

AUTOMATED ATTENDANCE SYSTEM USING FACE RECOGNITION

BY

V.ROHIT SAI (Reg No:16131A05N0)

Y.V.YAMINI (Reg No:16131A05O1)

S.LEELA SIRISHA (Reg No:16131A05L1)

Y.DEEPIKA (Reg No:17135A0547)

Under the Esteemed Guidance of

G. HINDUMATI
M.TECH, ASSISTANT PROFESSOR



Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (Autonomous)

Approved by AICTE, New Delhi and Affiliated to JNTU-Kakinada

Re-accredited by NAAC with "A" Grade with a CGPA of 3.47/4.00

Madhurawada, Visakapathanam-530 048

Year of submission:2019

AUTOMATED ATTENDANCE SYSTEM USING FACE RECOGNITION

Project thesis submitted in partial fulfilment of mini project for VII semester
B. Tech in Computer Science and Engineering

BY

V.ROHIT SAI (Reg No:16131A05N0)

Y.V.YAMINI (Reg No:16131A05O1)

S.LEELA SIRISHA (Reg No:16131A05L1)

Y.DEEPIKA (Reg No:17135A0547)

Under the Esteemed Guidance of

G.HINDHUMATI

M.TECH, ASSISTANT PROFESSOR



Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (Autonomous)

Approved by AICTE, New Delhi and Affiliated to JNTU-Kakinada

Re-accredited by NAAC with “A” Grade with a CGPA of 3.47/4.00

Madhurawada, Visakapathanam-530 048

Certificate

This is to certify that the project thesis entitled AUTOMATED ATTENDANCE SYSTEM

USING FACE RECOGNITION being submitted by

V.ROHIT SAI	(Reg No:16131A05N0)
Y.V.YAMINI	(Reg No:16131A05O1)
S.LEELA SIRISHA	(Reg No:16131A05L1)
Y.DEEPIKA	(Reg No:17135A0547)

In partial fulfillment for the mini project in Computer Science and Engineering to the Jawaharlal Nehru Technological University Kakinada, Kakinada is a record of bonafide work carried out under my guidance and supervision.

The results embodied in this project thesis have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Mrs. G. HINDHUMATI

M.Tech , Assitant Professor

Dr. P. KRISHNA SUBBA RA0

Head of the Department of
Computer Science & Engineering

ACKNOWLEDGEMENT

We would like to take this opportunity to extend our hearty gratitude to our esteemed institute “**Gayatri Vidya Parishad College of Engineering (Autonomous)**” where we got the platform to fulfill our cherished desire.

We express our deep sense of gratitude to **Dr .P.KRISHNA SUBBA RAO**, Prof and Head of the Department, Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering (Autonomous) for giving us an opportunity to do the project in college and for his constant encouragement.

We wish to express our sincere thanks to **Dr.A.B KOTESWARA RAO**, Prof and Principal of GayatriVidyaParishad College of Engineering (Autonomous), for his support and encouragement during the course of the project.

We are obliged to **Mrs.G.HINDHUMATI**, Asst.Prof ,Department of Computer Science and Engineering, who has been our guide, whose valuable suggestions , guidance and comprehensive assistance helped us a lot in realizing the project.

We also thank **Mrs. V. TULASI**, Associate Professor, and coordinator of the projects, Department of computer science and engineering, for guiding us throughout the project and helping us in completing our project efficiently.

Lastly, we are grateful to all our friends, for their relentless support in augmenting the value of work, our family, for being considerate and appreciative throughout.

Project Members:

V.ROHIT SAI

Y.V.YAMINI

S.LEELA SIRISHA

Y.DEEPIKA

ABSTRACT

The project is about the attendance management idea using face recognition technology which will replace the manual method of taking attendance and also overcomes difficulties in maintenance. The proposed system requires no papers and no attendance registers. All it needs is a camera and an algorithm that runs on the photo that is captured that recognises all the faces which are trained with the algorithm prior to the implementation of the system. The system updates an excel sheet that stores the attendance of students in a class everyday and every period. Since this a view of practical implementation , the algorithm is implemented using python since a high rate of accuracy is anticipated.

The proposed automated attendance system requires sound knowledge and skill in the following areas that belong to the field of Computer Science like Artificial Intelligence, Image Processing, Deep Learning and Python . This system can maintain the attendance up to date and it can be retrieved whenever required just by accessing the recorded data in the form of excel sheet. This also overcomes issues like noting the data incorrectly , proxy attendance and errors in manual updation to a greater extent and can also reduce the time and work required to analyse or report the status of attendance of students.

Table of contents

S.NO	CONTENTS	Pg No
1	INTRODUCTION	1
2	SYSTEM STUDY 2.1 EXISTING SYSTEM 2.2 DISADVANTAGE EXISTING SYSTEM 2.3 PROPOSED SYSTEM 2.4 ADVANTAGE OF PROPOSED SYSTEM 2.5 FEASIBILITY STUDY	4
3	SOFTWARE REQUIREMENT ANALYSIS 3.1 PROBLEM STATEMENT 3.2 DEEP LEARNING 3.3 FACE RECOGNITION MODULE	8
4	SOFTWARE DESIGN 4.1 UML DIAGRAMS 4.1.1 COLLABORATION DIAGRAM 4.1.2 SEQUENCE DIAGRAM 4.1.3 USECASE DIAGRAM	21
5	SOFTWARE REQUIREMENT SPECIFICATION 5.1 FUNCTIONAL REQUIREMENTS 5.1.1 SOFTWARE REQUIREMENTS 5.1.2 HARDWARE REQUIREMENTS 5.2 NON FUNCTIONAL REQUIREMENTS	26
6	CODING	27
7	TESTING AND OUTPUT SCREENS	45
8	CONCLUSION	37

9	FUTURE WORK	38
10	REFERENCES	40

1.INTRODUCTION

In this world, a school or a college is considered as the second home where a lot of students come, learn and apply the knowledge what they have gained during their study in their respective school or college. This is a place where students come and learn the concepts that are taught by their teacher. As a proof of the students' presence in the class in that particular school attendance system has been introduced . Till today , schools and colleges maintain attendance registers and attendance books for each and every class . They take the attendance by calling out the roll numbers of the people serial wise . This is practiced even today .

There are a lot of tedious tasks that involve in maintaining the attendance of the students especially when it comes to colleges and schools that have more number of students per class and more sections. This kills the time and thereby reducing the duration of the period that a lecturer has to tackle. It is evident that a teacher can complete her syllabus faster if the time that is taken to take the attendance is saved.

Researches say that on an average 8 minutes of time can be saved by automating the attendance system . This may seem effective if it is viewed in a broader perspective. It means that a teacher can utilize this time everyday in completing her lecture or making it as a doubts clearing session which helps the students .Along with this there is another point that has to be noticed. Manual attendance system takes time of the teachers in recording and analyzing the students' attendance regularly. There is scope of incorrect entry of data in the registers due to various reasons like misplacement of registers mistaking a present for absent and such problems.

There have been a lot of advancements in the field of Computer Science and Engineering which provide solution to various kinds of problems and made lives easier. In a

more similar way there is a solution for automating this attendance system. Elaborating the above statement, we can use latest software and most suited digital technologies in order to achieve our goal. In the context of this project it is Deep Learning that provides a feasible solution for the problem. The detailed description of the problem and the approach are explained in the further sections. The proposed methodology in this report is a step forward to make the attendance automated.

Coming to the idea of the project, it is viewed as a system that takes the photo of the people in the class recognizes the faces of the people and marks the attendance as present for the people whose faces are recognized in the photograph and absent for the people whose faces are not. This sounds to be simple but there are some kinds of concepts and programming languages that are used to achieve this goal. The details are elaborated in the software requirements section. There are a lot of methodologies that are proposed in order to achieve this goal. But the most feasible and efficient algorithm is required since attendance of every single student is equally important.

