PROJECT REPORT

ON

Efficient Real-Time Face Recognition-Based Attendance System with Deep Learning Algorithms

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Yukti Bisht (University Roll No: 2018889)

Kritagya Painuly (University Roll No: 2018436)

Under the Guidance of

Ms. Himadri Vaidya

Department of Computer Science and Engineering

Graphic Era Hill University

Dehradun, Uttarakhand

Graphic Era
HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956

CERTIFICATE

This is to certify that the thesis titled **"**Efficient Real-Time Face Recognition-Based Attendance System with Deep Learning Algorithms**" submitted** by **Kritagya Painuly and Yukti Bisht**, to Graphic Era Hill University for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him/her under our supervision. The contents of this project in full or in parts have not been submitted to any other Institute or University for the award of any degree or diploma.

Himadri Vaidya
GEHU, Dehradun
Place: Dehradun
Date: 15/05/2024

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on individual effort but on the guidance, encouragement, and cooperation of intellectuals, elders, and friends. Several personalities in their capacity have helped me in carrying out this project work.

Our sincere thanks to project guide **Ms. Himadri Vaidya** Department of Computer Science and Engineering, Graphic Era Hill University, for her valuable guidance and support throughout the course of project work and for being a constant source of inspiration.

We thank the management of Graphic Era Hill University for the support throughout the course of our bachelor's degree and for all the facilities they have provided.

Last, but certainly not least, we thank all teaching and non-teaching staff of Graphic Era Hill University for guiding us on the right path. Most importantly we wish to thank our parents for their support and encouragement.

Kritagya Painuly 2018436

Yukti Bisht 2018889

ABSTRACT

Efficient Real-Time Face recognition-based attendance systems with deep learning algorithms strive for precise and real-time performance in managing attendance across various institutions. This solution outlines key targets as well as robust face detection, model selection, real-time processing, information collection, and ensuring accuracy and reliability in face recognition by combining dlib, cvzone, and YOLO algorithms for face detection, and blur detection to improve the system's efficiency. In a nutshell, the suggested system offers a promising approach to attendance control in challenging situations by using advanced methods to signify high accuracy, real-time processing, and value.

Index Terms—CNNs, YOLO, Computer Vision, Deep learning.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

In an era defined by technological advancements, the intersection of deep learning algorithms and real-world applications continues to unlock new opportunities and redefine how we interact with technology. The study by John et al. [1] demonstrates the transformative potential of these technologies. This project embarks on a journey to explore the realm of "Efficient Real-Time Face Recognition-Based Attendance Systems with Deep Learning Algorithms." It seeks to leverage the power of deep learning to create an innovative solution for modern attendance management systems.

Face recognition technology has come a long way, and the utilization of deep learning algorithms has further refined its accuracy and efficiency. This project aims to dive deep into the nuances of facial recognition and attendance tracking while embracing the state-of-the-art in deep learning techniques. The research builds upon the existing knowledge in this area, as evidenced in John et al.'s work [1], and strives to address the evolving challenges while uncovering novel insights.

The main objective of this project is to explore how deep learning algorithms can be harnessed to create an efficient real-time face recognition-based attendance system. This system is envisioned to cater to a wide range of applications, from educational institutions to corporate settings. By facilitating efficient and accurate attendance management, this technology has the potential to streamline operations and enhance security.

1.2 Problem Statement

The core problem statement for this work can be succinctly stated as follows:

"To develop an efficient real-time face recognition-based attendance system using deep learning algorithms."

This project is poised to address several significant challenges in the domain of facial recognition and attendance tracking. These challenges include achieving high levels of accuracy in real-time face recognition, optimizing the system for efficient performance, and seamlessly integrating it into various educational and organizational contexts.

The project is driven by a set of primary objectives:

1. **Implementing Real-Time Face Recognition:** Creating a real-time face recognition system that can accurately identify individuals in diverse settings.
2. **Developing an Attendance Tracking Mechanism:** Designing and implementing an attendance tracking mechanism that is tightly integrated with the face recognition system, ensuring accurate and efficient tracking.
3. **Optimizing System Efficiency:** Optimizing the system for efficiency and minimal resource utilization, making it practical for a wide range of applications.
4. **User-Friendly Interface**: Providing a user-friendly interface that simplifies the deployment and utilization of the system.

The successful accomplishment of these objectives will result in an innovative solution that transcends the boundaries of attendance management. This system has the potential to reshape security measures, streamline access control, and revolutionize identity verification processes in diverse sectors.

As the subsequent chapters unfold, this research will delve into the background, research objectives, methodologies, and practical implementations of the proposed face recognition-based attendance system. The goal is to contribute significantly to the field of facial recognition and provide an efficient tool for attendance management that can benefit educational institutions and organizations alike. Figure shows the flow Diagram.

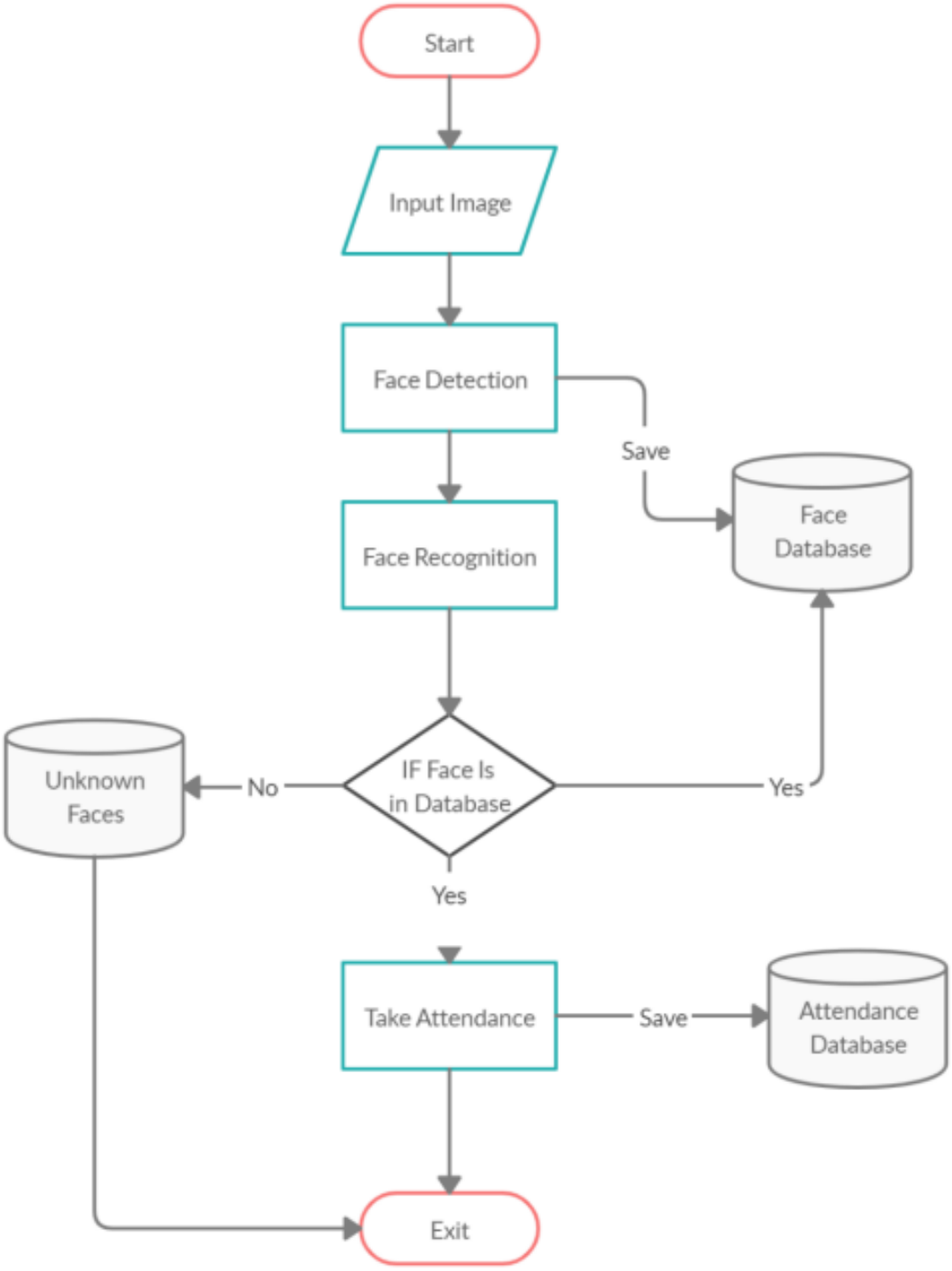


Fig: Flow Diagram

Chapter 2

Literature Survey

2.1 Introduction

In today's rapidly evolving technological landscape, research is constantly pushing the boundaries of innovation, particularly in the domain of facial recognition, deep learning, and real-time systems. This chapter provides a comprehensive overview of the existing body of work related to these areas and sets the stage for the current research.

2.2 Facial Recognition Technology

Facial recognition technology has improved dramatically in recent years and has established its place as one of the most important innovations in computer vision. Recent literature has shown improvements in neural networks and facial analysis algorithms that have greatly improved recognition accuracy. Notably, three fundamental approaches have traditionally dominated the field: Eigenfaces, Fisherfaces, and Local Binary Patterns (LBP). Eigenfaces use principal component analysis (PCA) for feature extraction [1]. In contrast, Fisherfaces use linear discriminant analysis (LDA) for improved face recognition ability [2] LBP on the other hand, focuses on texture characteristics of face images for analysis [3]. Figure shows the three different techniques for facial recognition.

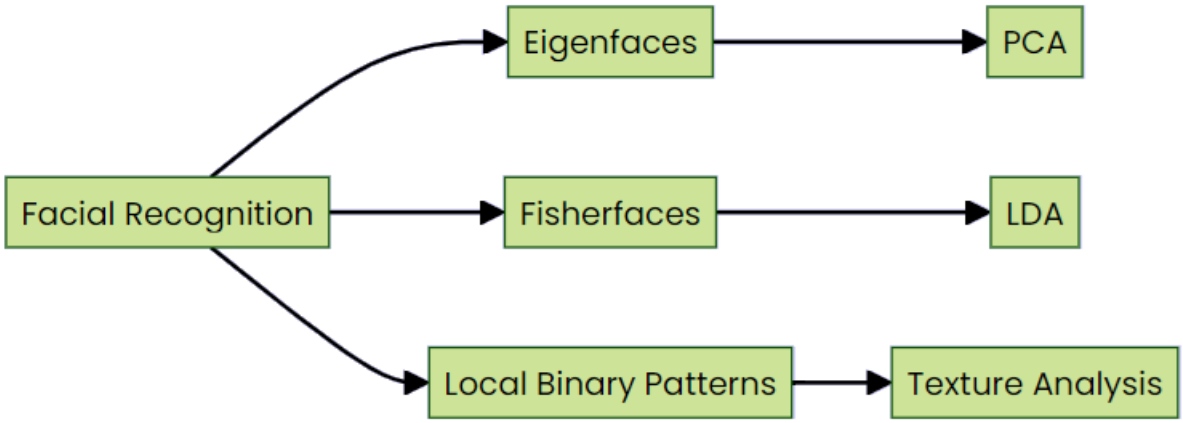


Fig Facial Recognition Approaches.

2.3 Deep Learning Algorithms

Despite these successes, there are still challenges in applying this technology to real-world situations, especially with changes in lighting, settings, and facial expressions Modern research aims to address these issues role by integrating deep learning to improve these classic techniques. Artificial intelligence has been transformed by the rise of deep learning algorithms, especially convolutional neural networks (CNNs) which have outstanding concepts like ImageNet that stimulate innovation. and in the Deep Face Recognition (DFR) subfield, deep neural networks are used to extract discriminatory features from face images, greatly improving recognition rates FaceNet, DeepFace, OpenFace and other architectures became prominent, showing how DFR is effectively used in face recognition tasks. Figure 2 shows the two different algorithms for deep learning algorithms.

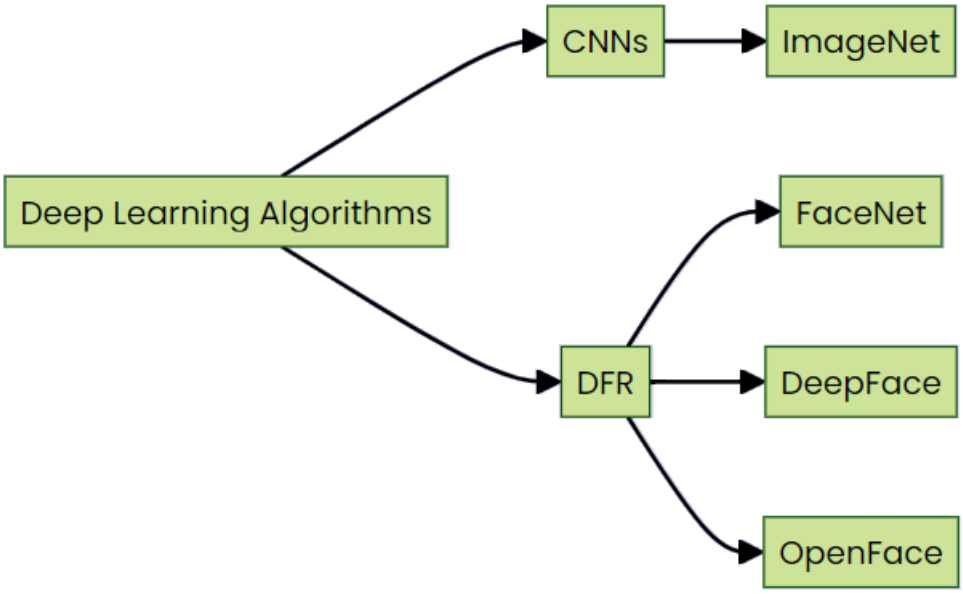


Fig: Deep Learning Algorithms

With advances in traditional techniques such as Eigenfaces, Fisherfaces, and LBP have laid the foundation, Deep Face Recognition (DFR) has taken center stage DFR uses deep neural networks to extract hidden features from face images very accurately and robustly. Products like FaceNet, DeepFace, and OpenFace have become industry standards, pushing the boundaries of real-time visual capabilities. Some recent developments in DFR are the attention-based methods. These models focus on specific facial regions, such as the eyes and mouth, to improve the intensity of lighting and the accuracy of state transitions [4]. The other method is knowledge embedding Small, agile machines are trained by" learning" from pre-trained larger examples, which allows them to be used more efficiently on machines with significant resources [5]. Cross-modal learning is a method which combining facial similarity with other data such as walking, voice, or thermal image improves recognition accuracy and spoof detection [6]. Real-time performance is critical for effective outreach planning. Recent research has focused lightweight architectures. By modifying DFR models for faster computation on edge devices such as smartphones or embedded systems. [7] The resource-aware resourcing

techniques such as dynamic resource allocation and pooling will be used to efficiently manage peak workloads [8] and edge computing decentralizing processing power to the point of data acquisition (e.g., camera) provides faster response times and improved privacy [9]. Despite the progress, some challenges remain some of them are discussed further. Privacy issues facial recognition systems require hard data creation and ethical considerations to balance security and user privacy. Bias and fairness which is used to ensure fair and unbiased algorithms accurately represent diverse populations. Adaptability to various environments helps to manages changes in lighting, camera angle and environment. In recent studies, various approaches have been explored in the field of face recognition. Masi et al. raised the question of whether collecting millions of faces is necessary for effective face recognition [10]. Arsenovic et al. developed FaceTime, a deep learning-based face recognition attendance system [11]. Prayogo et al. introduced a novel approach using YOLO-based face detection and FaceNet for face recognition [12]. Khalili and Shakiba proposed a face detection method via an ensemble of four versions of YOLOs [13]. Bittal et al. implemented a facial recognition system for automatic attendance tracking using an ensemble of deep-learning techniques [14]. Bittal et al. presented a multifarious face attendance system using machine learning and deep learning [15]. Suman Menon et al. implemented a multitudinous face recognition system using YOLOv3 [16].

