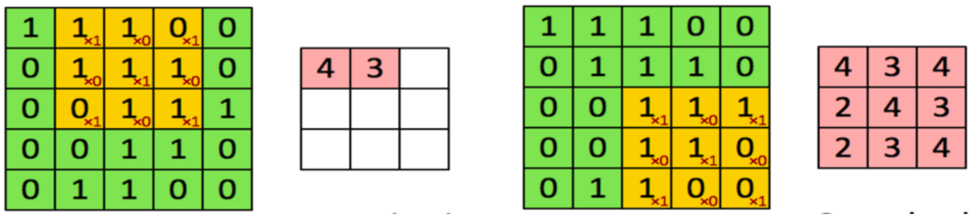
Figure 5.1. Convolutional Neural Network Architecture

5.3.2. Mathematical Model

Convolution Layer:

Convolution layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. Our CNN model contains 2 Convolutional Layer, 2 leaky Relu Activation.

The following example shows how convolutional layer works.



The formulas relating the output shape of the convolution to the input shape is:

$$n_H = \lfloor (n_{Hprev} - f + 2 \times pad)/stride \rfloor + 1$$
$$n_W = \lfloor (n_{Wprev} - f + 2 \times pad)/stride \rfloor + 1$$
$$NC = \text{number of filters used in the convolution}$$

Leaky RELU layer:

Leaky RELU layer will apply an element wise activation function, such as the max (, x) thresholding at zero.

Pooling Layer:

Pooling layers are developed to reduce the number of parameters needed to describe layers deeper in the network. These layers provide some limited amount of translational or rotational invariance. The pooling (POOL) layer reduces the height and width of the input. It helps reduce computation, as well as helps make feature detectors more invariant to its position in the input. The two types of pooling layers are:

- Max-pooling layer: This layer slides an (f, f) window over the input and stores the max value of the window in the output.
- Average-pooling layer: this layer slides an (f, f) window over the input and stores the average value of the window in the output.

As there's no padding, the formulas binding the output shape of the pooling to the input shape is:

$$n_H = \lfloor (n_{H_{prev}} - f)/stride \rfloor + 1$$

$$n_W = \lfloor (n_{W_{prev}} - f)/stride \rfloor + 1$$

$$n_C = n_{C_{prev}}$$

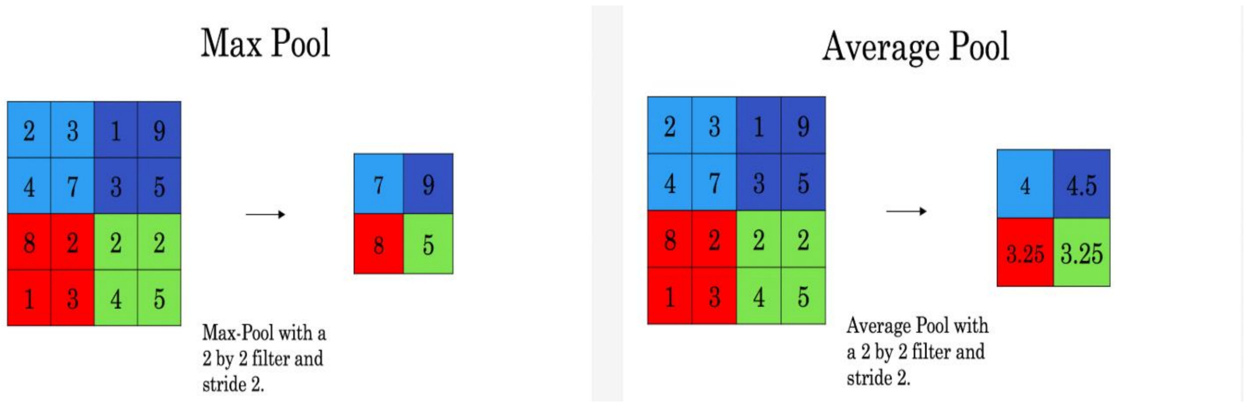
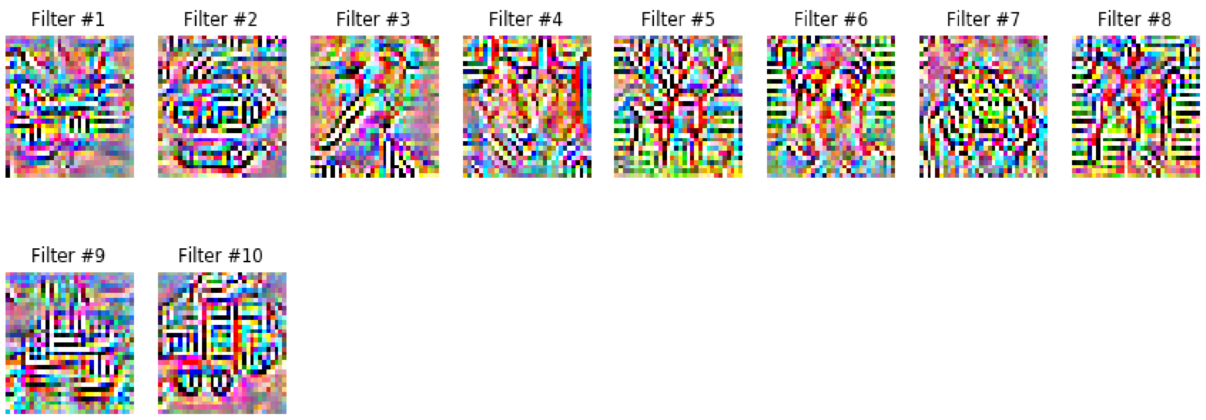
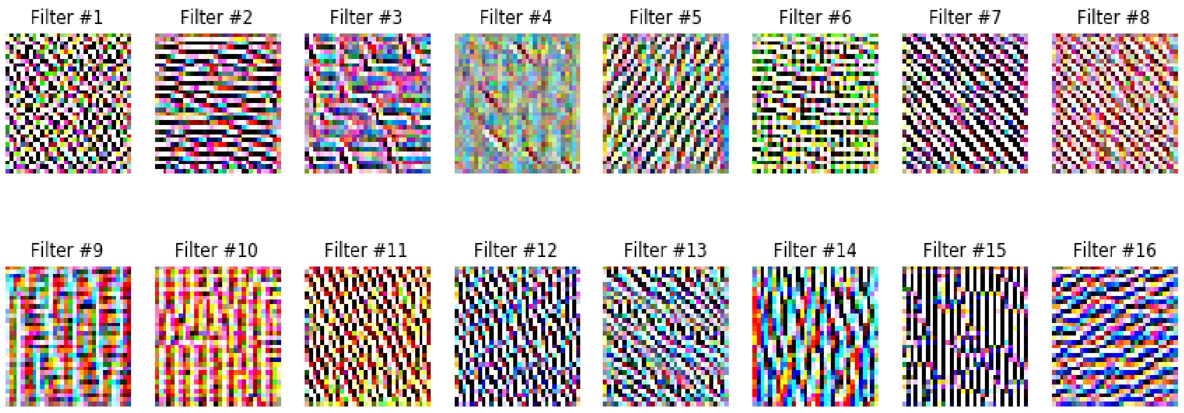


Figure 4.2. Max pooling and Average Pooling

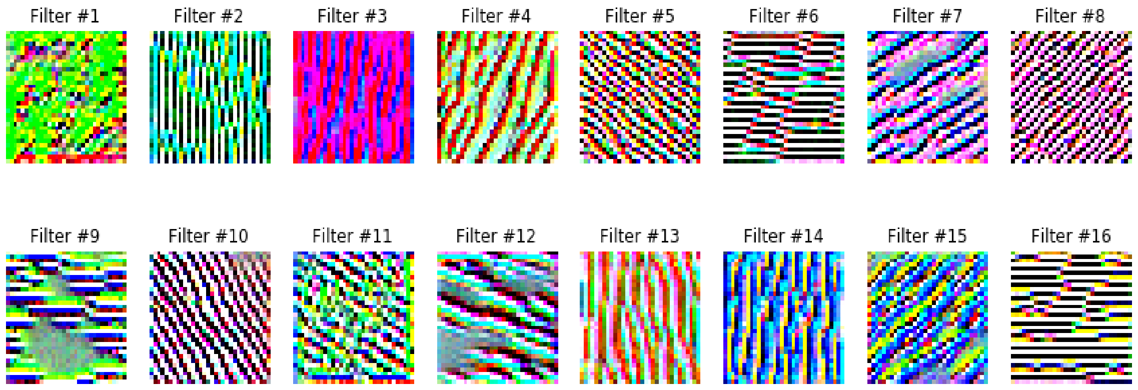
Features learned by first convolutional layer