SCOPE FOR EXTENSION:

We can further extend this project by using raspberry pi and can make it into an IoT project. The beauty of this thought is that it can run on any device or computer with minimum RAM it doesn't require any GPU's or any high performance system. It can run on normal machine for example it can be used in way that take input from a normal camera and we can get a list of the persons who are in that class . Raspberry pi is nothing but a computer without any input and output devices . It has to be supplied with power no less than 4.5V . There is an SD card slot for this device that has the operating system of this device .This operating system is called Raspbian. It has ports for connecting camera module, Ethernet cable, USB cables,

microphone jack for audio input and output and a port for connecting HDMI cable. This is how it looks.



Figure 1.1 Raspberry pi

Since this is no less than any other operating system, it can be affixed on the top of the class room along with the camera module and the algorithm is made to run on this device. The algorithm can also be modified so that this obtained output can be stored in cloud to improve portability and independence on another operating systems like windows or linux. This also ensures effective retrieval of the data. The data can be retrieved in any system if the user is authorized to access the cloud. This is a full scope implementation of the idea that makes the attendance system almost automated.

2. SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

- Manually taken using paper or file based approach or some have adopted methods of automatic attendance using biometric techniques.

2.2. DISADVANTAGE OF EXISTING SYSTEM

1. System is fully dependent on skilled individuals,
2. Inconsistency in data entry and generate errors.
3. Time consuming and costly to produce reports.
4. Duplication of data entry

2.3 PROPOSED SYSTEM

Our system uses face recognition approach for the automatic attendance of students in the class room environment without students' intervention.

- Security of data.
- Ensure data accuracy.
- Minimal manual data entry.
- Greater efficiency.
- Better service.
- Minimum time required

2.4. ADVANTAGE OF PROPOSED SYSTEM

1. Less time consuming,

2. More secure than traditional system.

3. 24 x 7 availability of information

2.5 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features:

TECHNICAL FEASIBILITY

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within the latest technology. Though the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So, there are minimal constraints involved with this project. The system has been developed using Python the project is technically feasible for development.

ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is minimal cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is behaviourally feasible.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 PROBLEM STATEMENT

The problem is to implement an effective algorithm to recognize the faces of the people in an image. There should be an algorithm that takes the photos of the people registered to sit in that class and the algorithm is trained with this data. Later the same methods are used in recognizing the test by using the data that is generated from the trained images. The algorithm implemented in this paper uses deep learning.

3.2 DEEP LEARNING

Introduction to Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. Deep learning algorithms are constructed with connected layers. Deep learning is a subset of machine learning which is again a subset of artificial intelligence. So this is again an application of AI.

For a network, we need two neurons. These neurons transfer information via synapse between the dendrites of one and the terminal axon of another.

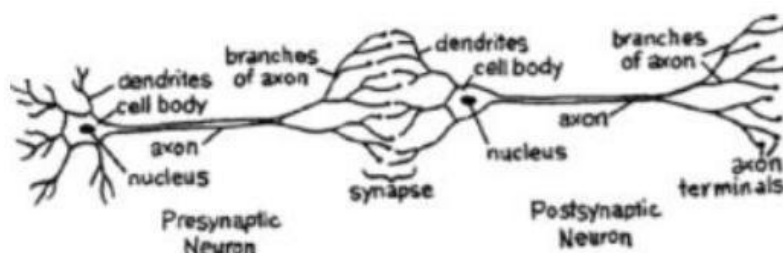


Fig 2.9 communication between neurons

A probable model of an artificial neuron looks like this –

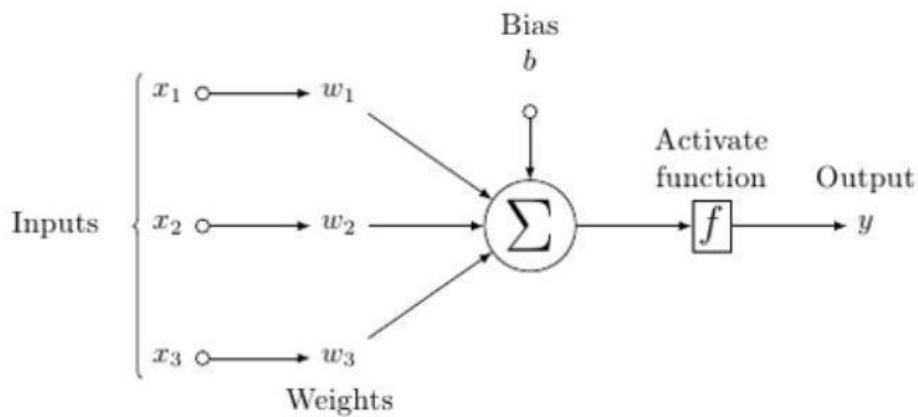


Fig 2.10 Diagram of an artificial neuron

The circles are neurons or nodes, with their functions on the data and the lines/edges connecting them are the weights/information being passed along. Each column is a layer. The first layer of your data is the input layer. Then, all the layers between the input layer and the output layer are the hidden layers. If you have one or a few hidden layers, then you have a shallow neural network. If you have many hidden layers, then you have a deep neural network. In this model, you have input data, you weigh it, and pass it through the function in the neuron that is called threshold function or activation function.

Basically, it is the sum of all of the values after comparing it with a certain value. If you fire a signal, then the result is (1) out, or nothing is fired out, then (0). That is then weighted and passed along to the next neuron, and the same sort of function is run.

We can have a sigmoid (s-shape) function as the activation function.

As for the weights, they are just random to start, and they are unique per input into the node/neuron.

In a typical "feed forward", the most basic type of neural network, you have your information pass straight through the network you created, and you compare the output to what you hoped the output would have been using your sample data.

From here, you need to adjust the weights to help you get your output to match your desired output.

The act of sending data straight through a neural network is called a feed forward neural network.

Our data goes from input, to the layers, in order, then to the output. When we go backwards and begin adjusting weights to minimize loss/cost, this is called back propagation.

This is an optimization problem. With the neural network, in real practice, we have to deal with hundreds of thousands of variables, or millions, or more.

The first solution was to use stochastic gradient descent as optimization method. Now, there are options like AdaGrad, Adam Optimizer and so on. Either way, this is a massive computational operation. That is why Neural Networks were mostly left on the shelf for over half a century. It was only very recently that we even had the power and architecture in our machines to even consider doing these operations, and the properly sized datasets to match.

For simple classification tasks, the neural network is relatively close in performance to other simple algorithms like K Nearest Neighbors. The real utility of neural networks is realized when we have much larger data, and much more complex questions, both of which outperform other machine learning models.

Generally the output of such problems are going to be categorical variables that indicate discrete values. For example to determine whether an image is a cat image or not, the output for the test image is going to be yes or no.

There are few types of neural networks that are applied based upon the problem.

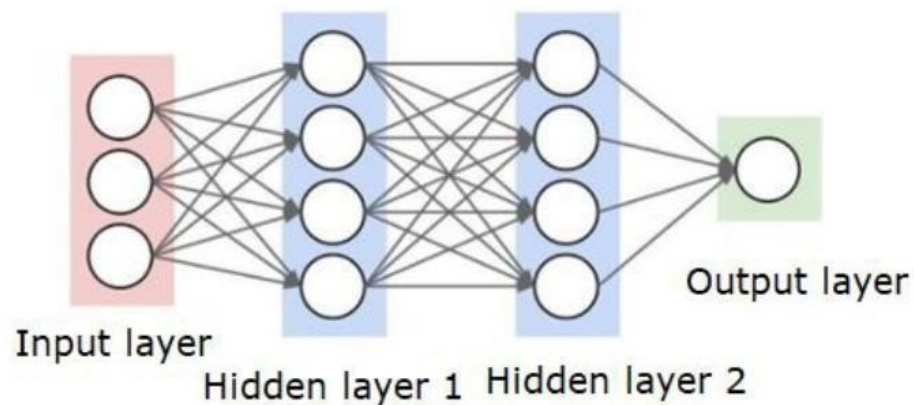
1. Feedforward Neural Network – Artificial Neuron

This is one of the simplest types of artificial neural networks. In a feedforward neural network, the data passes through the different input nodes till it reaches the output node.