2.4 Student Attendance System

Arun Katara et al. (2017) mentioned disadvantages of RFID (Radio Frequency Identification) card system, fingerprint system and iris recognition system. RFID card system is implemented due to its simplicity. However, the user tends to help their friends tocheck in as long as they have their friend’s ID card. The fingerprint system is indeed effective but not efficient becauseit takes time for the verification process so the user has to line up and perform the verification one by one. However

for face recognition, the human face is always exposed and contain less information compared to iris. Iris recognition system whichcontains more detail might invade the privacy of the user. Voicerecognition is available, but it is less accurate compared to other methods. Hence, face recognition system is suggested to beimplemented in the student attendance system.

System Type	Advantage	Disadvantages
RFID card system	Simple	Fraudulent usage
Fingerprint system	Accurate	Time-consuming
Voice recognition system		Less accurate compared toOthers
Iris recognition system	Accurate	Privacy Invasion

Table 2.4: Advantages & Disadvantages of Different BiometricSystem[1]

2.5 Digital Image Processing

Digital Image Processing is the processing of images which are digital in nature by a digital computer[2]. Digital image processing techniques are motivated by three major applicationsmainly:

- Improvement of pictorial information for human perception
- Image processing for autonomous machine application

- Efficient storage and transmission.

2.5 Image Representation in a DigitalComputer

An image is a 2-Dimensional light intensity function

$$\mathbf{f}(\mathbf{x},\mathbf{y}) = \mathbf{r}(\mathbf{x},\mathbf{y}) \times \mathbf{i}(\mathbf{x},\mathbf{y}) - \quad (2.0)$$

Where, r (x, y) is the reflectivity of the surface of the corresponding image point. i (x,y) Represents the intensity of the incident light. A digital image f(x, y) is discretized both in spatial co-ordinates by grids and in brightness by quantization[3]. Effectively, the image can be represented as a matrix whose row, column indices specify a point in the image and the element value identifies gray level value at that point.These elements are referred to as pixels or pels.

Typically following image processing applications, the image size which is used is$\mathbf{256 \times 256}$, elements, $\mathbf{640 \times 480}$ pels or $\mathbf{1024 \times 1024}$ pixels. Quantization of these matrix pixels is done at 8 bits for black and white images and 24 bits for colored images (because of the three color planes Red, Green and Blue each at 8 bits)[4].

2.6 Steps in Digital Image Processing

Digital image processing involves the following basic tasks:

Image Acquisition - An imaging sensor and the capability todigitize the signal produced by the sensor.

Preprocessing – Enhances the image quality, filtering,contrast enhancement etc.

Segmentation – Partitions an input image into constituentparts of objects.

Description/feature Selection – extracts the description ofimage objects suitable for further computer processing.

Recognition and Interpretation – Assigning a label to the object based on the information provided by its descriptor.

Interpretation assigns meaning to a set of labelledobjects.

Knowledge Base – This helps for efficient processing aswell as inter module cooperation.

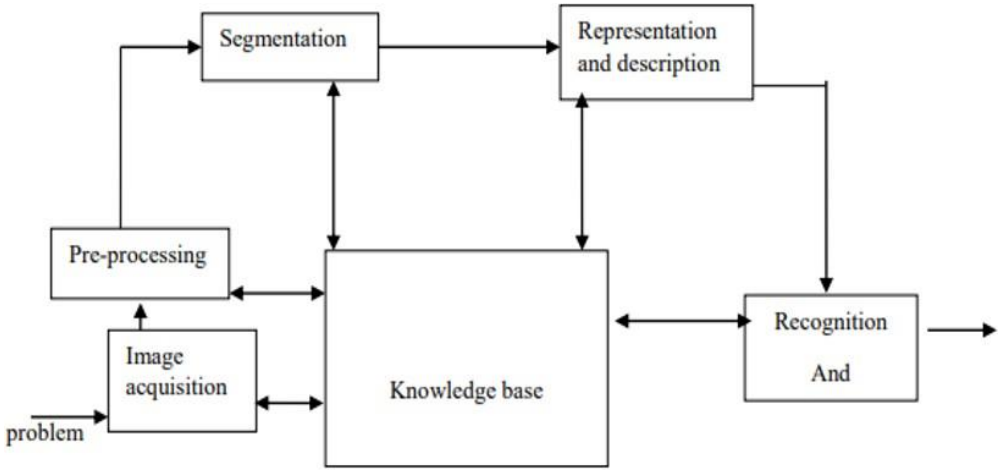


Fig: A diagram showing the steps in digital image processing

Definition of Terms and HistoryFace Detection

Face detection is the process of identifying and locating all the present faces in a single image or video regardless of their position, scale, orientation, age and expression. Furthermore, the detection should be irrespective of extraneous illumination conditions and the image and video content[5].

Face Recognition

Face Recognition is a visual pattern recognition problem, where the face, represented as a three dimensional object that is subject to varying illumination, pose and other factors, needs to be identified based on acquired images[6].

Face Recognition is therefore simply the task of identifying analready detected face as a known or unknown face and in more advanced cases telling exactly whose face it is[7

Difference between Face Detection and Face Recognition

Face detection answers the question, Where is the face? It identifies an object as a "face" and locates it in the input image. Face Recognition on the other hand answers the question who is this? Or whose face is it? It decides if the detected face is someone known or unknown based on the database of facesit uses to validate this input image[8].It can therefore be seenthat face detections output (the detected face) is the input tothe face recognizer and the face Recognition's output is the final decision i.e. face known or face unknown.

Face Detection

A face Detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. Face detection can be performed based on several cues: skin color (for faces incolor images and videos, motion (for faces in videos), facial/head shape, facial appearance or a combination of these parameters. Most face detection algorithms are appearance based without using other cues. An input image is scanned at all possible locations and scales by a sub window. Face detection isposed as classifying the pattern in the sub window either as a face or a non-face. The face/nonface classifier is learned from face and non-face training examples using statistical learning methods[9]. Most modern algorithms are based on the Viola Jones object detection framework, which is based on Haar Cascades.

Viola-Jones algorithm which was introduced by P. Viola, M. J. Jones (2001) is the most popular algorithm to localize the face segment from static images or video

frame. Basically the conceptof Viola-Jones algorithm consists of four parts. The first part is known as Haar feature, second part is where integral image iscreated, followed by implementation of Adaboost on the third part and lastly cascading process.

Viola-Jones algorithm analyses a given image using Haar features consisting of multiple rectangles (Mekha Joseph et al., 2016).

In the fig shows several types of Haar features. The features perform as window function mapping onto the image. A single value result, which representing each feature can be computed bysubtracting the sum of the white rectangle(s) from the sum of the black rectangle(s) (Mekha Joseph et al., 2016).

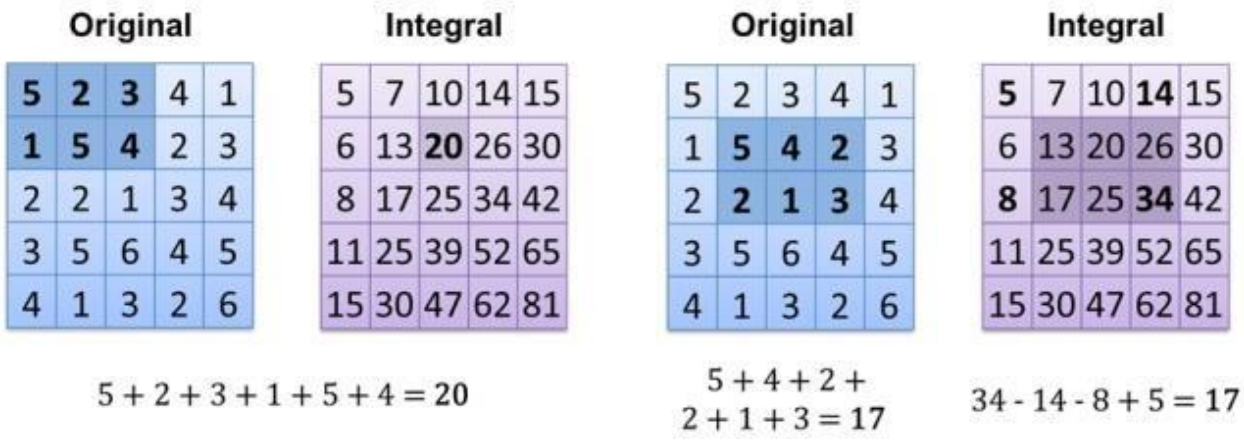


Fig: Integral of Image

The value of integrating image in a specific location is the sumof pixels on the left and the top of the respective location. Inorder to illustrate clearly, the value of the integral image at location 1 is the sum of the pixels in rectangle A. The values of integral image at the rest of the locations are cumulative.

For instance, the value at location 2 is summation of A and B, (A + B), at location 3 is summation of A and C, (A + C), and at location 4 is summation of all the regions, (A + B + C + D) [11].Therefore, the sum within the D region can be

computed with onlyaddition and subtraction of diagonal at location 4 + 1 − (2 + 3) to eliminate rectangles A, B and C.

Local Binary Patterns Histogram

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers theresult as a binary number.

It was first described in 1994 (LBP) and has since been found tobe a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with asimple data vector.

LBPH algorithm work step by step:

LBPH algorithm work in 5 steps.

1. **Parameters:** the LBPH uses 4 parameters:
    - **Radius:** the radius is used to build the circular localbinary pattern and represents the radius around the central pixel. It is usually set to 1.
    - **Neighbors:** the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computationalcost. It is usually set to 8.
    - **Grid X:** the number of cells in the horizontal direction.The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
    - **Grid Y:** the number of cells in the vertical direction.The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2. **Training the Algorithm:** First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We needto also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.
3. **Applying the LBP operation:** The first computational step ofthe LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radiusand neighbors.

The image below shows this procedure:

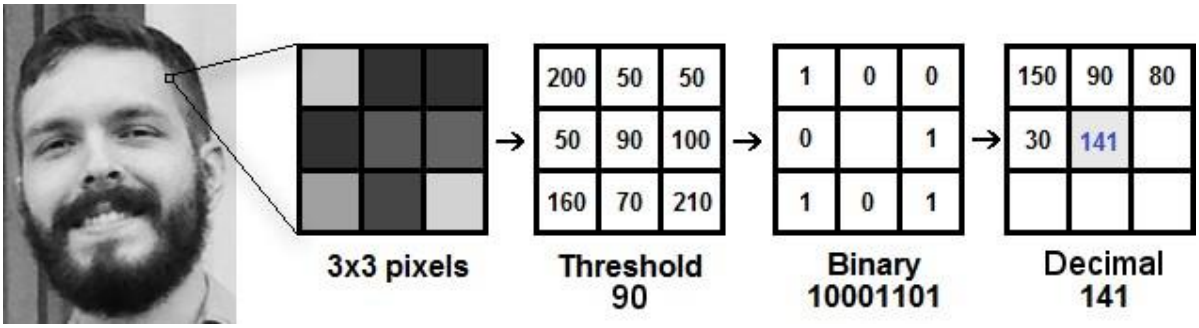


Fig: LBP Operation

Based on the image above, let's break it into several small

steps so we can understand it easily:

2.6.1 Suppose we have a facial image in grayscale.

2.6.2 We can get part of this image as a window of 3x3 pixels.

2.6.3 It can also be represented as a 3x3 matrix containing theintensity of each

pixel (0~255).

2.6.4 Then, we need to take the central value of the matrix to beused as the threshold.

2.6.5 This value will be used to define the new values from the 8neighbors.

2.6.6 For each neighbor of the central value (threshold), we seta new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.

2.6.7 Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line intoa new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.

2.6.8 Then, we convert this binary value to a decimal value andset it to the central value of the matrix, which is actually a pixel from the original image.

2.6.9 At the end of this procedure (LBP procedure), we have a newimage which represents better the characteristics of the original image.