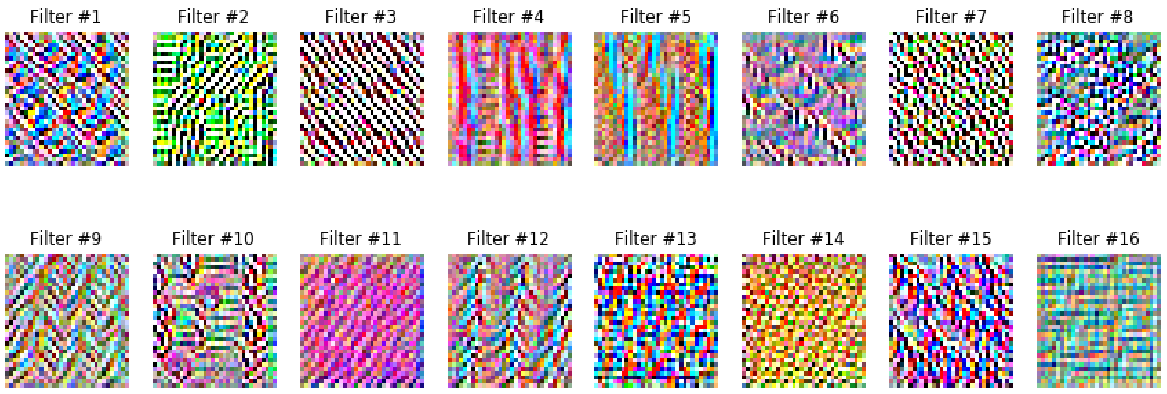
Features learned by first pooling layer



Features learned by Second Convolution layer



Features learned by second pooling layer



Fully-connected layer:

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. In practice, convolutional layers are used to learn a feature hierarchy and one or more fully connected layers are used for classification purposes based on the computed features. In general, the convolution and fully connected layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU and Pooling layers will implement a fixed function. The parameters in the convolutional and fully connected layers will be trained with gradient descent so that the class scores that the convolutional network computers are consistent with the labels in the training set for each image.

5.3.3. Tensorflow

For implementing Convolutional Neural Network (CNN) we have used some of the modules defined under an open source library called Tensorflow. Tensorflow is used or high performance computations and it comes with strong support for machine learning and deep learning.

For creating a different layers of Convolutional Neural Network we have used the following functions:

1. tf.nn.conv2d

This function s defined in a generated file; tensorflow/python/ops/gen_nn_ops.py.

It computes 2D convolution given 4D input and tensor filters. The function takes the following arguments:

tf.nn.conv2d(input, filter, stride, padding, use_cudnn_on_gpu=’True’,name)

- input: It is a Tensor. It must be of types: float32, bfloat16, float64, half.
- Filter: A 4-D tensor of shape [filte_height, filter_width, in_channels, out_channels]
- stride: it is a list of ints, 1-D tensor of length 4.
- padding: it denotes which padding algorithm to use. A string from “SAME”, “VALID”.
- use_cudnn_on_gpu: an optional bool type argument which defaults to true.
- Name: a name for the operation which is optional.

This function returns the Tensor which has the same type as input.

We’ve used this function two times since our model has two convolutional layer.

2. tf.nn.leaky_relu:

It is defined under tensorflow/python/ops/nn_ops.py. It computes the Leaky ReLU activation function. This takes the following arguments:

tf.nn.leaky_relu(features, alpha=0.2,name=none)

- features: A tensor representing preactivation values. It must of type float16, float32, float64, int32, int64.
- alpha: slope of the activation function at x<0.
- name: name of the operation and is optional.

This function returns activation value. This function is also used for two times for two convolution layer.

3. tf.nn.max_pool:

It is defined under tensorflow/python/ops/nn_ops.py. We have used max pooling technique for the pooling operation. This function performs max pooling in the input. The function takes the follwing arguments:

tf.nn.max_pool(value, ksize, stride, padding)

- value: A 4-D tensor of the format specified by data_format.
- Ksize: A list or tuple of 4 ints. The size of the window for each dimension of the input tensor.
- Stride: A list or tuple of 4 ints. The stride of sliding window for each dimension of the input tensor.
- Padding: The padding algorithm represented by string either "SAME" or "VALID".