In other words, data moves in only one direction from the first tier onwards until it reaches the output node. This is also known as a front propagated wave which is usually achieved by using a classifying activation function.

Unlike in more complex types of neural networks, there is no backpropagation and data moves in one direction only. A feedforward neural network may have a single layer or it may have hidden layers.

In a feed forward neural network, the sum of the products of the inputs and their weights are calculated. This is then fed to the output. Here is an example of a single layer feed forward neural network.



Feed forward Neural Network – Artificial Neuron

Feed forward neural networks are used in technologies like face recognition and computer vision. This is because the target classes in these applications are hard to classify.

A simple feed forward neural network is equipped to deal with data which contains a lot of noise. Feed forward neural networks are also relatively simple to maintain.

2. Convolutional Neural Network

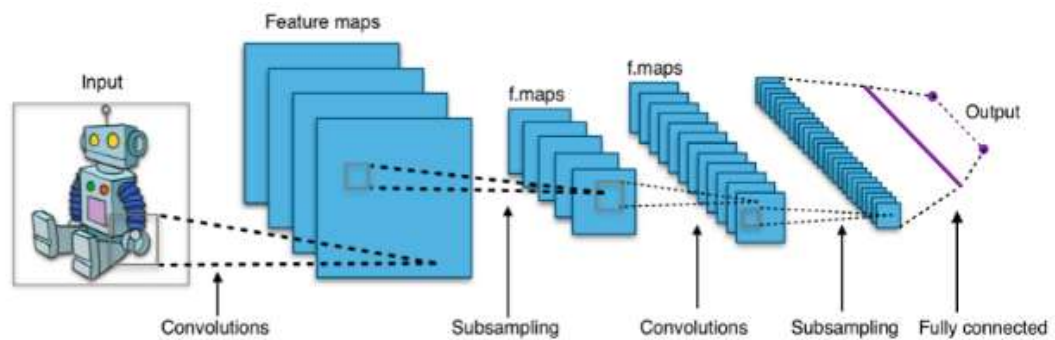
A convolutional neural network(CNN) uses a variation of the multilayer perceptrons. A CNN contains one or more than one convolutional layers. These layers can either be completely interconnected or pooled.

Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters.

Due to this ability, convolutional neural networks show very effective results in image and video recognition, natural language processing, and recommender systems.

Convolutional neural networks also show great results in semantic parsing and paraphrase detection. They are also applied in signal processing and image classification.

CNNs are also being used in image analysis and recognition in agriculture where weather features are extracted from satellites like LSAT to predict the growth and yield of a piece of land. Here's an image of what a Convolutional Neural Network looks like.



Convolutional Neural Network

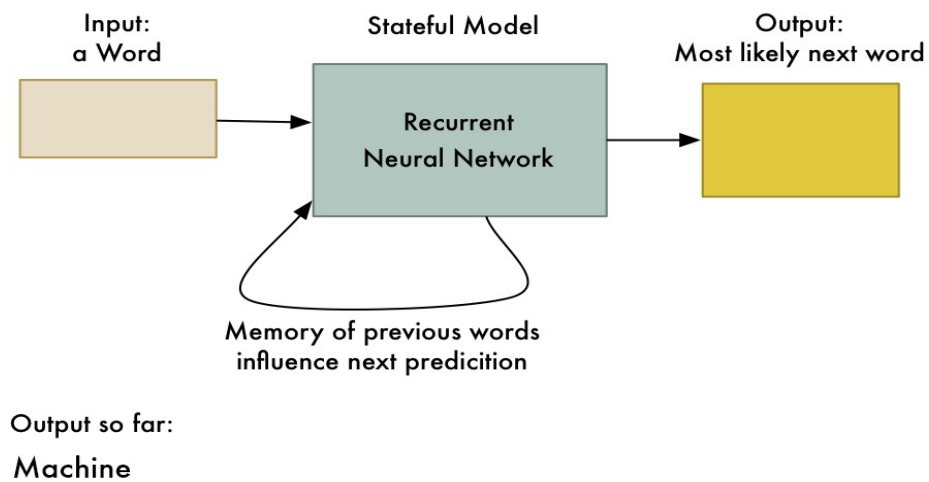
3. Recurrent Neural Network(RNN) – Long Short Term Memory

A Recurrent Neural Network is a type of artificial neural network in which the output of a particular layer is saved and fed back to the input. This helps predict the outcome of the layer.

The first layer is formed in the same way as it is in the feed forward network. That is, with the product of the sum of the weights and features. However, in subsequent layers, the recurrent neural network process begins.

From each time-step to the next, each node will remember some information that it had in the previous time-step. In other words, each node acts as a memory cell while computing and carrying out operations. The neural network begins with the front propagation as usual but remembers the information it may need to use later.

If the prediction is wrong, the system self-learns and works towards making the right prediction during the backpropagation. This type of neural network is very effective in text-to-speech conversion technology. Here's what a recurrent neural network looks like.



Recurrent Neural Network(RNN) – Long Short Term Memory

Deep Learning Process

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge or being explicitly coded by the programmers.

To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.

The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration.

Coming to our project,

The technique behind this project is Deep Metric Learning. Deep Metric Learning is an emerging field in metric learning by introducing deep neural network taking advantage of the nonlinear feature representation learning ability of deep learning and discrimination power of metric learning.

If you have any prior experience with deep learning you know that we typically train a network to:

Accept a single input image

And output a classification/label for that image

However, deep metric learning is different.

Instead, of trying to output a single label (or even the coordinates/bounding box of objects in an image), **we are instead outputting a real-valued feature vector.**

For the dlib facial recognition network, the output feature vector is **128-d** (i.e., a list of 128 real-valued numbers) that is used to *quantify the face*. The approach we are going to use for face recognition is fairly straight forward. The key here is to get a deep neural network to produce a bunch of numbers that describe a face (known as face encodings). When you pass in two different images of the same person, the network should return similar outputs (i.e. closer

numbers) for both images, whereas when you pass in images of two different people, the network should return very different outputs for the two images. This means that the neural network needs to be trained to automatically identify different features of faces and calculate numbers based on that. The output of the neural network can be thought of as an identifier for a particular person's face — if you pass in different images of the same person, the output of the neural network will be very similar/close, whereas if you pass in images of a different person, the output will be very different.

Thankfully, we don't have to go through the hassle of training or building our own neural network. We have access to a trained model through dlib that we can use. It does exactly what we need it to do — outputs a bunch of numbers (face encodings) when we pass in the image of someone's face; comparing face encodings of faces from different images will tell us if someone's face matches with anyone we have images of.

Here are the steps we will be taking

1. Detect/identify faces in an image (using a face detection model)
2. Predict face poses/landmarks (for the faces identified in step 1)
3. Using data from step 2 and the actual image, calculate face encodings (numbers that describe the face)
4. Compare the face encodings of known faces with those from test images to tell who is in the picture

Here is the heart of the algorithm:

3.3 Face_recognition module

The Face_recognition module identifies faces by using a famous technique called Histogram of Oriented Gradients. Let us see the procedure, first for a colour image we have to turn into greyscale image and later we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it.

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker.

Looking at just this one pixel and the pixels touching it, the image is getting darker towards the upper right. If you repeat that process for every single pixel in the image, you end up with every pixel being replaced by an arrow. These arrows are called gradients and they show the flow from light to dark across the entire image.