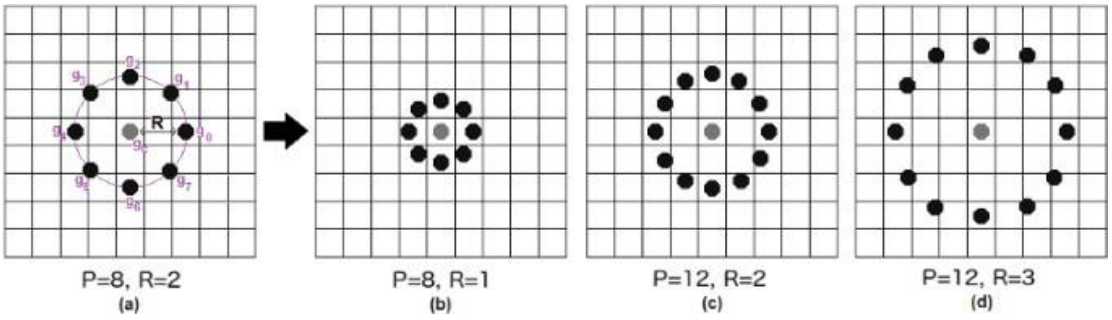


Fig: The LBP operation Radius Change

It can be done by using bilinear interpolation. If some datapoint is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. **Extracting the Histograms:** Now, using the image generatedin the last

step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids,

as can be seen in the following image:

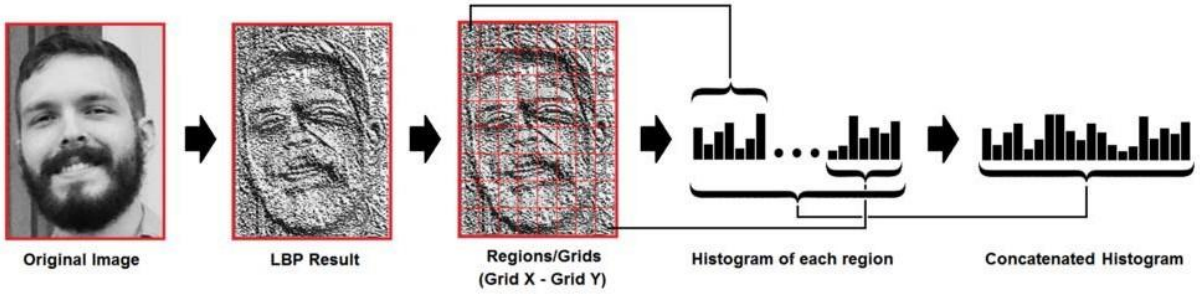


Fig: Extracting The Histogram

Based on the image above, we can extract the histogram of eachregion as follows:

- As we have an image in grayscale, each histogram (fromeach grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, wewill have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

5. **Performing the face recognition:** In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So,given an input image, we perform the steps again for this new image and creates a histogram which represents the image.
   - So to find the image that matches the input image we justneed to compare two histograms and return the image with the closest histogram.
   - We can use various approaches to compare the histograms (calculate the

distance between two histograms), for example: Euclidean distance, chi-square, absolute value,etc. In this example, we can use the **Euclidean distance** (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a ‘confidence’ measurement. Note: don’t be fooled about the ‘confidence’name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithmhas successfully recognized if the confidence is lowerthan the threshold defined.

Chapter 3

MODEL IMPLEMENTATION

Face detection involves separating image windows into two classes; one containing faces (turning the background (clutter).It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin color and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. Anideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).After taking the picture the system will compare the equality of the pictures in its database and give the most related result.

We will use Raspbian operating system, open CV platform and willdo the coding in python language.

Model Implementation

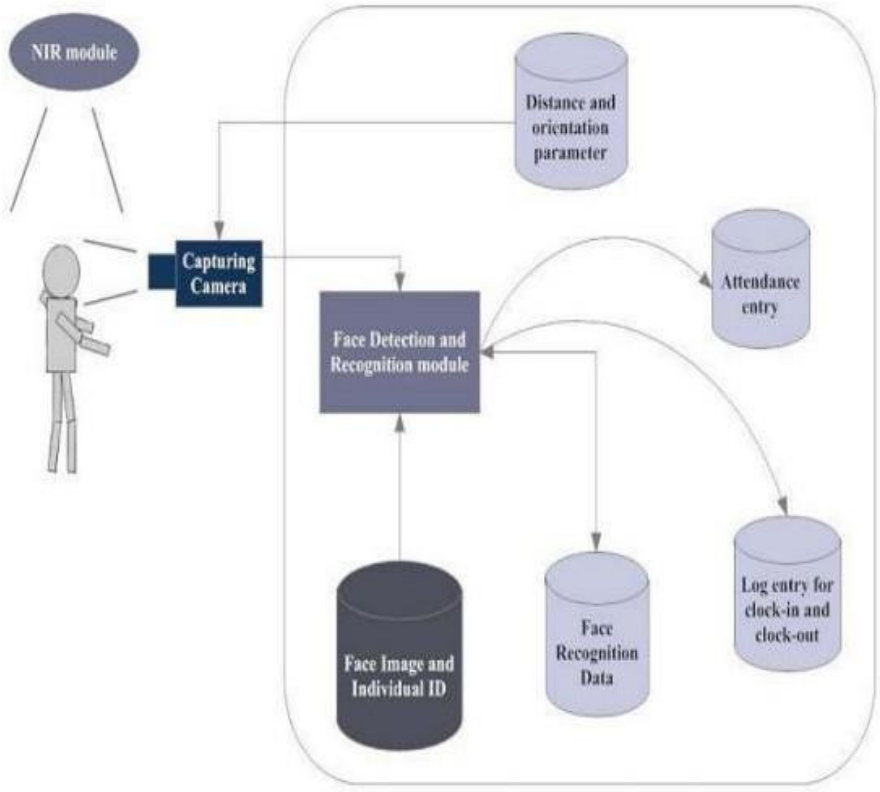


Fig: Model Implement

The main components used in the implementation approach are open source computer vision library (OpenCV). One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applicationsquickly. OpenCV library contains over 500 functions that span many areas in vision. The primary technology behind Face recognition is OpenCV. The user stands in front of the camera keeping a minimum distance of 50cm and his image is taken as an input. The frontal face is extracted from the image then converted to gray scale and stored. The Principal component Analysis (PCA) algorithm is performed on the images and the eigen values are stored in an xml file. When a user requests forrecognition the frontal face is extracted from the captured

video frame through the camera. The eigen value is re-calculatedfor the test face and it is matched with the stored data for theclosest neighbor.

Design Requirements

We used some tools to build the HFR system. Without the help ofthese tools it would not be possible to make it done. Here we will discuss about the most important one.

Software Implementation

1. **OpenCV:** We used OpenCV 3 dependency for python 3. OpenCV is library where there are lots of image processing functions are available. This is very useful library for imageprocessing. Even one can get expected outcome without writinga single code. The library is cross-platform and free for use under the open-source BSD license. Example of some supported functions are given bellow:
   - **Derivation**: Gradient / laplacian computing, contoursdelimitation
   - **Hough transforms:** lines, segments, circles, and geometrical shapes detection
   - **Histograms**: computing, equalization, and object localization with back projection algorithm
   - **Segmentation**: thresholding, distance transform, foreground / background detection, watershed segmentation
   - **Filtering**: linear and nonlinear filters, morphologicaloperations
   - **Cascade detectors**: detection of face, eye, car plates
   - **Interest points**: detection and matching
   - **Video processing:** optical flow, background subtraction, camshaft (object tracking)
   - **Photography**: panoramas realization, high definition

imaging (HDR), image inpainting So it was very important to install OpenCV. But installingOpenCV 3 is a complex process. How we did it is given below:

```
#!/bin/bash
#Usage : sudo bash./installopencv.bash
echo OpenCV 3.0.0 Raspbian Jessie auto install script - Thomas Cyrix
echo ============================================================
FILE="/tmp/out.$$"
GREP="/bin/grep"
if [ "$(id -u)" != "0" ]; then
   echo "This script must be run as root" 1>&2
   exit 1
fi
echo installing core dependencies ...
apt-get -y install cmake python3-dev python3.4-dev python3-numpy gcc build-essential cmake-curses-gui
echo installing other dependencies ...
apt-get -y install pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g
zlib1g-dev pngtools libtiff5-dev libtiff5 libtiffxx0c2 libtiff-tools libeigen3-dev
echo installing helper apps ...
apt-get -y libav-tools
#apt-get -y ffmpeg libavcodec55 libavformat55
apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs libavcodec-dev libavformat-dev libgstreamer0.10-0-dbg
libgstreamer0.10-0 libgstreamer0.10-dev libxine2-ffmpeg libxine2-dev libxine2-bin libunicap2 libunicap2-dev swig libv4l-0 libv4l-dev libpython3.4 libgtk2.0-dev
echo Receving OpenCV 3.0.0 source...
git clone --branch 3.0.0 --depth 1 https://github.com/Itseez/opencv.git
cd opencv
mkdir release
cd release
echo Preparing compilation, may take a long while...
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=$(python3 -c "import sys; print(sys.prefix)") -D PYTHON_EXECUTABLE=$(which python3) ..
echo Compiling Open CV 3.0.0, may take 2 to 36 hours
make -j4
echo Compilation Ok, installing...
make install
cd ../..
rm -r opencv
echo Completed !
echo You now can use OpenCV 3.0.0 in both Python 2 and Python 3 !
```

Fig: Installing OpenCV

```
1. Sudo chmod 755 /myfile/pi/installopencv.bash
2. sudo /myfile/pi/installopencv.bash
```

We copied this script and place it on a directory on our raspberrypi and saved it. Then through terminal we made this script executable and then ran it.

These are the command line we used.

2. **Python IDE:** There are lots of IDEs for python. Some of them arePyCharm, Thonny, Ninja, Spyder etc. Ninja and Spyder both are veryexcellent and free but we used Spyder as it feature- rich than ninja. Spyder is a little bit heavier than ninja but still much lighter than PyCharm. You can run them in pi and get GUI on your PC through ssh-Y. We installed Spyder through the command line below.

```
1.sudo apt-get isntall spyder
```

Possible Hardware Implementation

Raspberry Pi 3:

1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernetsupport (with separate PoE HAT)

Fig: Raspberry Pi 3 Model B+

Specification: The Raspberry Pi 3 Model B+ is the final revisionin the Raspberry Pi 3 range.

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header

- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreendisplay
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storingdata
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

Webcam:

ELP HD 8Megapixel USB CMOS board camera module adopt Sensor Sony(1/3.2”) IMX179 is nice to use in Linux equipment, or those equipment which come with windows, linux, Android system etc.

Fig: Webcam

Specification:

- 1/3.2 inch Sony IMX179 USB webcam
- 8 megapixel high resolution Mjpeg USB camera
- UVC usb camera, Support windows, linux, Mac with UVC, alsofor android system.
- Compatible with raspberry pi, Ubuntu, Opencv, Amcap andmany other USB web camera software and hardware.
- Webcam USB with 2.8mm lens
- 38×38/32x32mm mini micro usb board camera
- USB webcam, well used in many machines, atm machine,medical machine, automatic vending machine, industrymachine.
- USB camera module Parameters changable (Brightness, Contrast, Saturation, White Balance, Gamma, Definition,Exposure…)

Power Source:

We use Mi 10000 mAH Power Bank for our power sources.

Fig: Power Source

Project Machine:

Here is our prototype device.

Fig: Prototype Device

Experimental Results

The step of the experiments process are given below:

Face Detection:

Start capturing images through web camera of the client side:Begin:

- Pre-process the captured image and extract face image
- calculate the eigen value of the captured face image andcompared with eigen values of existing faces in the database.
- If eigen value does not matched with existing ones,save thenew face image information to the face database (xml file).
- If eigen value matched with existing one then recognitionstep will done.

Face Recognition:

Using PCA algorithm the following steps would be followed in forface recognition:

Begin:

- Find the face information of matched face image in from thedatabase.
- update the log table with corresponding face image andsystem time that makes completion of attendance for anindividual students.

This section presents the results of the experiment conducted tocapture the face into a grey scale image of 50x50 pixels.

Here is our data set sample.

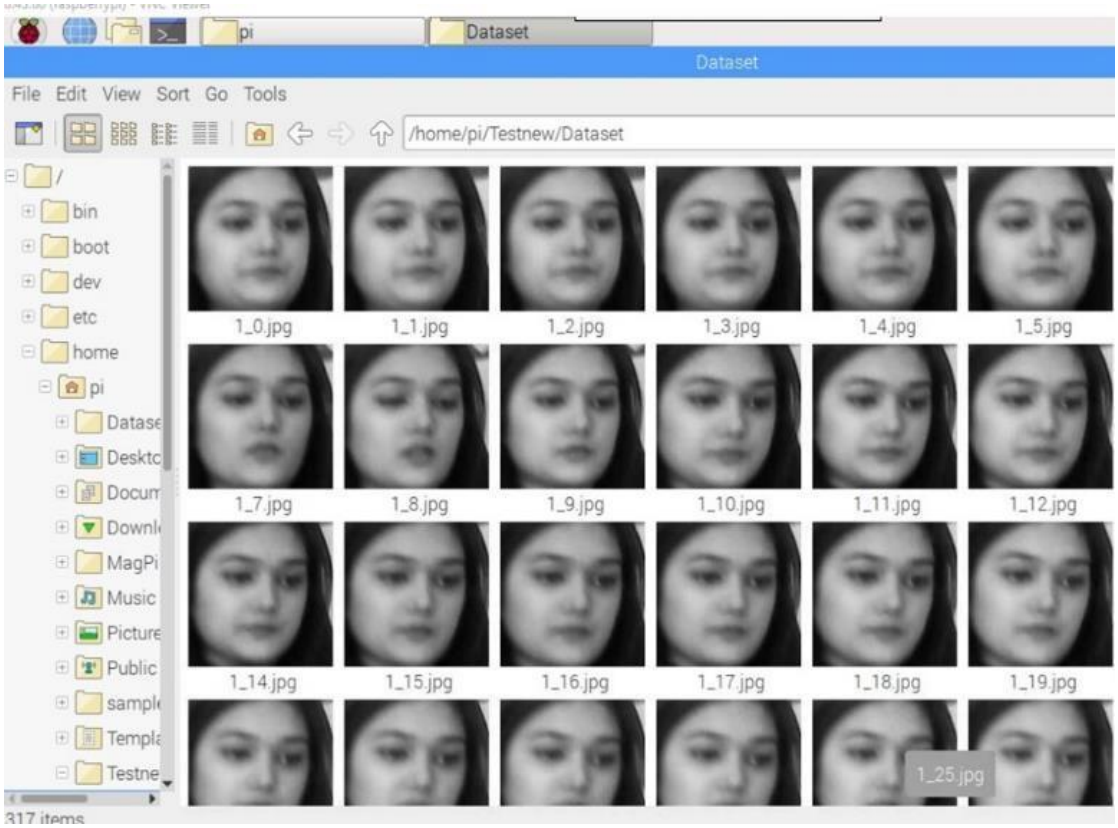


Fig: Dataset sample

Face Orientations	Detection Rate	Recognition Rate
O^0 (Frontal face)	98.7 %	95%
18º	80.0 %	78%
54º	59.2 %	58%
72º	0.00 %	0.00%
90º(Profile face)	0.00 %	0.00%

We performed a set of experiments to demonstrate the efficiency of the proposed method. 30 different images of 10 persons are used intraining set. Figure 3 shows a sample binary image detected by theExtractFace() function using Paul-Viola Face extracting Frame workdetection method.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

Sl. No	Name of the Hardware	Specification
1	Computer	Processor: Multi-core CPU RAM: 8GB or higher Storage: 500GB HDD or SSD
2	Webcam	High-resolution webcam (720p or higher)
3	Graphics Processing Unit	Nvidia GeForce GTX 1060 or Equivalent for deep learning acceleration

4.2 Software Requirements

Sl. No	Name of the Software	Specification
1	Operating System	Windows 10/11 or Linux (Ubuntu 20.04)
2	Python	Python 3.7 or later
3	Integrated Development	PyCharm, Visual Studio Code, or any Environment (IDE) Python-compatible IDE
4	Deep Learning Framework	PyTorch, TensorFlow
5	Computer Vision Library	OpenCV
6	Face Detection Library	cvzone, dlib
7	Ultralytics YOLO	YOLO (You Only Look Once) framework
8	Dataset Management Tool	LabelImg, Labelbox
9	Version Control	Git, GitHub
10	Text Editor	Notepad++, Visual Studio Code, or similar text editor

Chapter 5

Possible Algorithms

In this chapter, we will elaborate on the possible approaches and algorithms that will be leveraged in the development of our Efficient Real-Time Face Recognition-Based Attendance System. Ensuring the system's accuracy, efficiency, and real-time processing capabilities requires a detailed examination of the algorithms and methods we will utilize.