This function returns max pooled output tensor. Since we have two pooling layers, we have used this function for two times.

4. tf.contrib.layers.fully_connected

This function adds a fully connected layer. It takes the following arguments:

tf.contrib.layers.fully_connected(inputs, num_outputs, activation_fn)

- inputs: A tensor of with at least rank 2 and vlaue for the last dimension ie. [batch_size, depth], [None, None, None, channels].
- num_outputs: The number of output units in a layer which are of type Integer or Long.
- activation_fn: denotes activation function. We can set it to none and maintain linear activation.

It returns the tensor variable representing the result of the series of operations.

5. tf.contrib.layers.flatten

This function is defined in tensorflow/contrib/layers/python/layers/layers.py. This flattens the input while maintaining the batch_size. The function takes the following arguments:

tf.contrib.layers.flatten(inputs, outputs_collections=None, scope=None)

- inputs: It denotes the tensor of size [batch_size, …]
- outputs_collections: collection to add the outputs
- scope: an optional scope for name

This function returns a flattened tensor with size [batch_size,k].

The implementation code for these five functions in our model is:

```
def model(X_train, Y_train,X_test,Y_test,im_Size,names,learning_rate=0.005,
num_epochs=65, minibatch_size=64, print_cost=True):
        (m,n_H0,n_W0,n_C0)=X_train.shape
        n_y = Y_train.shape[1]
        costs=[]
        X,Y=create_placeholders(n_H0,n_W0,n_C0,n_y,im_Size)
        seed=3
        parameters=initialize_parameters()
        #forward propogatation
        W1=parameters["W1"]
        W2=parameters["W2"]
        Z1=tf.nn.conv2d(X,W1,strides=[1,1,1,1],padding="SAME")
        A1=tf.nn.leaky_relu(Z1)
        P1=tf.nn.max_pool(A1,ksize =[1,8,8,1],strides=[1,8,8,1],padding="SAME")
        Z2=tf.nn.conv2d(P1,W2,strides=[1,1,1,1],padding="SAME")
        A2=tf.nn.leaky_relu(Z2)
        P2=tf.nn.max_pool(A2,ksize=[1,4,4,1],strides=[1,4,4,1],padding="SAME")

        P = tf.contrib.layers.flatten(P2)
        Z3 = tf.contrib.layers.fully_connected(P, 5, activation_fn=None)
        cost=compute_cost(Z3,Y)
```

5.3.4. Cost Function

The cost function is used to return a number representing how well the neural network performed to map training examples to correct output. It measures the difference between estimator (the datasets) and estimated value. While calculation a cost function we have an hypothesis as,

$$h_\theta(x)= \theta_0+\theta_1 x$$

where θ_0 and θ_1 are the model parameters which we have to choose so that the value for $h_\theta(x)$ is close to the y for our training example (x,y).
The next is the formula for the squared error cost function which can be represented as,

$$1/2m(\Sigma^m_{i=1} (h_\theta(x^{(i)})-y^{(i)})^2)$$

where, m is the size of training set and ½ remains constant which means minimizing one and half of something. $\Sigma^m_{i=1}$ refers to the sum from I to m. we repeat the calculation to the right of sigma,

$$(h_\theta(x^{(i)})-y^{(i)})^2$$

for each sample. The actual calculation is just the hypothesis value for h(x), minus the actual value of y. Then you square whatever you get. The final result will be a single number. We repeat this process for all the hypothesis, in this case best_fit_1 , best_fit_2… best_fit_m. Whichever has the lowest result, or the lowest "cost" is the best fit of the three hypothesis.

5.4. Testing

5.4.1Unit Testing

Unit testing is the level of software testing of the project's system in which the smallest testable parts of a system called unit is individually tested. Unit testing concentrates on each unit of the system as implemented in the source code. The main purpose of unit testing is to validate each unit of the software to perform as designed.