Fig 2.13 example greyscale image of a face

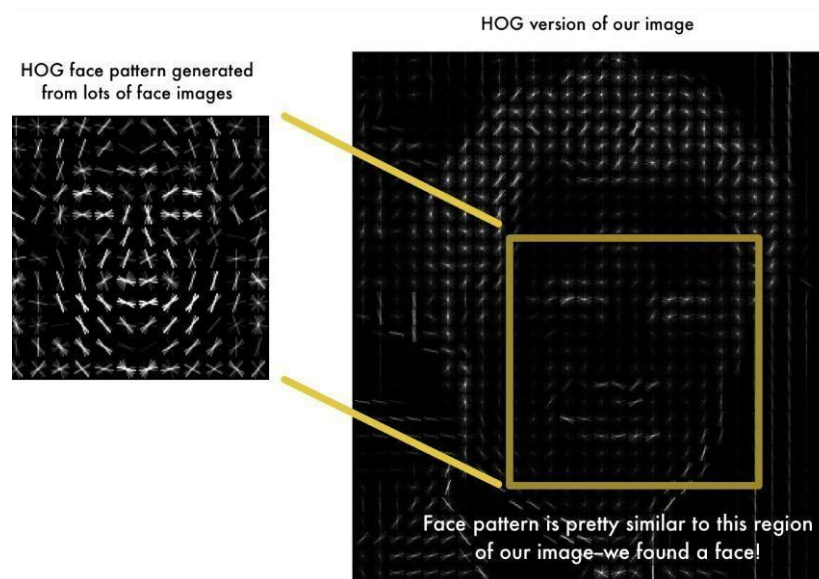


Fig 2.14 HOG version of the above pic.

When, we isolated the faces in our image. But now we have to deal with the problem that faces turned different directions look totally different to a computer.

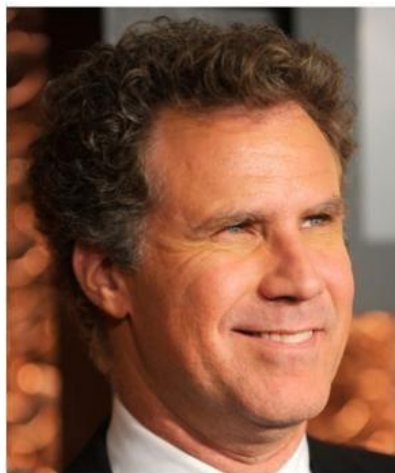


Fig 2.15 another example of the above person

Humans can easily recognize that both images are of Will Ferrell, but computers would see these pictures as two completely different people.

To account for this, we will try to wrap each picture so that the eyes and lips are always in the same place in the image. This will make it a lot easier for us to compare faces in the next steps.

To do this, we are going to use an algorithm called face landmark estimation. There are lots of ways to do this, but we are going to use the approach invented in 2014 by VahidKazemi and Josephine Sullivan.

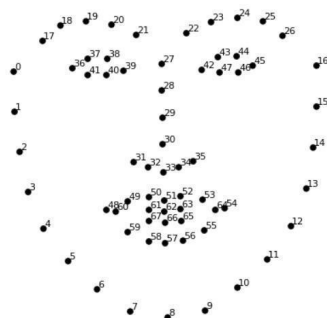


Fig 2.16 Landmarks on a face

The basic idea is we will come up with 68 specific points (called landmarks) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face. The 68 landmarks we will locate on every face.

Now that we know where the eyes and mouth are, we'll simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible. We won't do any fancy 3d wraps because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines (called affine transformations):

Now no matter how the face is turned, we are able to center the eyes and mouth are in roughly the same position in the image. This will make our next step a lot more accurate.

After that the model which is already trained by DavisKing which is used to generate a 128d vector value for any face is used to generate those 128 different values.

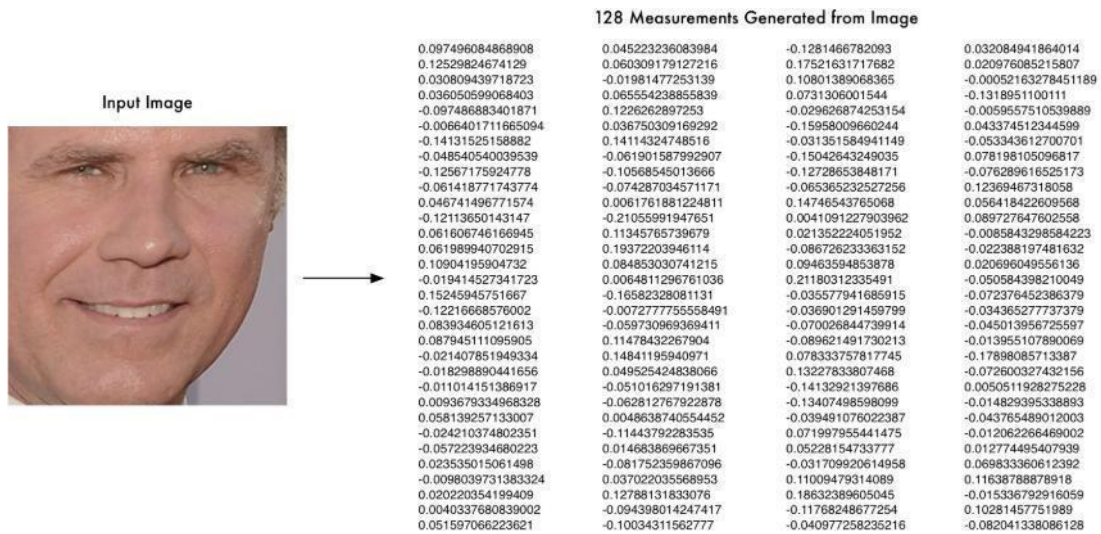


Fig 2.17 example of a face and 128 d vector value

Methods:

A.face encodings:

This method take an image as input and return a 128 values in an array.

B.face_locations:

This method takes an image as input and gives list of all face locations in that image.

C.compare_faces:

This method takes two inputs one an array of face encodings and second input is the image encodings to be compared and it returns the boolean array size of the given array in which if both are equal returns true else false.

4. SOFTWARE DESIGN

4.1. INTRODUCTION TO UML

The system to be developed is best designed using UML i.e Unified Modeling Language. The Unified Modeling Language includes a set of graphics notation techniques to create visual models of object oriented intensive systems. UML is a visual language for specifying, constructing, and documenting the artifacts of software- intensive systems. Complex software designs difficult for you to describe with text alone can readily be conveyed through diagrams using UML. You can use UML with all processes throughout the development lifecycle and across different implementation.

A model is a simplification of reality. We build models so that we can better understand the system we are developing. Through modeling, we achieve four aims. They are:

1. Models help us to visualize a system as it is or as we want it to be.
2. Models permit us to specify the structure or behavior of a system.
3. Models give us a template that guides us in constructing a system.
4. Models document the decisions we have made.

We build models of complex systems because we cannot comprehend such a System in its entirety.

PRINCIPLES OF MODELING

There are four basic principles common to designing any kind of system. They are:

1. The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
2. The right models will brilliantly illuminate the most wicked development problems,
4. Models document the decisions we have made.

We build models of complex systems because we cannot comprehend such a System in its entirety.