5.1 Face Detection Algorithm

Face detection is the cornerstone of any face recognition system. We are adopting a robust approach to this aspect using a combination of the dlib library and cvzone's Face DetectionModule. Dlib is renowned for its high-quality pre-trained face detector, which offers exceptional accuracy in locating faces within images. The cvzone library complements this by providing an easy-to-integrate interface for real-time face detection. This combination ensures that our system can rapidly and accurately detect faces, a critical step in attendance tracking.

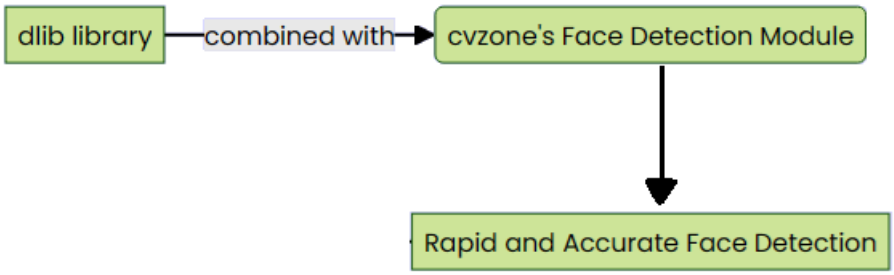


Fig. 3. Face Detection Algorithm

5.2 Blur Detection Algorithm

To enhance the reliability of our face recognition system, we have introduced a sophisticated blur detection algorithm. The presence of blurriness in face images can significantly affect the accuracy of recognition. We employ a Laplacian-

based method to quantify the blurriness of detected faces. This technique measures the variance of pixel intensities in the Laplacian of the image. Faces with excessive blurriness are excluded from the recognition process, guaranteeing that only sharp, clear images are used for attendance records. This step is essential for maintaining high recognition accuracy.

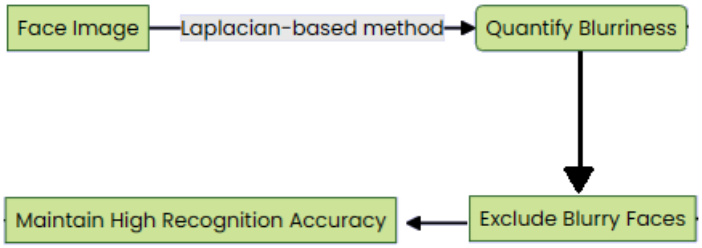


Fig. 4. Blur Detection Algorithm

5.3 YOLO (You Only Look Once) for Face Recognition

For the core face recognition task, we have chosen to implement YOLO (You Only Look Once), a real-time object detection system celebrated for its speed and accuracy. YOLO is an optimal choice for facial recognition applications due to its ability to efficiently identify and locate objects within images and video streams. In our system, YOLO performs the following steps:

- **Face Localization:** YOLO is responsible for precisely locating faces within the input frames by drawing bounding boxes around them.
- **Class Label Assignment:** It assigns class labels to each recognized face. In our context, the class labels "real" or "fake" are based on the model's training data.
- **Confidence Assessment:** YOLO calculates the confidence level associated with each recognition. When this confidence level exceeds a predefined threshold, the recognized face is further processed.

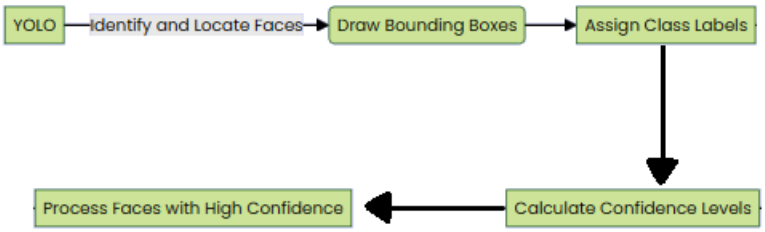


Fig. 5. YOLO for Face Recognition

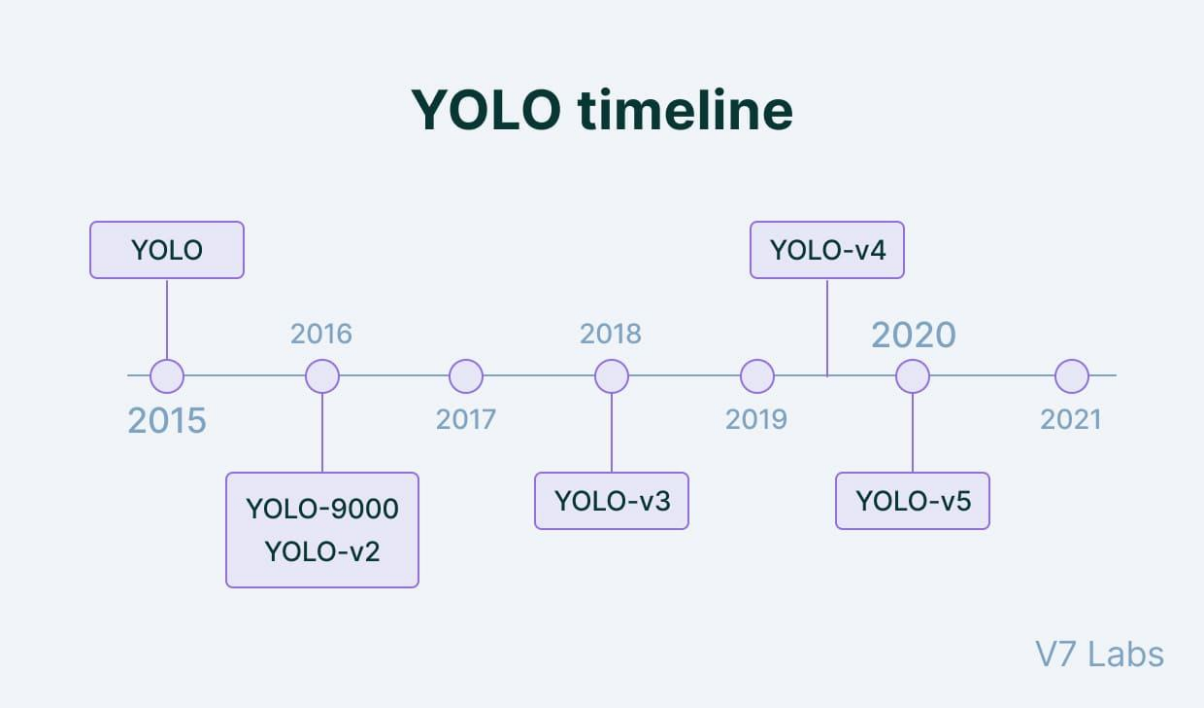
You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

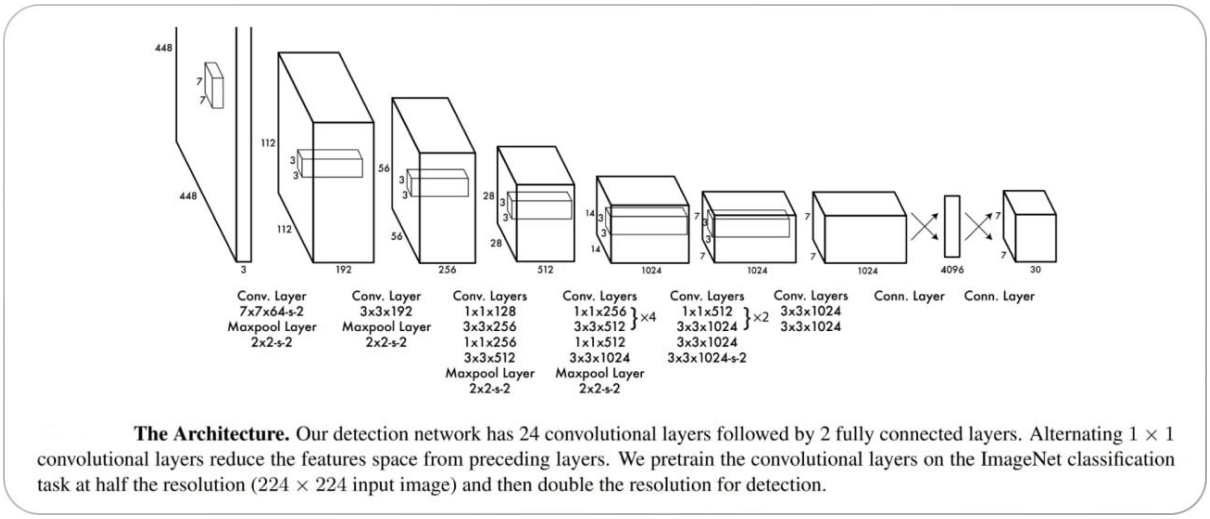
Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.

Several new versions of the same model have been proposed since the initial release of YOLO in 2015, each building on and improving its predecessor. Here's a timeline showcasing YOLO's development in recent years.



How does YOLO work? YOLO Architecture

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased

that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is **non-maximum suppression (NMS)**. NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

Now, let us look into the improvements that the later versions of YOLO have brought to the parent model.

YOLO v2

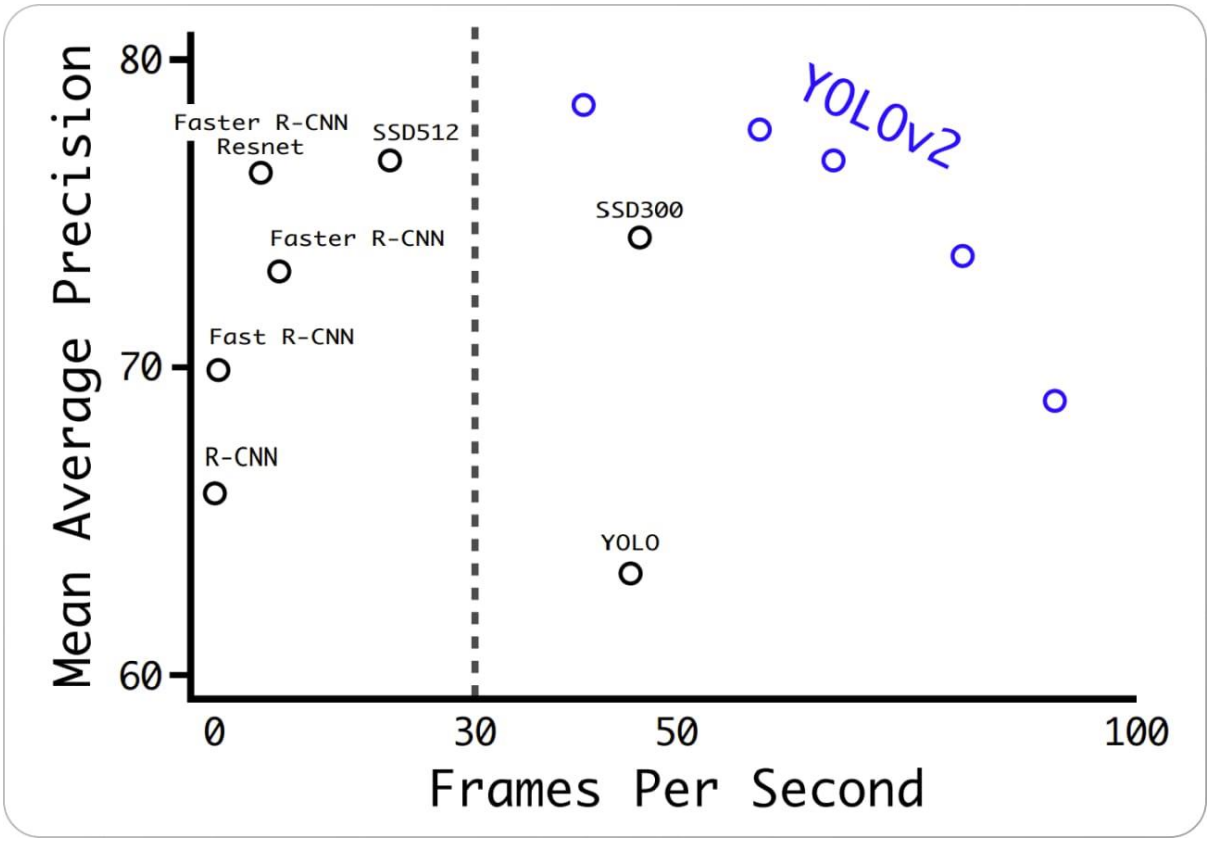
YOLO v2, also known as YOLO9000, was introduced in 2016 as an improvement over the original YOLO algorithm. It was designed to be faster and more accurate than YOLO and to be able to detect a wider range of object classes. This updated version also uses a different CNN backbone called Darknet-19, a variant of the VGGNet architecture with simple progressive convolution and pooling layers.

One of the main improvements in YOLO v2 is the use of anchor boxes. Anchor boxes are a set of predefined bounding boxes of different aspect ratios and scales. When predicting bounding boxes, YOLO v2 uses a combination of the anchor boxes and the predicted offsets to determine the final bounding box. This allows the algorithm to handle a wider range of object sizes and aspect ratios.