Table 5.1. Test case 1

S.N.	Test Unit	Test Input Data	Expected Results	Output	Remarks
1	Initiate IP-cam to capture image	IP of the device	IP-cam should initialize	IP-cam was initialized	Successful initialization
2	Initiate IP cam to capture image	IP of the device	IP-cam should be initialized	Not initialized	Unsuccessful initialization

Table 5.2. Test case 2

S.N.	Test Unit	Test Input Data	Expected Results	Output	Remarks
1	Restore trained model	CNN model	CNN model should reload successfully	Loaded the model and process flow to next step	Successful
2	Restore trained model	CNN model	CNN model should reload successfully	Error message	Unsuccessful restoration of model

Table 5.3. Test case 3

S.N.	Test Unit	Test Input Data	Expected Results	Output	Remarks
1	Classify and detect face	Test image data	The trained model should correctly detect the face of individual	Face detected	Successful detection of faces

2	Classify and detect face	Test image data	The trained model should correctly detect the face of individual	Face not detected	Unsuccessful detection of faces

Table 5.4. Test case 4

S.N.	Test Unit	Test Input Data	Expected Results	Output	Remarks
1	Update database	Results of detected faces	The detected faces name should be stored in database	Stored and updated the database	Successful
2	Update database	Results of detected faces	The detected faces name should be stored in database	Database was not updated	Unsuccessful

5.4.2 System Testing

System Testing is a level of the software testing where a complete software is tested to know whether the system works as per the defined objectives or not.

The test case description for the system testing is described below.

Test case: Execute the whole system

Description: All the modules of the system should work properly so that the correct and accurate detection of the face is possible.

Test case input: trained CNN model, test image data of an individual

Expected output: the face should be detected correctly and the results obtained should be stored in the database automatically.

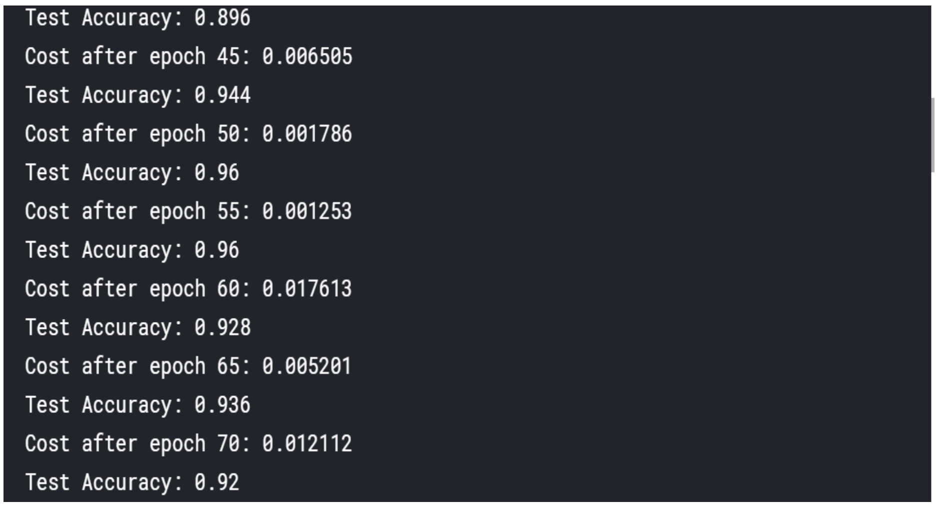
Actual Output: The system detected the face and the database is updated with the new result.

Remarks: Successful execution of the system

Chapter 6 - Conclusion and Recommendation

6.1. Conclusion

‘Attendance System Using Face Recognition’ was developed using the CNN. The data we used in this project are primary data i.e. images of the person. The image data called training data are fed into the model so as to train the model. The trained model was saved and restored at the time of implementation. For providing the input IP cam was used which capture the image and the model classifies and detects the image. For accurate detection we have used five time verification in which verifies the image five times. On completing this our system is now able to detect the face of an individual on the basis of provided training data with average of 90% of accuracy, and update the database. The confidence level of accuracy can be justified through the following results and curve that we obtained after each iterations while testing for the accuracy of the system with the learning rate of 0.005.



```
Test Accuracy: 0.896
Cost after epoch 45: 0.006505
Test Accuracy: 0.944
Cost after epoch 50: 0.001786
Test Accuracy: 0.96
Cost after epoch 55: 0.001253
Test Accuracy: 0.96
Cost after epoch 60: 0.017613
Test Accuracy: 0.928
Cost after epoch 65: 0.005201
Test Accuracy: 0.936
Cost after epoch 70: 0.012112
Test Accuracy: 0.92
```