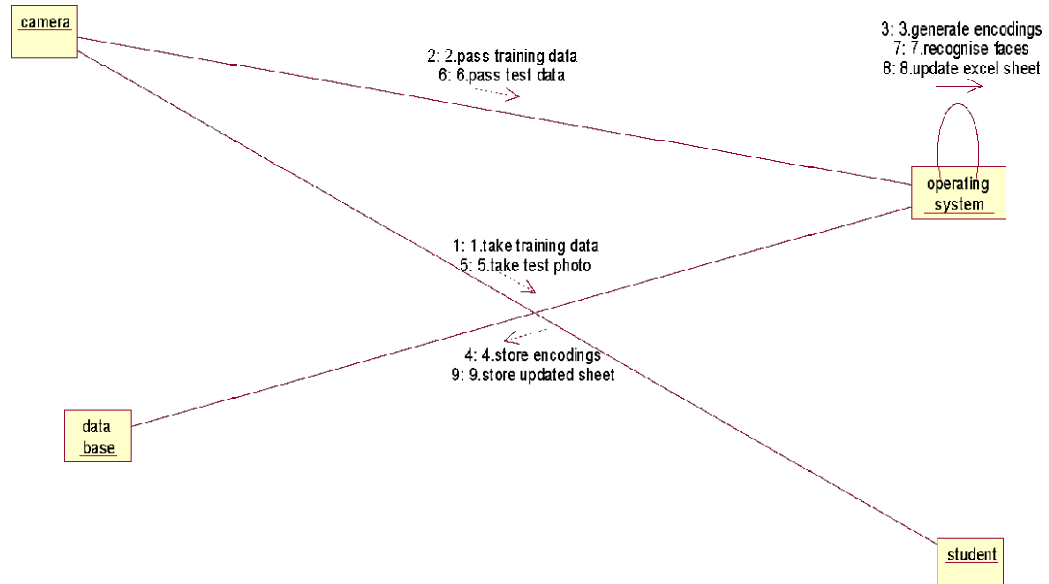
PRINCIPLES OF MODELING

There are four basic principles common to designing any kind of system. They are:

1. The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
2. The right models will brilliantly illuminate the most wicked development problems, offering insight that you simply could not gain otherwise; the wrong models will mislead you, causing you to focus on irrelevant issues.
3. Every model may be expressed at different levels of precision.
4. The best models are connected to reality.
5. No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models .To understand the architecture of a system, we need several complementary and interlocking views: a use case view (exposing the requirements of the system).

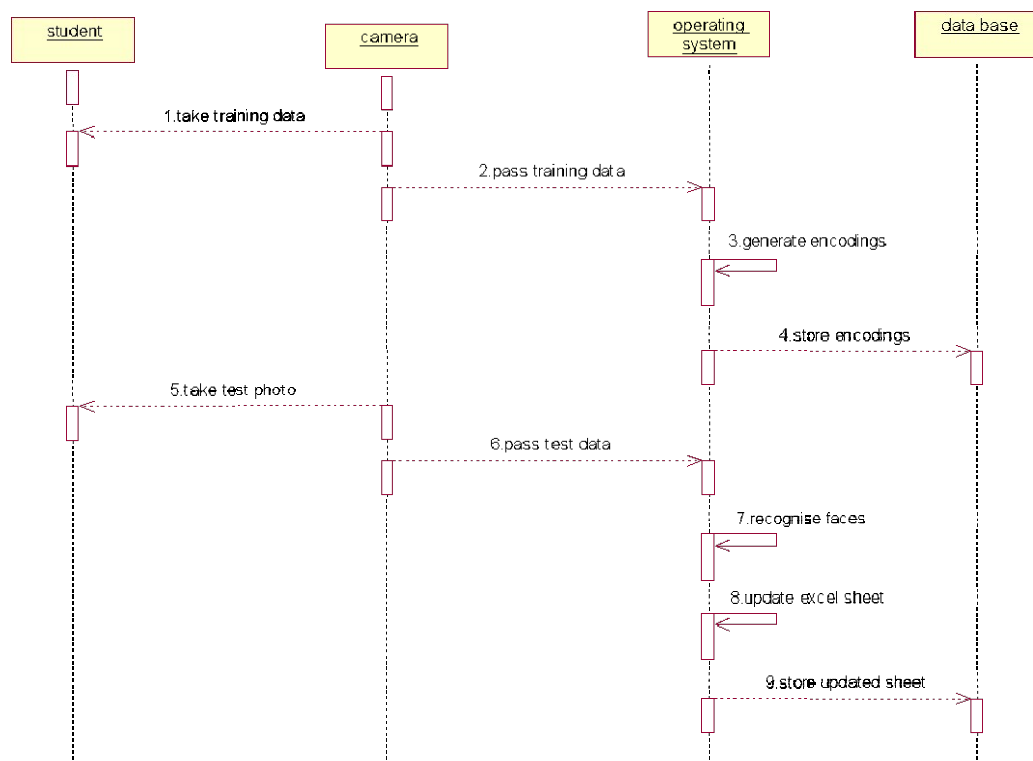
4.1.1 COLLABORATION DIAGRAM

These diagrams show the real time implementation of the project and the process that the system undergoes right from capturing images to marking the attendance. However, a clear description of the working is depicted in the Deep Learning Process section.



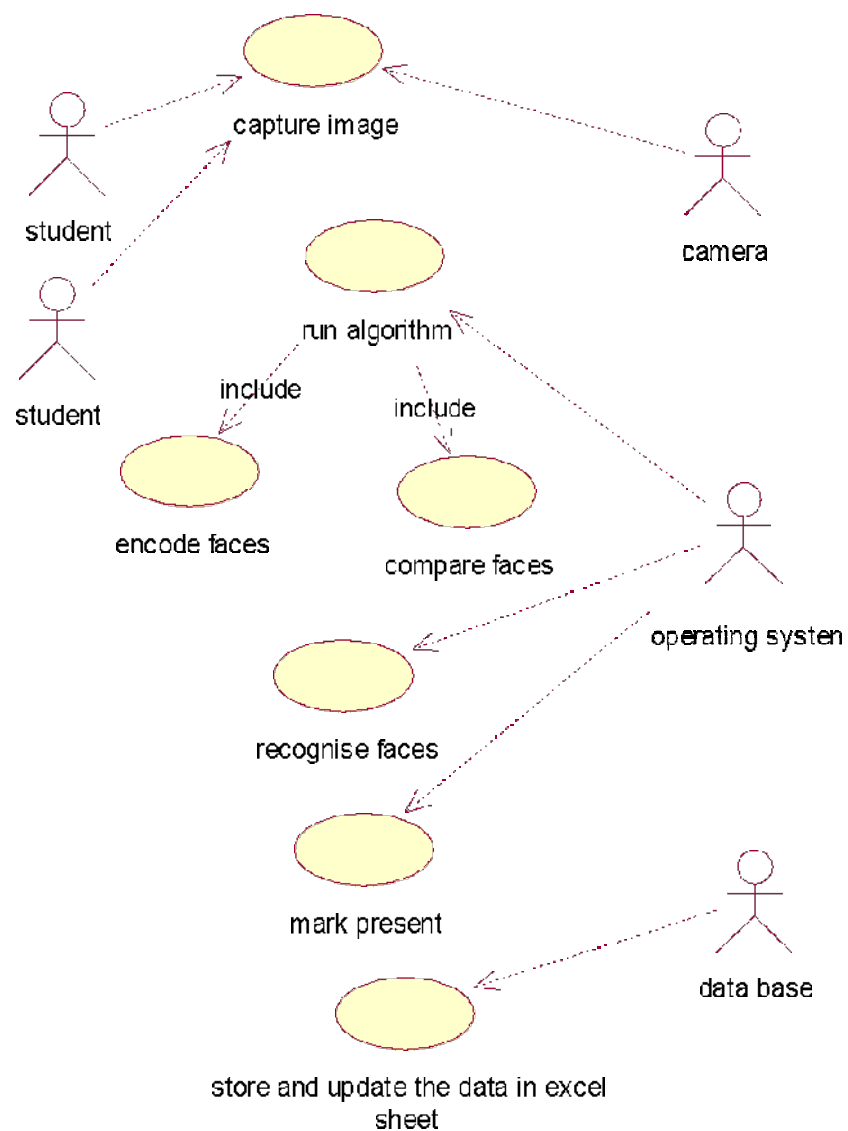
4.1.2 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.



4.1.3 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the users are involved



5. SOFTWARE REQUIREMENT SPECIFICATION

5.1 Functional Requirements

A functional requirement defines a function of system or its component .A function described as a set of inputs ,the behaviour and outputs.