Another improvement in YOLO v2 is the use of batch normalization, which helps to improve the accuracy and stability of the model. YOLO v2 also uses a multi-scale training strategy, which involves training the model on images at multiple scales and then averaging the predictions. This helps to improve the detection performance of small objects.

YOLO v2 also introduces a new loss function better suited to object detection tasks. The loss function is based on the sum of the squared errors between the predicted and ground truth bounding boxes and class probabilities.

The results obtained by YOLO v2 compared to the original version and other contemporary models are shown below.



Source: Paper

YOLO v3

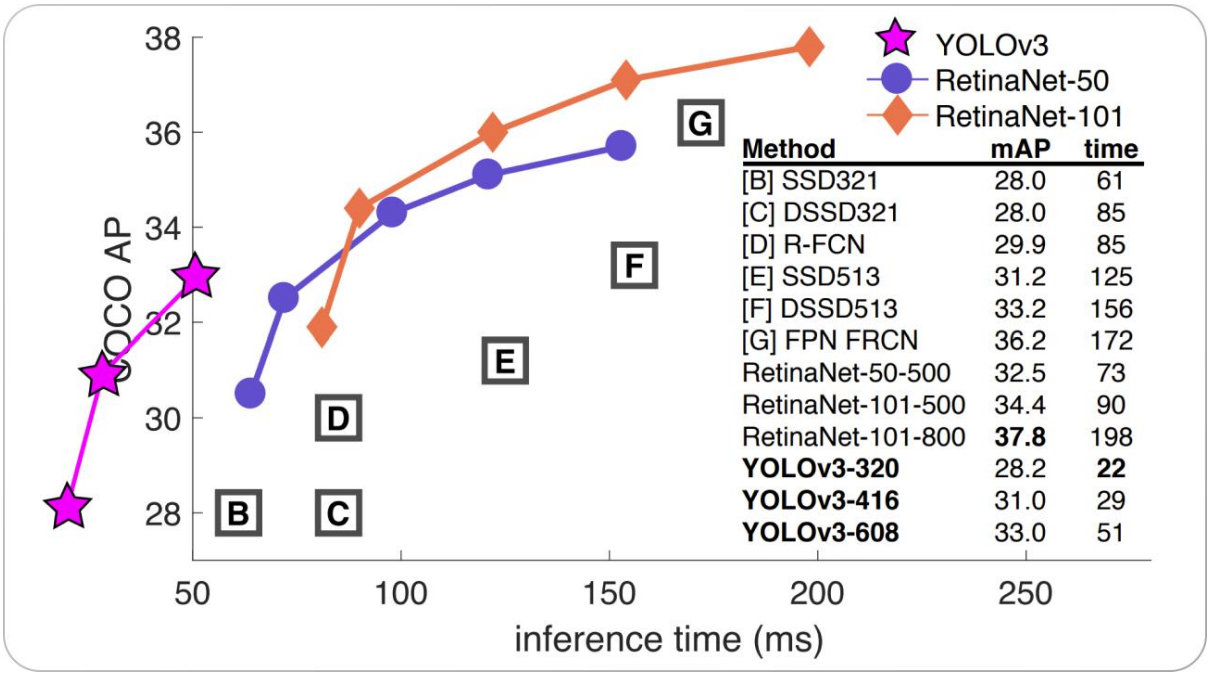
YOLO v3 is the third version of the YOLO object detection algorithm. It was introduced in 2018 as an improvement over YOLO v2, aiming to increase the accuracy and speed of the algorithm.

One of the main improvements in YOLO v3 is the use of a new CNN architecture called Darknet-53. Darknet-53 is a variant of the ResNet architecture and is designed specifically for object detection tasks. It has 53 convolutional layers and is able to achieve state-of-the-art results on various object detection benchmarks.

Another improvement in YOLO v3 are anchor boxes with different scales and aspect ratios. In YOLO v2, the anchor boxes were all the same size, which limited the ability of the algorithm to detect objects of different sizes and shapes. In YOLO v3 the anchor boxes are scaled, and aspect ratios are varied to better match the size and shape of the objects being detected.

YOLO v3 also introduces the concept of "feature pyramid networks" (FPN). FPNs are a CNN architecture used to detect objects at multiple scales. They construct a pyramid of feature maps, with each level of the pyramid being used to detect objects at a different scale. This helps to improve the detection performance on small objects, as the model is able to see the objects at multiple scales.

In addition to these improvements, YOLO v3 can handle a wider range of object sizes and aspect ratios. It is also more accurate and stable than the previous versions of YOLO.



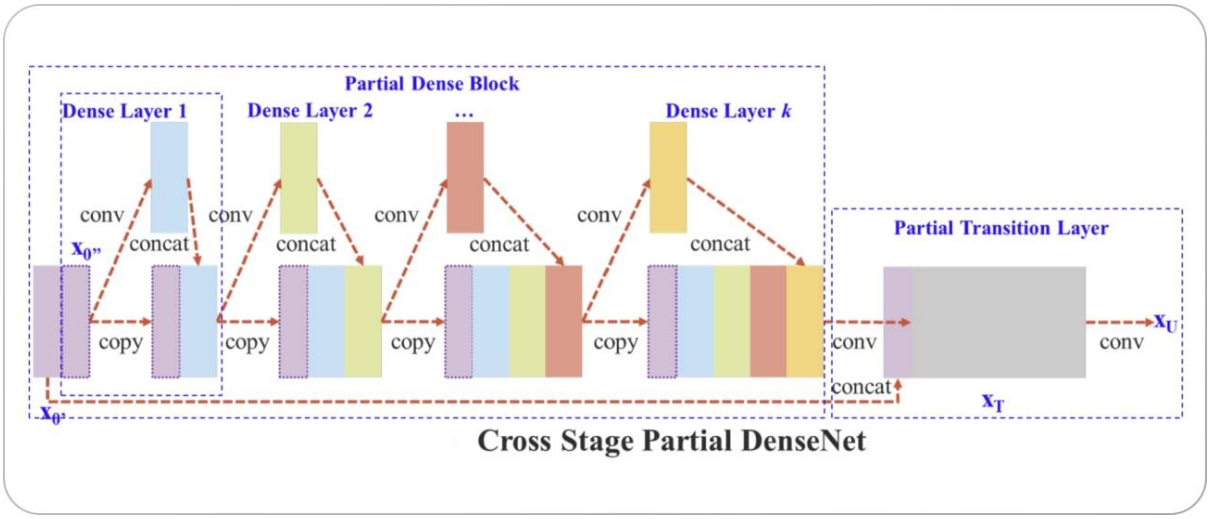
Comparison of the results obtained by YOLO v3. Source: Paper

YOLO v4

Note: Joseph Redmond, the original creator of YOLO, has left the AI community a few years before, so YOLOv4 and other versions past that are not his official work. Some of them are maintained by co-authors, but none of the releases past YOLOv3 is considered the "official" YOLO.

YOLO v4 is the fourth version of the YOLO object detection algorithm introduced in 2020 by Bochkovskiy et al. as an improvement over YOLO v3.

The primary improvement in YOLO v4 over YOLO v3 is the use of a new CNN architecture called CSPNet (shown below). CSPNet stands for "Cross Stage Partial Network" and is a variant of the ResNet architecture designed specifically for object detection tasks. It has a relatively shallow structure, with only 54 convolutional layers. However, it can achieve state-of-the-art results on various object detection benchmarks.



Architecture of CSPNet. Source: Paper

Both YOLO v3 and YOLO v4 use anchor boxes with different scales and aspect ratios to better match the size and shape of the detected objects. YOLO v4 introduces a new method for generating the anchor boxes, called "k-means clustering." It involves using a clustering algorithm to group the ground truth bounding boxes into clusters and then using the centroids of the clusters as the anchor boxes. This allows the anchor boxes to be more closely aligned with the detected objects' size and shape.

While both YOLO v3 and YOLO v4 use a similar loss function for training the model, YOLO v4 introduces a new term called "GHM loss." It's a variant of the focal loss function and is designed to improve the model's performance on imbalanced datasets. YOLO v4 also improves the architecture of the FPNs used in YOLO v3.

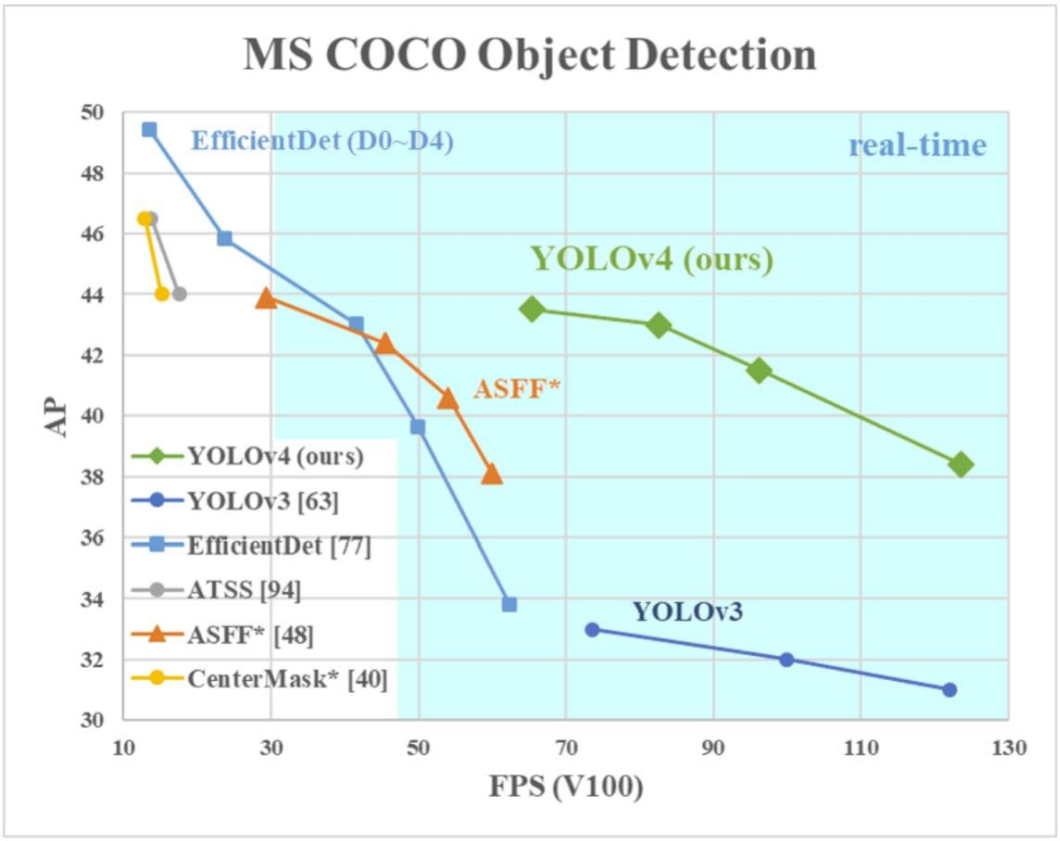


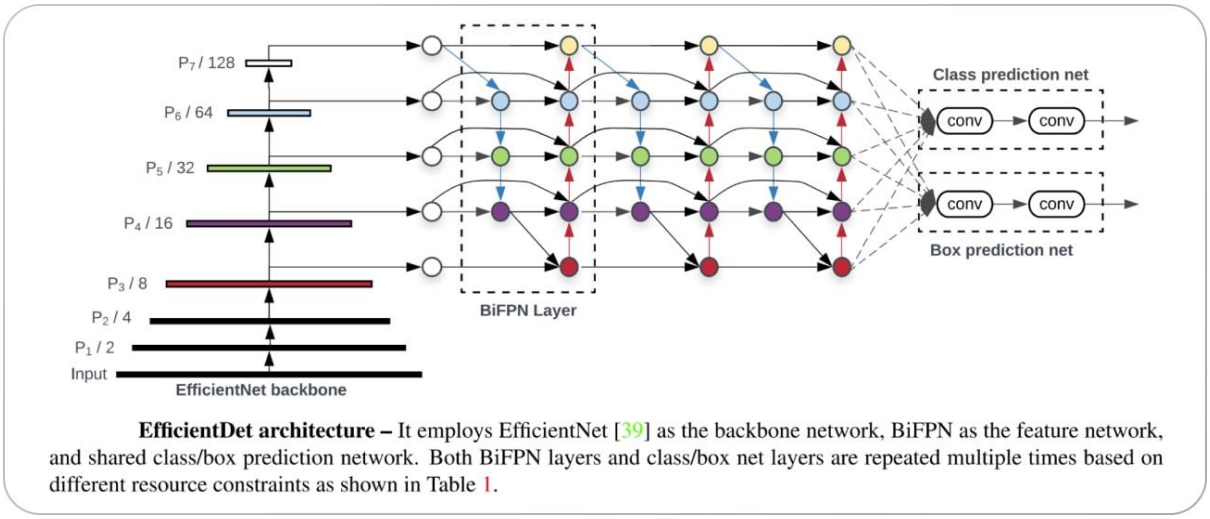
Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

Comparative performance of YOLO v4. Source: Paper

YOLO v5

YOLO v5 was introduced in 2020 by the same team that developed the original YOLO algorithm as an open-source project and is maintained by Ultralytics. YOLO v5 builds upon the success of previous versions and adds several new features and improvements.

Unlike YOLO, YOLO v5 uses a more complex architecture called EfficientDet (architecture shown below), based on the EfficientNet network architecture. Using a more complex architecture in YOLO v5 allows it to achieve higher accuracy and better generalization to a wider range of object categories.



EfficientDet architecture – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

Architecture of the EfficientDet model. Source: Paper

Another difference between YOLO and YOLO v5 is the training data used to learn the object detection model. YOLO was trained on the PASCAL VOC dataset, which consists of 20 object categories. YOLO v5, on the other hand, was trained on a larger and more diverse dataset called D5, which includes a total of 600 object categories.

YOLO v5 uses a new method for generating the anchor boxes, called "dynamic anchor boxes." It involves using a clustering algorithm to group the ground truth bounding boxes into clusters and then using the centroids of the clusters as the anchor boxes. This allows the anchor boxes to be more closely aligned with the detected objects' size and shape.

YOLO v5 also introduces the concept of "spatial pyramid pooling" (SPP), a type of pooling layer used to reduce the spatial resolution of the feature maps. SPP is used to improve the detection performance on small objects, as it allows the model to see the objects at multiple scales. YOLO v4 also uses SPP, but YOLO v5 includes several improvements to the SPP architecture that allow it to achieve better results.