Figure 6.1. Accuracy of the system after each iterations

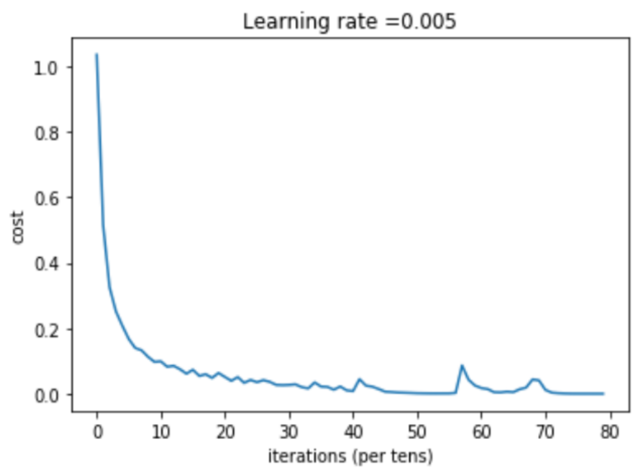


Figure 6.2. Curve showing the cost of each iterations with 0.005 learning rate

6.2. Recommendations

The data used for training the model are all the images of each of the individual. So for further improvement or to make more accurate we need images as much as possible and for detecting new face we need to train model for that face too. The test accuracy is not as consistent as shown in figure 6.1 because while testing the system in real environment the lighting of the room, photo quality and several other factors may affect the systems performance, thus causing the fluctuation of the accuracy. This is also because of the limitation of the resource we need to run the system. Therefore, we need as much as possible images, good quality photos and resources in order to run the system smoothly and more accurately.

References

[1] T. Lim, S. Sim, and M. Mansor, “RFID based attendance system,” in Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium on, vol. 2. IEEE, 2009, pp. 778–782.

[2] B. K. Mohamed and C. Raghu, “Fingerprint attendance system for classroom needs,” in India Conference (INDICON), 2012 Annual IEEE. IEEE, 2012, pp. 433–438.

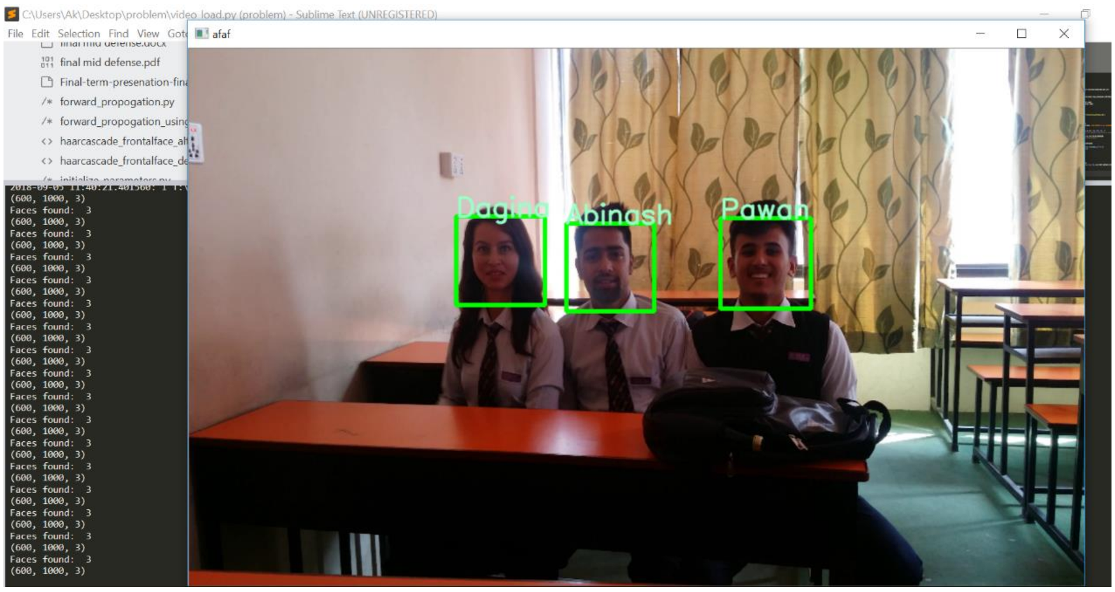
[3] S. Kadry and K. Smaili, “A design and implementation of a wireless iris recognition attendance management system,” Information Technology and control, vol. 36, no. 3, pp. 323–329, 2007