5.1.1 Software Requirements

- Tool kit :Anaconda prompt
- language: python
- Operating System: Windows XP
- Packages: numpy, opencv, pandas, dlib, face_recognition

5.1.2 Hardware Requirements

- Primary Memory:8GB RAM
- Hard Disk:1TB
- Webcam

5.2 Non Functional Requirements

A non- functional requirement is a requirement that specify criteria that can be used to judge the operation of a system, rather than specific behaviours .Non functional requirements are called qualities of a system, there are as follows:

1. Performance and Processing times
2. Availability, Storage and Recovery
3. Query and reporting time
4. Capacity and scalability

6.CODING

6.1 CODE

capture.py:

This code enables the acquisition of training data. Photos of an individual person are captured using space button. Every time we click the space button a photo is clicked by the camera. This process can be ended by clicking Esc button.

```
#code starts
# -*- coding: utf-8 -*-
import cv2
cam = cv2.VideoCapture(0)
cv2.namedWindow("test")
img_counter = 1001
while True:
    ret, frame = cam.read()
    cv2.imshow("test", frame)
    if not ret:
        break
    k = cv2.waitKey(1)

    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name = "test691_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()
```

encodings_faces.py:

This python program is to train the algorithm with the data i.e., the photos that are collected using the above program and encodings are generated from the data which is used in the next program. All the encodings are stored in a pickle file

```
#code starts
# USAGE
# python encode_faces.py --dataset dataset --encodings encodings.pickle

# import the necessary packages
from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
                help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
                help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
                help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# grab the paths to the input images in our dataset
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []
```

```

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # load the input image and convert it from RGB (OpenCV ordering)
    # to dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # detect the (x, y)-coordinates of the bounding boxes
    # corresponding to each face in the input image
    boxes = face_recognition.face_locations(rgb,
        model=args["detection_method"])

    # compute the facial embedding for the face
    encodings = face_recognition.face_encodings(rgb, boxes)

    # loop over the encodings
    for encoding in encodings:
        # add each encoding + name to our set of known names and
        # encodings
        knownEncodings.append(encoding)
        knownNames.append(name)

# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open(args["encodings"], "wb")
f.write(pickle.dumps(data))
f.close()

```

recognize.py:

This is the main algorithm. It takes the encodings from the pickle file in the arguments specified and generates encodings for the test image that is taken as input by this program. The stored encodings are compared with the encodings generated by this program. The detected faces in the test images are assigned with the label that best matches its encodings among the training data.

```
#code starts
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 24 20:48:25 2019

@author: Rohitsai
"""

# USAGE
# python recognize_faces_image.py --encodings encodings.pickle --image
# examples/example_01.png

# import the necessary packages
import face_recognition
import argparse
import pickle
import cv2
import numpy as np
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile
from datetime import datetime

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
                help="path to serialized db of facial encodings")
```

```

ap.add_argument("-i", "--image", required=True,
                help="path to input image")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
                help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# load the known faces and embeddings
print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())

# load the input image and convert it from BGR to RGB
image = cv2.imread(args["image"])
image = np.array(image, dtype=np.uint8)
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# detect the (x, y)-coordinates of the bounding boxes corresponding
# to each face in the input image, then compute the facial embeddings
# for each face
print("[INFO] recognizing faces...")
boxes = face_recognition.face_locations(rgb,
    model=args["detection_method"])
encodings = face_recognition.face_encodings(rgb, boxes)

# initialize the list of names for each face detected
names = []

# loop over the facial embeddings
for encoding in encodings:
    # attempt to match each face in the input image to our known
    # encodings
    matches = face_recognition.compare_faces(data["encodings"],
        encoding)
    name = "Unknown"

    # check to see if we have found a match
    if True in matches:
        # find the indexes of all matched faces then initialize a

```



```

# dictionary to count the total number of times each face
# was matched
matchedIdxs = [i for (i, b) in enumerate(matches) if b]
counts = {}

# loop over the matched indexes and maintain a count for
# each recognized face face
for i in matchedIdxs:
    name = data["names"][i]
    counts[name] = counts.get(name, 0) + 1

# determine the recognized face with the largest number of
# votes (note: in the event of an unlikely tie Python will
# select first entry in the dictionary)
name = max(counts, key=counts.get)

# update the list of names
names.append(name)

# loop over the recognized faces
for ((top, right, bottom, left), name) in zip(boxes, names):
    # draw the predicted face name on the image
    cv2.rectangle(image, (left, top), (right, bottom), (0, 255, 0), 2)
    y = top - 15 if top - 15 > 15 else top + 15
    cv2.putText(image, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
                0.75, (0, 255, 0), 2)

l1=['m8','m9','n0','n1','n2','n3','n4','o1','o2','o3','o4']
l2=[]
for i in l1:
    if i in names:
        l2.append('p')
    else:
        l2.append('n')

```

```

df = pd.DataFrame({'roll numbers':['m8','m9','n0','n1','n2','n3','n4','o1','o2','o3','o4'],
                  '1st day':[l2[0],l2[1],l2[2],l2[3],l2[4],l2[5],l2[6],l2[7],l2[8],l2[9],l2[10]],
                  })

df = pd.read_excel("Pandas-Example2.xlsx")
print(df.head())
#df['d']=[1,2,3,4,5,6,7,8,9,10,11]
#df2 = pd.DataFrame([5, 6], columns='a')
#df.append(df2,ignore_index=False)
#df.append({'e':[1,2,3,4,5,6,7,8,9,10,11]})
i=df.shape[1]
atr1=str(datetime.now())
df.insert(i, atr1,l2, allow_duplicates = False)

writer = ExcelWriter('Pandas-Example2.xlsx')
df.to_excel(writer,'Sheet1',index=False)
writer.save()

# show the output image
cv2.imshow("Image", image)
cv2.waitKey(0)

```

7.TESTING AND OUTPUT SCREENS

```
Anaconda Prompt - python encode_faces.py --dataset trainsdata --encodings 169.pickle

(base) C:\Users\Rohit sai>activate test1

(test1) C:\Users\Rohit sai>cd C:\Users\Rohit sai\.spyder-py3

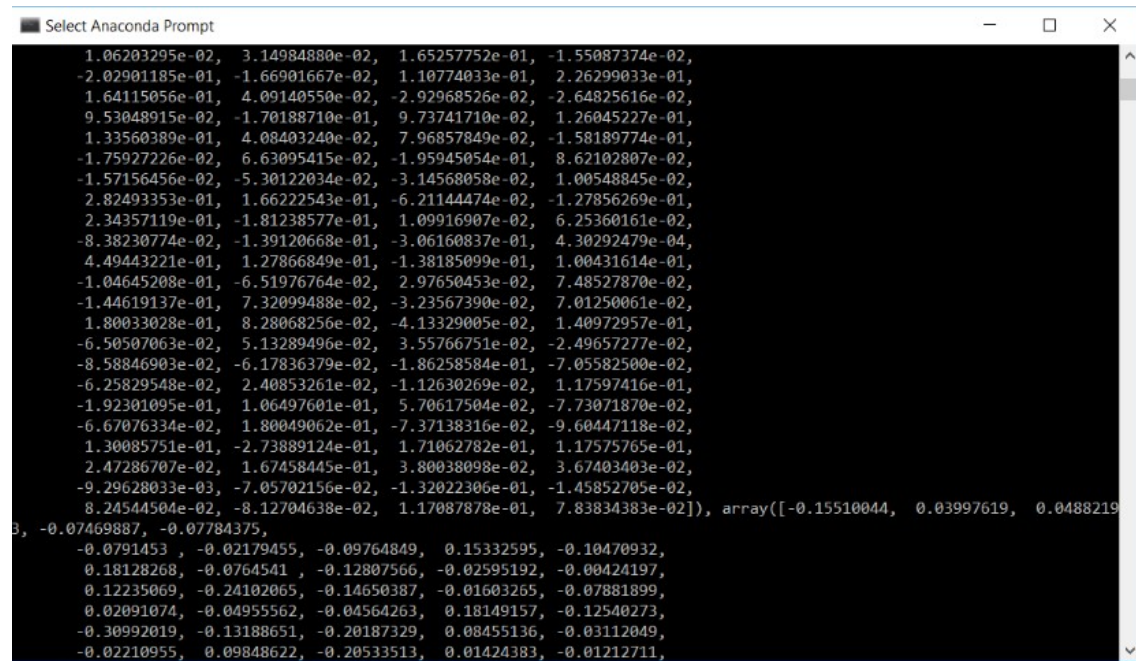
(test1) C:\Users\Rohit sai\.spyder-py3>python encode_faces.py --dataset trainsdata --encodings 169.pickle
[INFO] quantifying faces...
[INFO] processing image 1/83
[INFO] processing image 2/83
[INFO] processing image 3/83
[INFO] processing image 4/83
[INFO] processing image 5/83
[INFO] processing image 6/83
[INFO] processing image 7/83
[INFO] processing image 8/83
[INFO] processing image 9/83
[INFO] processing image 10/83
```