YOLO v4 and YOLO v5 use a similar loss function to train the model. However, YOLO v5 introduces a new term called "CIoU loss," which is a variant of the IoU loss function designed to improve the model's performance on imbalanced datasets.

YOLO v6

YOLO v6 was proposed in 2022 by Li et al. as an improvement over previous versions. One of the main differences between YOLO v5 and YOLO v6 is the CNN architecture used. YOLO v6 uses a variant of the EfficientNet architecture called EfficientNet-L2. It's a more efficient architecture than EfficientDet used in YOLO v5, with fewer parameters and a higher computational efficiency. It can achieve state-of-the-art results on various object detection benchmarks. The framework of the YOLO v6 model is shown below.

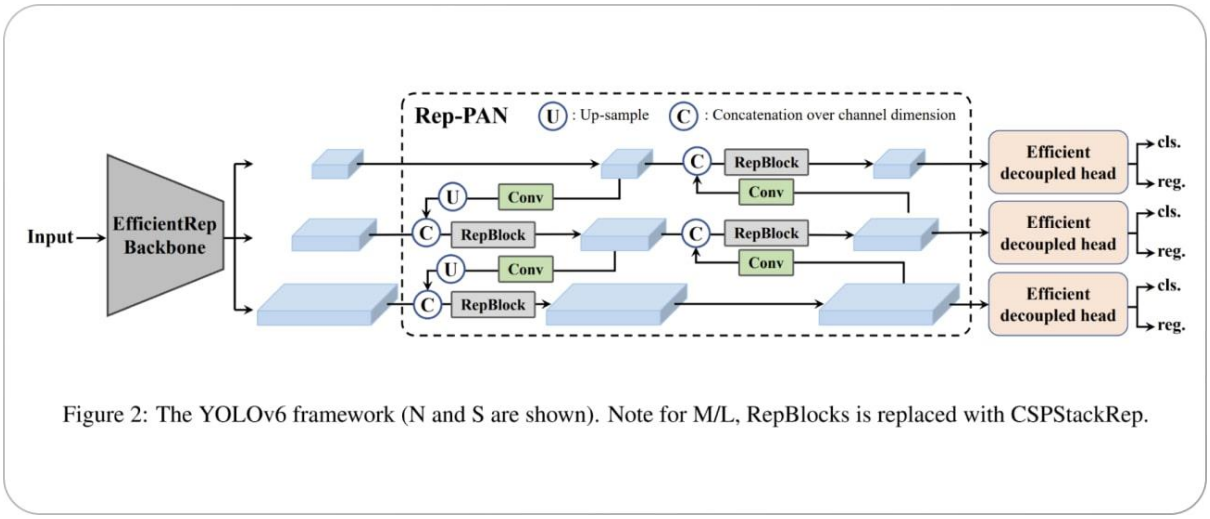
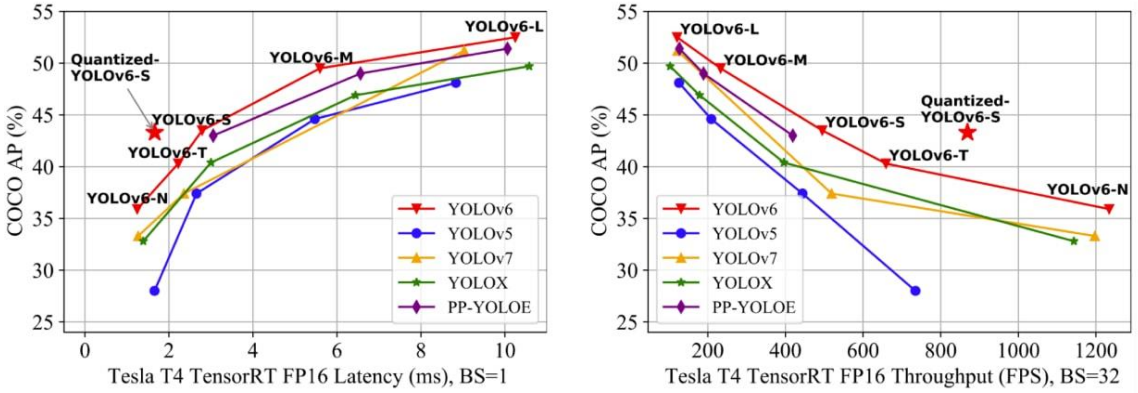


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

Overview of YOLO v6. Source: Paper

YOLO v6 also introduces a new method for generating the anchor boxes, called "dense anchor boxes."

The results obtained by YOLO v6 compared to other state-of-the-art methods are shown below.



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

What's new with YOLO v7?

YOLO v7, the latest version of YOLO, has several improvements over the previous versions. One of the main improvements is the use of anchor boxes.

Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes. YOLO v7 uses nine anchor boxes, which allows it to detect a wider range of object shapes and sizes compared to previous versions, thus helping to reduce the number of false positives.

Here is YOLO v7 in action:

A key improvement in YOLO v7 is the use of a new loss function called "focal loss." Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss battles this issue by down-weighting the loss for well-classified examples and focusing on the hard examples—the objects that are hard to detect.

YOLO v7 also has a higher resolution than the previous versions. It processes images at a resolution of 608 by 608 pixels, which is higher than the 416 by 416 resolution used in YOLO v3. This higher resolution allows YOLO v7 to detect smaller objects and to have a higher accuracy overall.

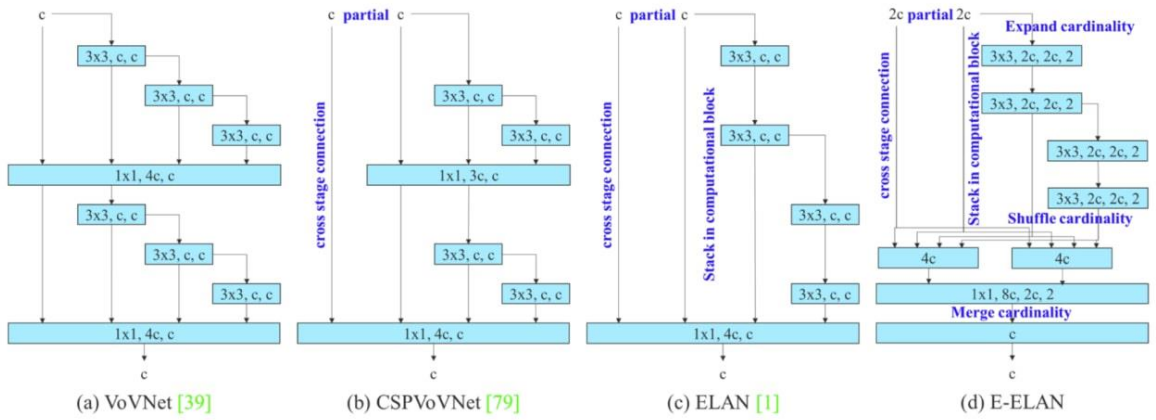
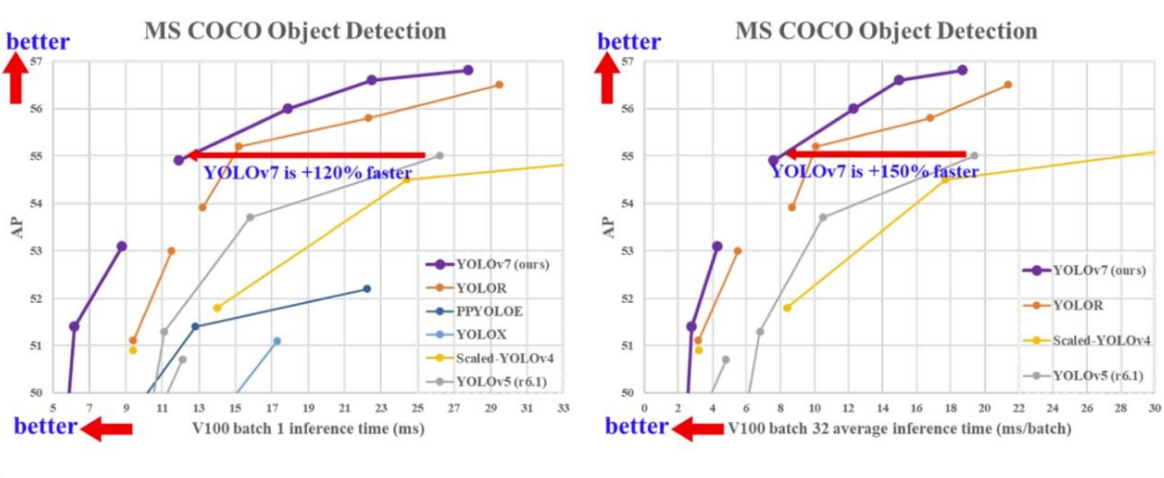


Figure 2: Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different feature maps and improve the use of parameters and calculations.

Change in the layer aggregation scheme of YOLO v7 for efficient object feature learning. Source: Paper

One of the main advantages of YOLO v7 is its speed. It can process images at a rate of 155 frames per second, much faster than other state-of-the-art object detection algorithms. Even the original baseline YOLO model was capable of processing at a maximum rate of 45 frames per second. This makes it suitable for sensitive real-time applications such as surveillance and self-driving cars, where higher processing speeds are crucial.



Comparison in performance and inference speed of YOLO v7 with contemporary state-of-the-art real-time object detectors. AP = Average Precision. Source: Paper

Regarding accuracy, YOLO v7 performs well compared to other object detection algorithms. It achieves an average precision of 37.2% at an IoU (intersection over union) threshold of 0.5 on the popular COCO dataset, which is comparable to other state-of-the-art object detection algorithms. The quantitative comparison of the performance is shown below.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Source: Paper

However, it should be noted that YOLO v7 is less accurate than two-stage detectors such as Faster R-CNN and Mask R-CNN, which tend to achieve higher average precision on the COCO dataset but also require longer inference times.

Limitations of YOLO v7

YOLO v7 is a powerful and effective object detection algorithm, but it does have a few limitations.

1. YOLO v7, like many object detection algorithms, struggles to detect small objects. It might fail to accurately detecting objects in crowded scenes or when objects are far away from the camera.
2. YOLO v7 is also not perfect at detecting objects at different scales. This can make it difficult to detect objects that are either very large or very small compared to the other objects in the scene.
3. YOLO v7 can be sensitive to changes in lighting or other environmental conditions, so it may be inconvenient to use in real-world applications where lighting conditions may vary.
4. YOLO v7 can be computationally intensive, which can make it difficult to run in real-time on resource-constrained devices like smartphones or other edge devices.

YOLO v8

At the time of writing this article, the release of YOLO v8 has been confirmed by Ultralytics that promises new features and improved performance over its predecessors. YOLO v8 boasts of a new API that will make training and inference much easier on both CPU and GPU devices and the framework will support previous YOLO versions. The developers are still working on releasing a scientific paper that will include a detailed description of the model architecture and performance.

5.4 Data Splitting and Offline Training

To ensure the model's robustness and its ability to generalize effectively, we carefully split the collected data into distinct sets for training, validation, and testing. This strategic data splitting helps mitigate overfitting and ensures that the model can generalize its learning accurately. To achieve this, we employ the Ultralytics YOLO framework, streamlining the training process:

- **Data Splitting:** The collected face data is partitioned into three subsets, facilitating training, validation, and testing.
- **Offline Training:** The YOLO model is trained on this labeled face data. Through this training process, it learns to recognize the characteristics of real and fake faces, ultimately enhancing recognition accuracy.

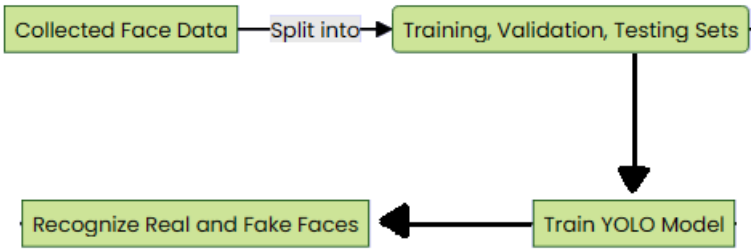


Fig. 6. Data Splitting and Offline Training

5.5 Real-Time Face Recognition Workflow

In the operational phase of the system, real-time processing capabilities are crucial. The sequence of operations for real-time face recognition is as follows:

- **Frame Processing:** The webcam continuously captures frames, which are processed in real time by the system.
- **Face Recognition:** The YOLO model identifies faces within these frames and calculates their confidence levels.
- **Class Labeling:** Based on the training data, recognized faces are labeled as either "real" or "fake."
- **Attendance Recording:** The system records attendance data in real time, based on the recognized faces, thus serving the core purpose of the system.

5.6 Summary and Integration

In summary, this chapter has provided a comprehensive exploration of the key algorithms and methodologies that will be integrated into our Efficient Real-Time

Face Recognition-Based Attendance System. Each component plays a pivotal role in ensuring the system's accuracy and efficiency. From precise face detection to blurriness assessment and YOLO-based recognition, these algorithms and methods are fundamental in the creation of a robust real-time face recognition system. They ensure that the system meets its objectives while processing data in real time.

The subsequent chapter will delve into the specifics of the dataset utilized for training and validation, an essential element in the development of a successful face recognition system.

Chapter 6

Methodology

In this chapter, we present the methodology that will guide the implementation of the "Efficient Real-Time Face Recognition-Based Attendance System with Deep Learning Algorithms." The methodology outlines the step-by-step process that will be followed to achieve the project's objectives.

6.1 Project Development Life Cycle

The project will adhere to a structured development life cycle to ensure efficient planning, execution, and monitoring of the system's implementation. The chosen life cycle model is the Iterative and Incremental Development Model, which allows for flexibility and adaptability during the project's evolution.

6.2 System Design and Architecture

Before any development begins, a detailed system design and architecture will be created. This includes defining the software and hardware components, data flow, and user interfaces. The system will be designed to be modular and scalable, accommodating future enhancements and adaptations.

6.3 Data Collection and Preprocessing

The success of a deep learning-based system heavily relies on the quality and diversity of the training data. To address this, the following steps will be taken:

- Data Collection: A diverse dataset of facial images will be collected from multiple sources to ensure a wide range of faces and conditions are represented.
- Data Labelling: Each image in the dataset will be accurately labelled with the identity of the person in the image.

- Data Augmentation: To increase the dataset's size and diversity, data augmentation techniques will be applied, including image rotation, scaling, and brightness adjustments.