https://www.tutorialspoint.com

https://www.tensorflow.org/api_docs/python

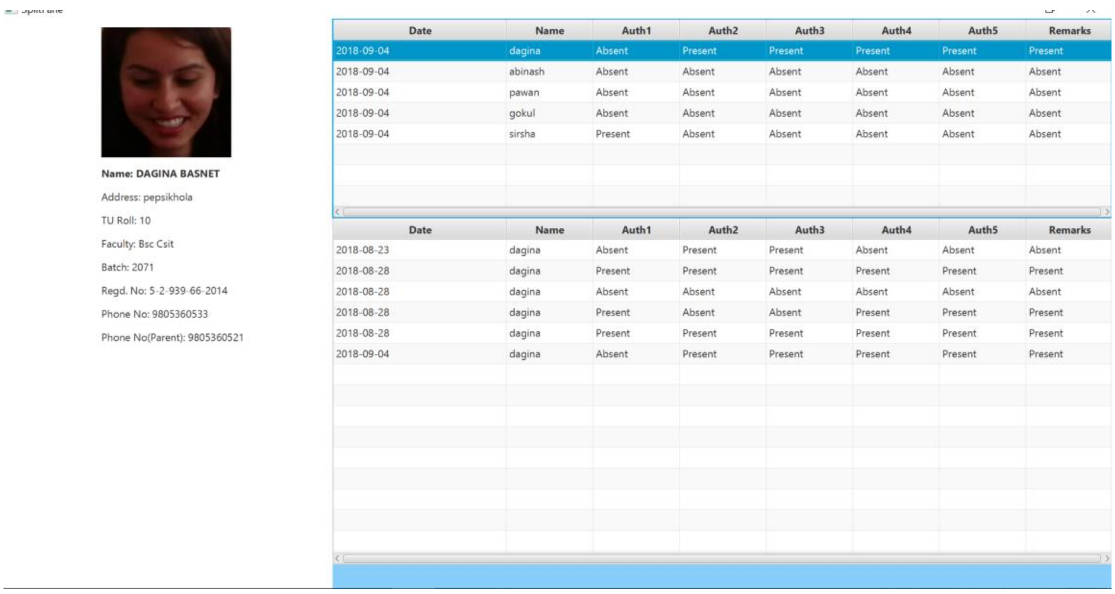
Appendices

1. Screenshot of face detection


C:\Users\Ak\Desktop\problem\video_load.py (problem) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto
afaf
final mid defense.pdf
Final-term-presenation-fin
forward_propogation.py
forward_propogation_using
haarcascade_frontalface_al
haarcascade_frontalface_de
Dagina
Abinash
Pawan

```
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
Faces found:  3
(600, 1000, 3)
```

2. Screenshot of UI



Name: DAGINA BASNET

Address: pepsikhola

TU Roll: 10

Faculty: Bsc Csit

Batch: 2071

Regd. No: 5-2-939-66-2014

Phone No: 9805360533

Phone No(Parent): 9805360521

Date	Name	Auth1	Auth2	Auth3	Auth4	Auth5	Remarks
2018-09-04	dagina	Absent	Present	Present	Present	Present	Present
2018-09-04	abinash	Absent	Absent	Absent	Absent	Absent	Absent
2018-09-04	pawan	Absent	Absent	Absent	Absent	Absent	Absent
2018-09-04	gokul	Absent	Absent	Absent	Absent	Absent	Absent
2018-09-04	sirsha	Present	Absent	Absent	Absent	Absent	Absent

Date	Name	Auth1	Auth2	Auth3	Auth4	Auth5	Remarks
2018-08-23	dagina	Absent	Present	Present	Absent	Absent	Absent
2018-08-28	dagina	Present	Present	Present	Present	Present	Present
2018-08-28	dagina	Absent	Absent	Absent	Absent	Absent	Absent
2018-08-28	dagina	Present	Absent	Absent	Present	Present	Present
2018-08-28	dagina	Present	Present	Present	Present	Present	Present
2018-09-04	dagina	Absent	Present	Present	Present	Present	Present