Output for encodings_faces.py:

```
Anaconda Prompt - python recognize.py --encodings 169.pickle --image test69/test.png

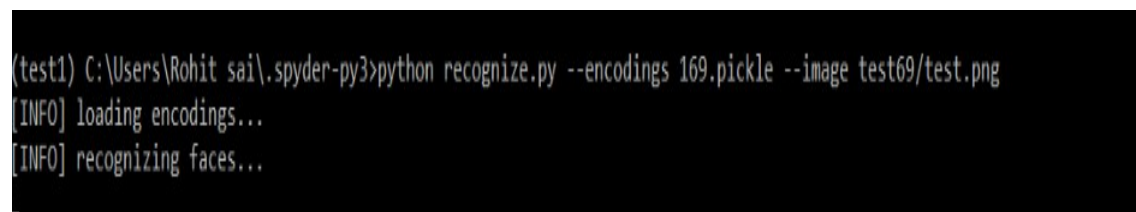
[INFO] processing image 60/83
[INFO] processing image 61/83
[INFO] processing image 62/83
[INFO] processing image 63/83
[INFO] processing image 64/83
[INFO] processing image 65/83
[INFO] processing image 66/83
[INFO] processing image 67/83
[INFO] processing image 68/83
[INFO] processing image 69/83
[INFO] processing image 70/83
[INFO] processing image 71/83
[INFO] processing image 72/83
[INFO] processing image 73/83
[INFO] processing image 74/83
[INFO] processing image 75/83
[INFO] processing image 76/83
[INFO] processing image 77/83
[INFO] processing image 78/83
[INFO] processing image 79/83
[INFO] processing image 80/83
[INFO] processing image 81/83
[INFO] processing image 82/83
[INFO] processing image 83/83
[INFO] serializing encodings...
```

This is how encodings of training data look like:



```
Select Anaconda Prompt
1.06203295e-02, 3.14984880e-02, 1.65257752e-01, -1.55087374e-02,
-2.02901185e-01, -1.66901667e-02, 1.10774033e-01, 2.26299033e-01,
1.64115056e-01, 4.09140550e-02, -2.92968526e-02, -2.64825616e-02,
9.53048915e-02, -1.70188710e-01, 9.73741710e-02, 1.26045227e-01,
1.33560389e-01, 4.08403240e-02, 7.96857849e-02, -1.58189774e-01,
-1.75927226e-02, 6.63095415e-02, -1.95945054e-01, 8.62102807e-02,
-1.57156456e-02, -5.30122034e-02, -3.14568058e-02, 1.00548845e-02,
2.82493353e-01, 1.66222543e-01, -6.21144474e-02, -1.27856269e-01,
2.34357119e-01, -1.81238577e-01, 1.09916907e-02, 6.25360161e-02,
-8.38230774e-02, -1.39120668e-01, -3.06160837e-01, 4.30292479e-04,
4.49443221e-01, 1.27866849e-01, -1.38185099e-01, 1.00431614e-01,
-1.04645208e-01, -6.51976764e-02, 2.97650453e-02, 7.48527870e-02,
-1.44619137e-01, 7.32099488e-02, -3.23567390e-02, 7.01250061e-02,
1.80033028e-01, 8.28068256e-02, -4.13329005e-02, 1.40972957e-01,
-6.50507063e-02, 5.13289496e-02, 3.55766751e-02, -2.49657277e-02,
-8.58846903e-02, -6.17836379e-02, -1.86258584e-01, -7.05582500e-02,
-6.25829548e-02, 2.40853261e-02, -1.12630269e-02, 1.17597416e-01,
-1.92301095e-01, 1.06497601e-01, 5.70617504e-02, -7.73071870e-02,
-6.67076334e-02, 1.80049062e-01, -7.37138316e-02, -9.60447118e-02,
1.30085751e-01, -2.73889124e-01, 1.71062782e-01, 1.17575765e-01,
2.47286707e-02, 1.67458445e-01, 3.80038098e-02, 3.67403403e-02,
-9.29628033e-03, -7.05702156e-02, -1.32022306e-01, -1.45852705e-02,
8.24544504e-02, -8.12704638e-02, 1.17087878e-01, 7.83834383e-02], array([-0.15510044, 0.03997619, 0.0488219
3, -0.07469887, -0.07784375,
-0.0791453, -0.02179455, -0.09764849, 0.15332595, -0.10470932,
0.18128268, -0.0764541, -0.12807566, -0.02595192, -0.00424197,
0.12235069, -0.24102065, -0.14650387, -0.01603265, -0.07881899,
0.02091074, -0.04955562, -0.04564263, 0.18149157, -0.12540273,
-0.30992019, -0.13188651, -0.20187329, 0.08455136, -0.03112049,
-0.02210955, 0.09848622, -0.20533513, 0.01424383, -0.01212711,
```

Output for recognize.py:



```
(test1) C:\Users\Rohit sai\.spyder-py3>python recognize.py --encodings 169.pickle --image test69/test.png
[INFO] loading encodings...
[INFO] recognizing faces...
```

Pandas-Example2.xlsx - Excel (Unlicensed)