6.4 Deep Learning Model Selection and Training

The core of the system's intelligence is the deep learning model for face recognition. This phase involves:

- Model Selection: Various deep learning architectures, such as Convolutional Neural Networks (CNNs) and Deep Face Recognition (DFR) networks, will be evaluated to select the most suitable model for the task.
- Model Training: The chosen model will be trained using the labelled dataset, with a focus on optimizing parameters to achieve high recognition accuracy.

6.5 Real-Time System Development

Creating a real-time face recognition system that operates swiftly and efficiently is crucial. This step includes:

- Integration with Hardware: The system will be integrated with the necessary hardware components, including webcams and GPUs, to enable real-time processing.
- Frame Capture and Processing: The system will continuously capture frames from the webcam, process them in real-time, and detect faces using the selected deep learning model.

6.6 User Interface Development

To ensure user-friendliness, the following steps will be taken:

- Graphical User Interface (GUI): A user-friendly graphical interface will be developed to facilitate system setup and operation.
- Reporting and Visualization: The system will provide real-time attendance reports and visualizations for users, making it easy to track attendance.

6.7 System Testing and Quality Assurance

Before deployment, thorough testing will be conducted:

- Unit Testing: Individual components of the system will be tested for correctness and reliability.
- Integration Testing: The integration of various components will be tested to ensure seamless operation.
- User Acceptance Testing (UAT): The system will be evaluated by end-users to confirm that it meets their requirements.

6.8 Deployment and Maintenance

After successful testing, the system will be deployed in the intended environments, such as educational institutions and workplaces. Regular maintenance will be carried out to address issues, update models, and ensure optimal performance.

6.9 Ethical Considerations and Security

Throughout the implementation process, ethical considerations and security measures will be rigorously upheld. These include privacy protection, data security, and ensuring that the system is used in an ethical and responsible manner.

This methodology provides a structured and comprehensive plan for implementing the Efficient Real-Time Face Recognition-Based Attendance

System. It ensures that the project progresses smoothly from design and data collection to real-time system development and deployment, all while maintaining ethical and security standards.

CHAPTER 7
RESULT

Sample Images:

The Attendance Camera application redefines attendance management by seamlessly integrating facial recognition technology into the check-in process. With a minimalist interface and intuitive controls, the application efficiently scans its surroundings for individuals, capturing facial features with precision as soon as they enter its field of view. Instantaneously comparing captured data against a secure database, the application verifies identities and records attendance with remarkable speed, eliminating the need for manual input. Prioritizing user privacy, the system employs robust encryption protocols for secure data storage. Its versatility allows seamless integration into various organizational settings, offering unparalleled efficiency and reliability in attendance tracking.

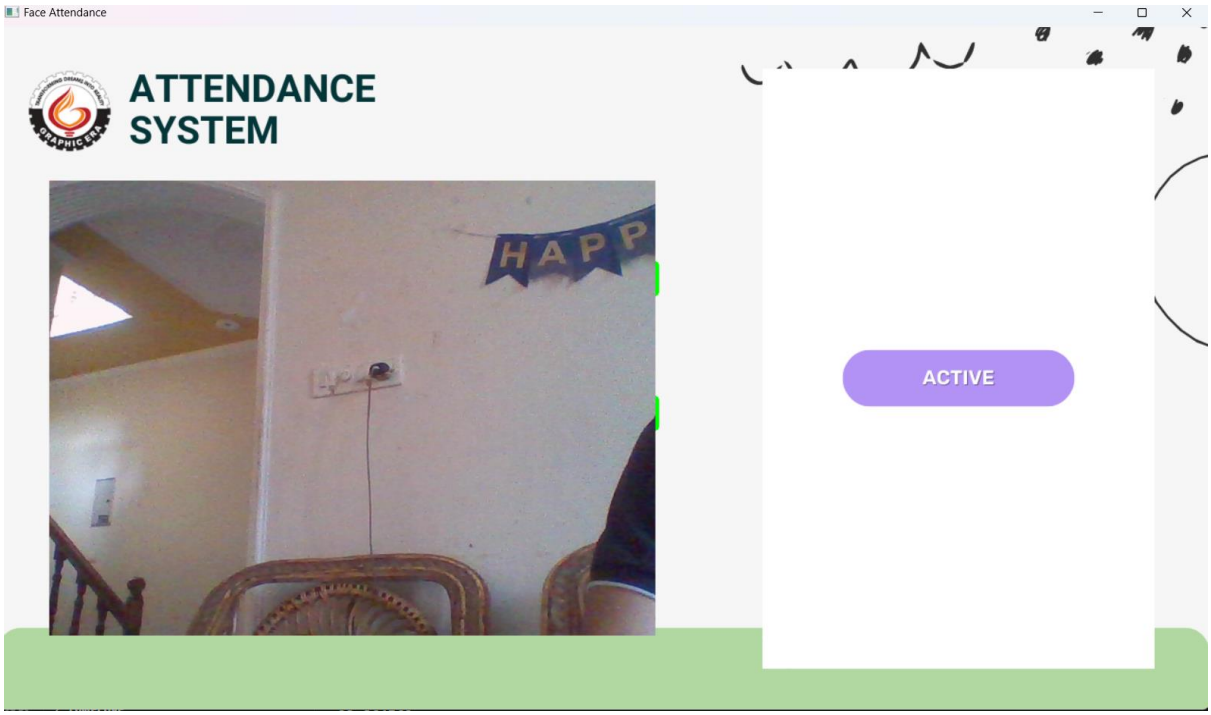


Fig: Camera in Active State

Our cutting-edge attendance marking system harnesses the power of instant student recognition, revolutionizing traditional attendance processes. Seamlessly integrated into our user-friendly interface, the system instantly identifies students as they come into view, ensuring accurate attendance tracking with unprecedented efficiency. With a simple glance, instructors can mark students present, eliminating the need for manual attendance sheets and tedious data entry. This streamlined approach not only saves valuable instructional time but also reduces administrative burdens, allowing educators to focus on what truly matters: teaching. Embracing the latest advancements in facial recognition technology, our system prioritizes both accuracy and privacy, ensuring that attendance records are securely stored and accessible only to authorized personnel. Experience the future of attendance management with our instant student recognition system—where efficiency meets reliability.

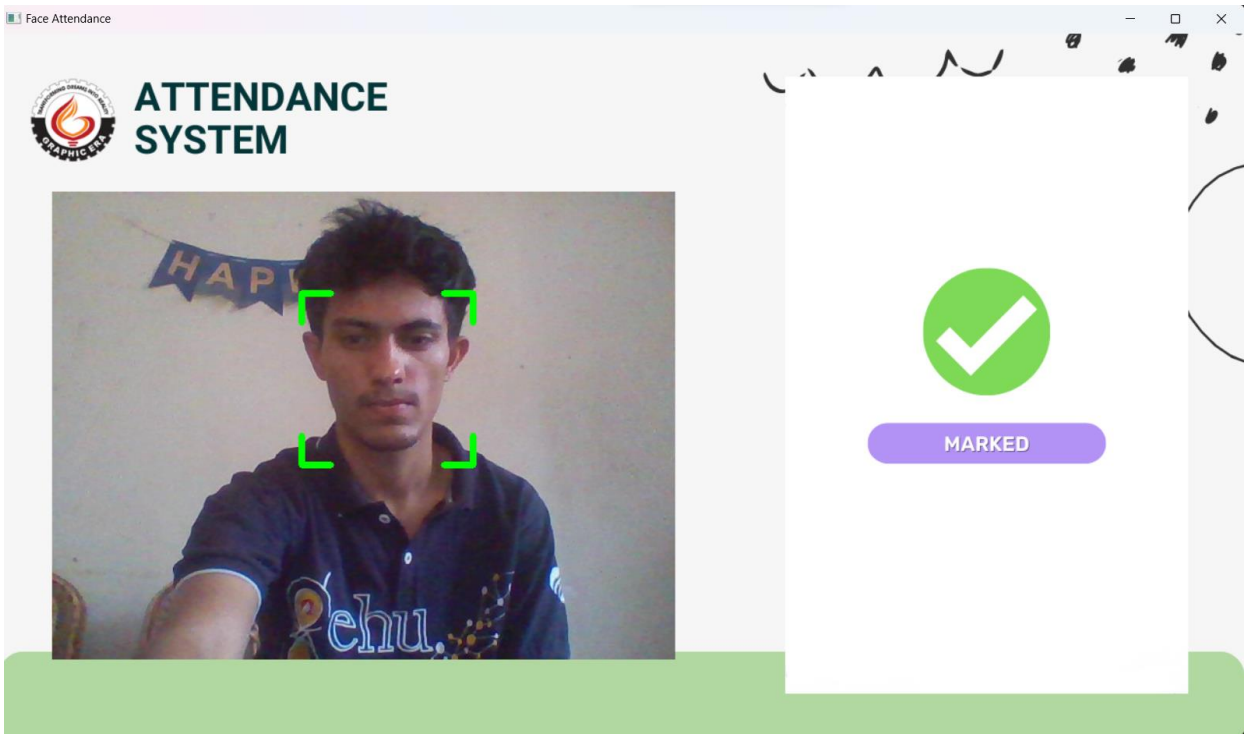


Fig: Showing Attendance Marked

Our attendance solution goes beyond mere marking—it offers instant visibility into student presence, enhancing classroom management and instructional efficiency. Once attendance is marked, our intuitive interface provides real-time updates, enabling educators to monitor class participation with unparalleled ease. Whether in-person or remote, instructors can access attendance data instantly, facilitating timely interventions and personalized support for students. Designed with user convenience in mind, our system offers seamless integration with existing workflows, eliminating the need for manual data entry and tedious paperwork. By leveraging advanced technology, including cloud-based storage and encryption protocols, we ensure the security and integrity of attendance records. Experience the power of real-time attendance tracking and elevate your teaching experience with our innovative solution.

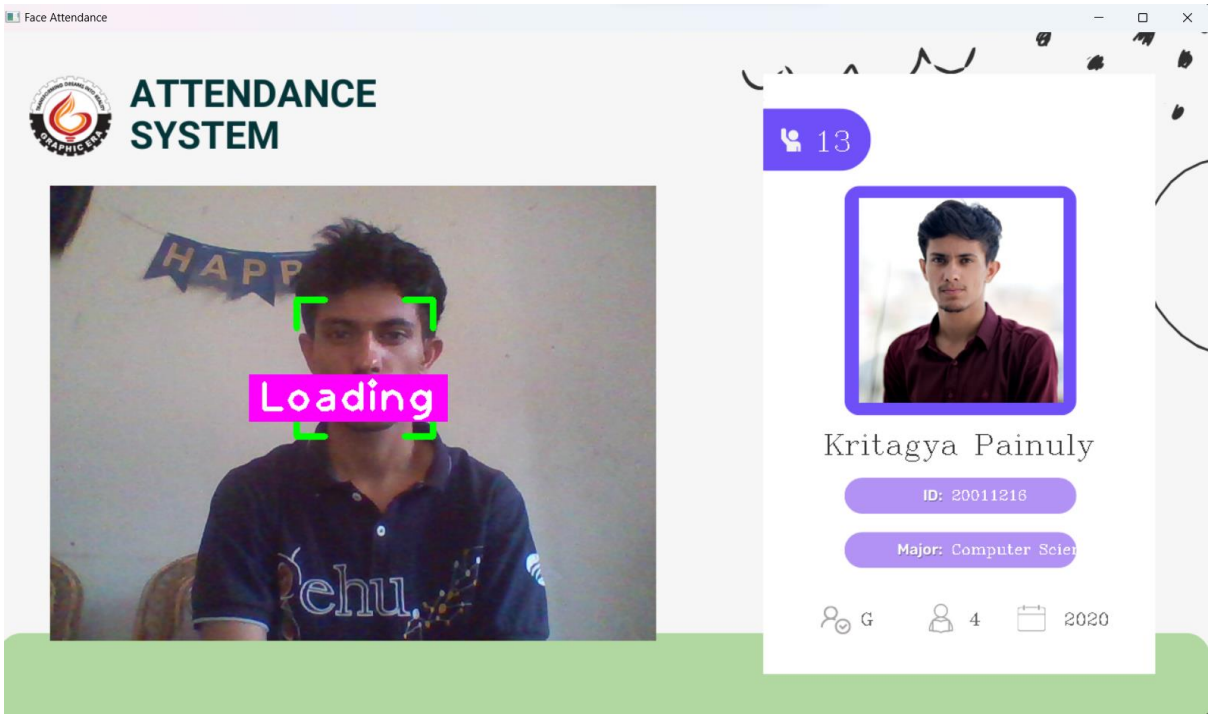
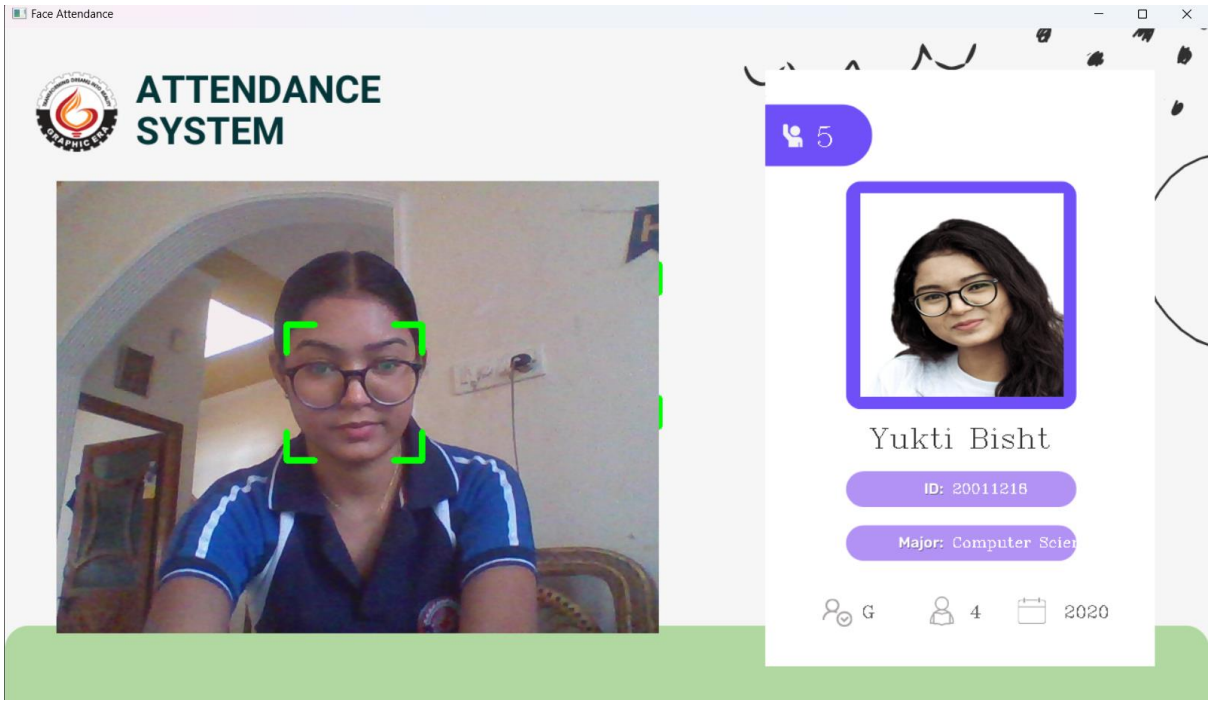


Fig: Showing Total Attendance

With our attendance solution, staying informed about student presence is effortless and instantaneous. Even after marking attendance for one student, our system enables educators to capture a snapshot of the entire class's attendance status at a glance. Whether reviewing attendance in real-time or analyzing past sessions, our intuitive interface offers comprehensive visibility into each student's participation. Seamlessly integrated into existing workflows, our system eliminates the need for manual tracking, empowering educators to focus on teaching rather than administrative tasks. Backed by robust security measures, including encrypted data storage, we prioritize the confidentiality and integrity of attendance records. Experience the efficiency and precision of attendance tracking with our innovative solution, ensuring every student's presence is accounted for with accuracy and reliability.