3. Code for creating model

```
import tensorflow as tf
from prepare_dataset_for_images import *
import numpy as np
from scipy import ndimage
import scipy.misc
import numpy as np
import shutil
import os
import numpy as np
import time
from insert_data import *
from insert_data_individual import *
from delete_record import *
#
X_test,Y_test,names=prepare_dataset_for_images(25,75,5,"C:/Users/AK/Des
ktop/problem/Test_Data/")
from taking_input import *
from taking_input1 import *
X=np.zeros((5,5,5))
names=["Dagina","Abinash","Pawan","Gokul","Sirsha"]
haar_face_cascade=cv2.CascadeClassifier("./haarcascade_frontalface_default.
xml")
sess=tf.Session()
signature_key =
tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_D
EF_KEY
input_key = "images"
output_key = "scores"
export_path =  "./savemodel"
meta_graph_def =
tf.saved_model.loader.load(sess,[tf.saved_model.tag_constants.SERVING],ex
port_path)
signature = meta_graph_def.signature_def
x_tensor_name = signature[signature_key].inputs[input_key].name
y_tensor_name = signature[signature_key].outputs[output_key].name
x = sess.graph.get_tensor_by_name(x_tensor_name)
y = sess.graph.get_tensor_by_name(y_tensor_name)
def predict(X,index,middle):

        folder="./faces/"
        files=os.listdir(folder)
        for i in range(len(files)):
                path=folder+files[i]
                image=np.array(ndimage.imread(path,flatten=False))
                if image.shape[2]>3:
                        image = cv2.cvtColor(image,
cv2.COLOR_BGRA2BGR)
                # print(image.shape)

        image=scipy.misc.imresize(image,size=(75,75)).reshape(1,75*75*3)
```

```
				image=np.reshape(image,(image.shape[0],75,75,3))
				image=(image/255)-0.5
				y_out = sess.run(y, {x: image})
				correct_prediction=tf.argmax(y_out,1)
				value=sess.run(correct_prediction)
				value=np.squeeze(value)
				# print(value)
				if value==0:
					print(names[0])
					# dagina[0,inxex]=1
					X[0,middle,index]=1
				if value==1:
					print(names[1])
					# abinash[0,index]=1
					X[1,middle,index]=1
				if value==2:
					print(names[2])
					# pawan[0,index]=1
					X[2,middle,index]=1
				if value==3:
					print(names[3])
					# gokul[0,index]=1
					X[3,middle,index]=1
				if value==4:
					print(names[4])
					# sirsha[0,index]=1
					X[4,middle,index]=1
		print(X[0,0,:])
		print(X[1,0,:])
		print(X[2,0,:])
		print(X[3,0,:])
		print(X[4,0,:])

def extract_face(path):
		try:
				shutil.rmtree('./faces')
				os.makedirs('./faces')
		except:
				os.makedirs('./faces')
		image=cv2.imread(path)
		image2=image.copy()
		faces = haar_face_cascade.detectMultiScale(image, scaleFactor=1.2,
minNeighbors=5);  #1.3
		j=0
		#print the number of faces found
		print('Faces found: ', len(faces))
		#go over list of faces and draw them as rectangles on original colored
		for (x, y, w, h) in faces:
				cv2.rectangle(image, (x-10, y-10), (x+w+10, y+h+10), (0,
255, 0), 4)
```

```
                        try:
                                check=image2[y-10:y+10+h,x-10:x+w+10].shape
                                if check[0]>75:

        plt.imsave("./faces/"+str(j)+".jpg",cv2.cvtColor(image2[y-10:y+10+h,x-
10:x+w+10],cv2.COLOR_BGR2RGB))
                                        j=j+1
                        except:
                                print("error message")
                                continue
def load_model_and_predict(middle):
        for i in range(5):
                path="./frames/capture"+str(i)+".jpg"
                print("times: ",i)
                extract_face(path)
                predict(X,i,middle)

        print("The final result is ")
def load_model():
        for i in range(5):
                for j in range(5):
                        taking_input1(j)
                        time.sleep(2)
                print("5 image store now computing")
                load_model_and_predict(i)
load_model()
print(X)
delete_record()
X=np.array(X)
index=0
for i in X:
print("Value")
value=np.count_nonzero(i== 1,axis=1)
# print(value)
for j in range(5):

if(value[j]>2):
value[j]=1
else:
value[j]=0
print(value)
insert_data(names[index].lower(),int(value[0]),int(value[1]),int(value[2]),int(value[3]
),int(value[4]))
insert_data_individul(names[index].lower(),int(value[0]),int(value[1]),int(value[2]),in
t(value[3]),int(value[4]))

index=index+1
```