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

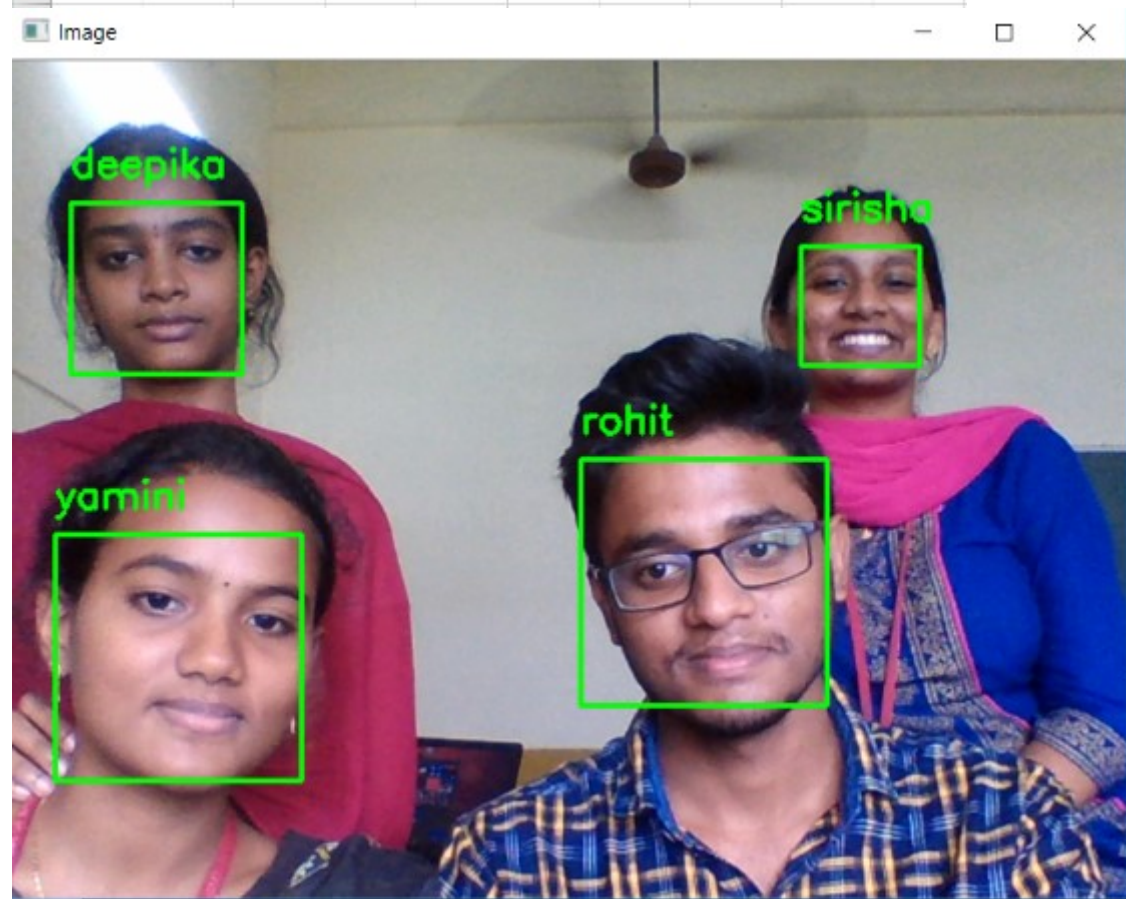
Clipboard Font Alignment Number

SUBSCRIPTION EXPIRED Most of the features of Excel have been disabled. [Reactivate](#)

ADDITIONAL INFORMATION If you need to make a quick edit before renewing your subscription, learn more about the subscription.

J6

	A	B	C	D	E	F	G	H	I	J
1	roll number	1st day	06 16:57:10	06 16:59:01	16 10:43:41	16 10:45:50	16 10:50:11	16 11:01:05.553	414	
2	m8	n	n	n	n	n	n	n		
3	m9	n	n	n	n	n	n	n		
4	n0	n	n	n	n	n	n	p		
5	n1	n	n	n	n	n	n	n		
6	n2	n	n	n	n	n	n	p		
7	n3	n	n	n	n	n	n	n		
8	n4	n	n	n	n	n	n	p		
9	o1	n	n	n	n	n	n	n		
10	o2	n	n	n	n	n	n	n		
11	o3	n	n	n	n	n	n	p		
12	o4	n	n	n	n	n	n	n		
13										
14										



8.CONCLUSION

An automatic attendance management system is a necessary tool for any kind of schools and colleges. Most of the existing systems are time consuming and require a semi manual work from the teacher or students. Our approach aims to solve the issues by integrating face recognition in the process. Even though this system has disadvantages like effectively detecting large number of faces and lack of more scope of automation, there is much more room for improvement. Since we implement a modular approach we can improve different modules until we reach an acceptable detection and identification rate. The system can be enhanced in such a way that the accuracy, detection rate and recognition rate can be increased so that more number of students can be detected and recognized for those who are present in the class. The deep learning algorithm implemented has shown an accuracy of about 98% in recognizing the face of a single person and is shows more efficient accuracy in detecting more number of faces with better accuracy. Of the whole project, encoding faces is the tricky and time consuming part and this decides the probability if a person being recognized correctly. A better approach from our side to make the algorithm run efficiently is to give the training data in a better way that is to capture more number of images that can extract each and every detail of the face clearly and in different conditions like photos of a person with spectacles and without them, with beard and without beard and with different expressions and angles of face. In this way, we can make a step forward in automating the attendance system and make our schools and colleges more smart.

9. FUTURE WORK

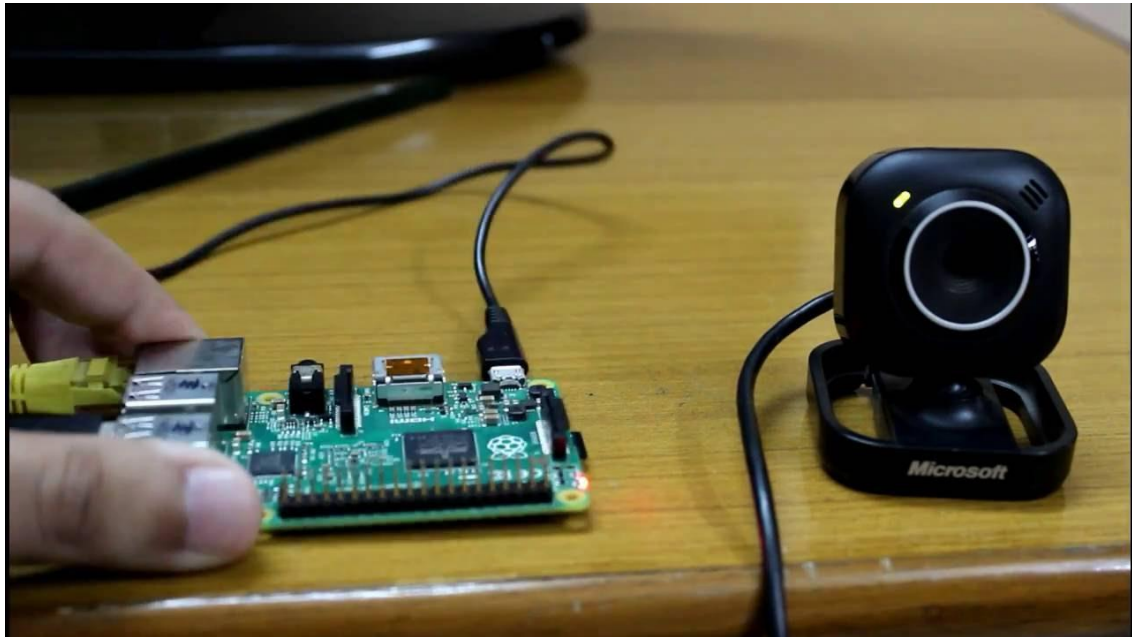
The system we have developed has successfully, able to accomplish the task of marking the attendance in the classroom automatically and output is obtained in an excel sheet as desired in real-time. However, in order to develop a dedicated system which can be implemented in an educational institution, a very efficient algorithm which is insensitive to the lighting conditions of the classroom has to be developed. Also a camera of the optimum resolution has to be utilized in the system. Another important aspect where we can work towards is creating an online database of the attendance and automatic updating of the attendance into it keeping in mind the growing popularity of Internet of Things. This can be done by creating a standalone module which can be installed in the classroom having access to internet, preferably a wireless system. These developments can greatly improve the applications of the project.

The approach explained in this thesis is a powerful algorithm that is used for face identification. It can be taken further and extended into a project that involves Internet of Things. The idea of the proposed future work is that there is a web camera connected to the raspberry pi camera module that is set up above the blackboard on the wall. The raspberry pi itself being an operating system with an integrated python development environment, our algorithm can be made to run in the device itself and the data that will be obtained as output and this data can be updated in the cloud so that the data can be accessible remotely by a person who is authorized to use that cloud.

The choice of cloud depends on the individual or the organization. Preferred clouds could be

Google cloud or amazon web services.

A sample image of how the connection of a web camera with a raspberry pi:



10. REFERENCES

<https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/>

https://github.com/ageitgey/face_recognition

<https://www.learnopencv.com/install-dlib-on-windows/>

<https://www.edureka.co/blog/convolutional-neural-network/>