Fig; Showing Total Attendance

Our attendance solution offers a seamless experience by automatically flagging students as 'already marked' upon re-entry. Whether a student briefly steps out and returns or attempts to check-in multiple times, our system intelligently detects previous attendance records, preventing duplicate entries. This feature not only enhances the accuracy of attendance tracking but also streamlines classroom management, allowing educators to focus on delivering engaging lessons. With our intuitive interface, instructors can easily identify students who have already been accounted for, ensuring efficient and precise attendance management. Experience the convenience and reliability of our system, where every student's presence is accurately tracked with minimal effort.

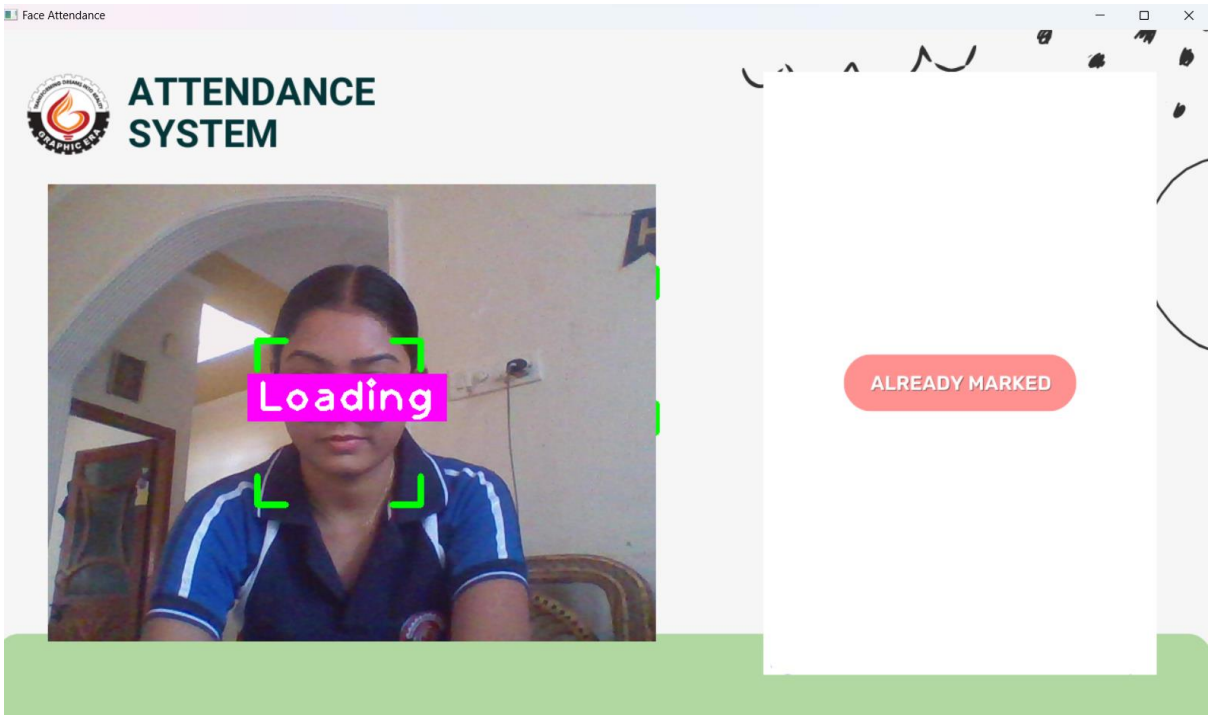


Fig: Showing Already Marked

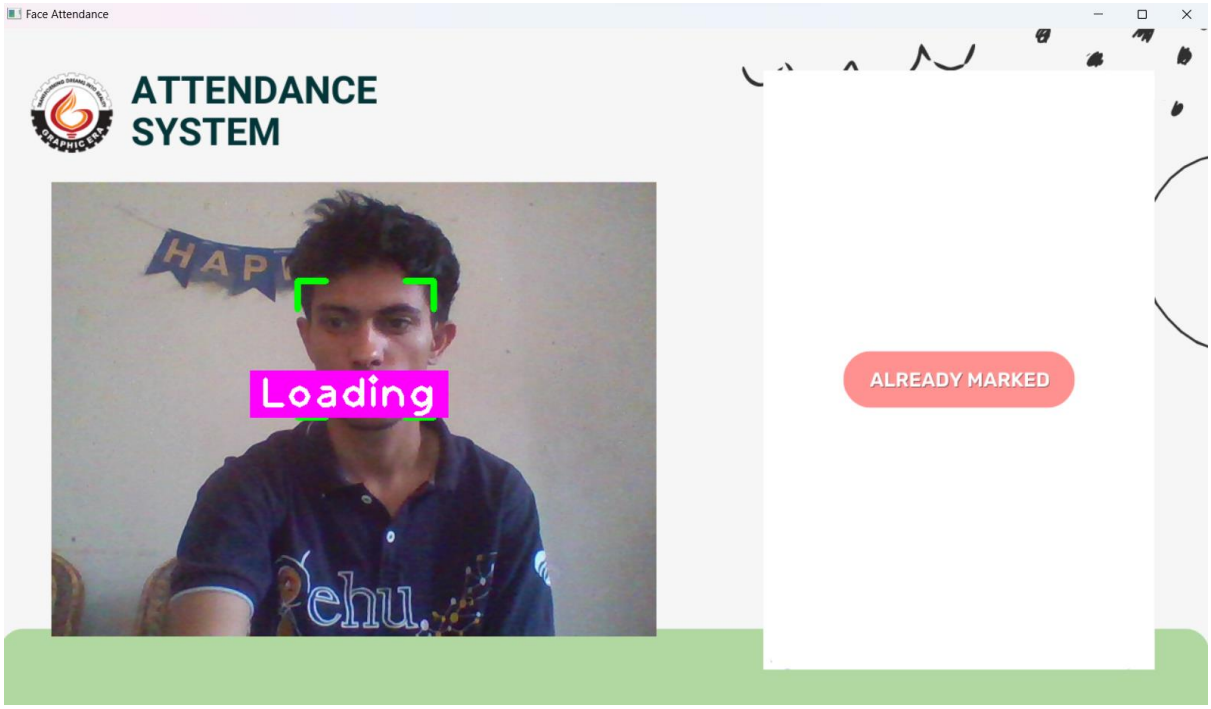


Fig: Showing Already Marked

Our Firebase attendance database serves as a central repository for storing student images and attendance records, revolutionizing the way educational institutions manage their data. Designed with security and accessibility in mind, this robust platform offers a secure environment for housing sensitive information, including facial images and attendance logs.

Each student's profile is meticulously curated within the database, featuring their unique facial image for streamlined identification. Leveraging Firebase's versatile storage capabilities, these images are securely stored and easily retrievable, facilitating seamless integration with attendance tracking systems.

In addition to student images, our database meticulously logs attendance records in real-time, providing educators with actionable insights into class participation. Whether accessed through a web interface or mobile application, instructors can view and analyze attendance data with unparalleled ease, enabling timely interventions and personalized support for students.

Furthermore, Firebase's robust authentication and authorization mechanisms ensure that only authorized personnel have access to sensitive information, safeguarding student privacy and confidentiality.

With our Firebase attendance database, educational institutions can embrace the power of technology to streamline attendance management, enhance data security, and empower educators with actionable insights for student success. Experience the future of education data management with our comprehensive and reliable solution.

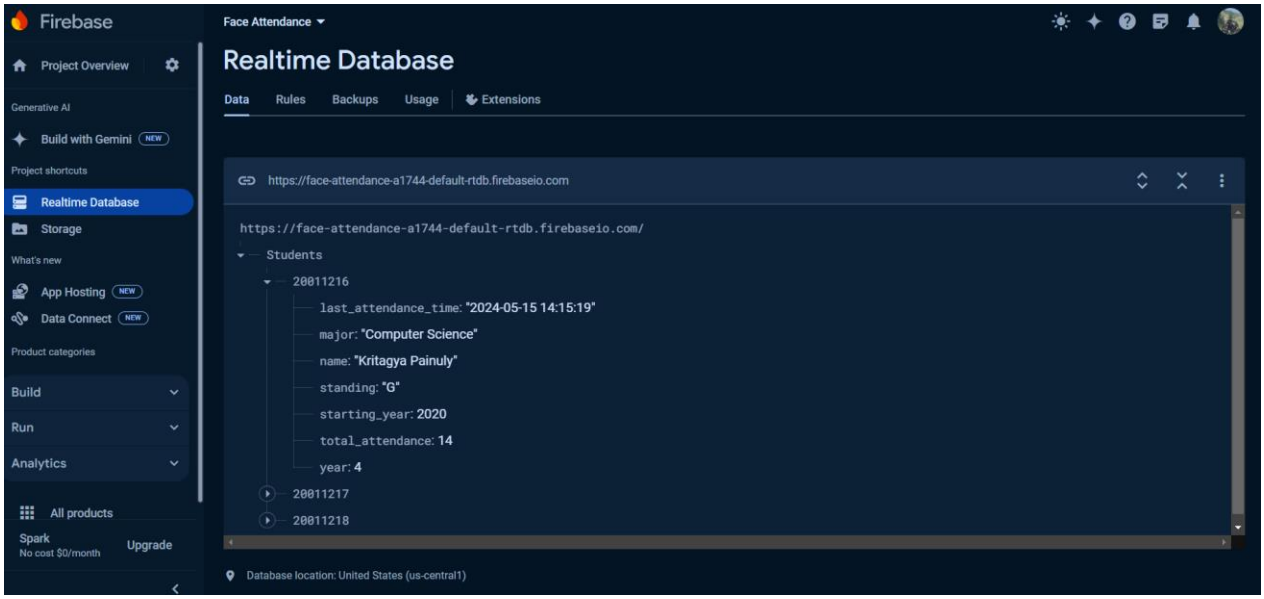


Fig: Firebase Database

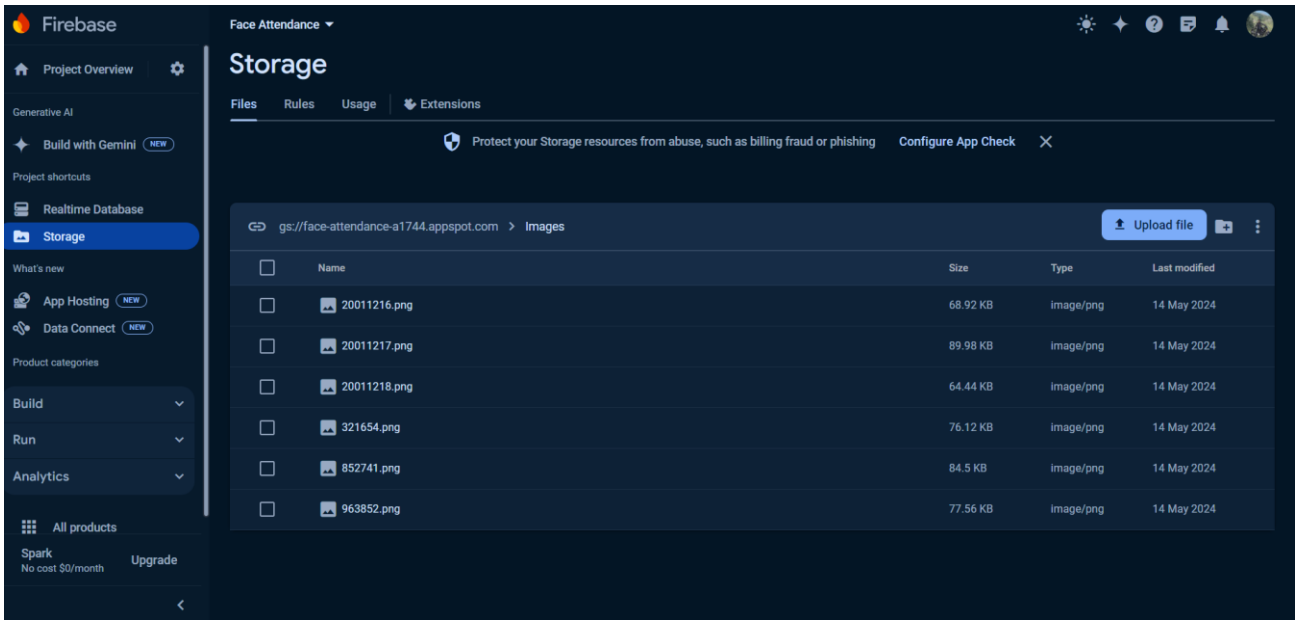


Fig: Firebase Storage

CHAPTER 8
FUTURE SCOPE

Introduction

This chapter discusses the future scope or the implementation of this robot. To incerease the scope of this device we can add somenew features. As technology is becoming more advance it will be mandatory to change the sturctute some day with better replacementand sometimes based on customer requirements.

Future Scope of Work

There are so many future scope on this project. Some of them are

- Can improve security.
- Can use Neural Network for high accuracy.
- Can used in big factory or employee attendance.
- Can build on fully web base system.

Summary

This chapter has described the possible future applications of thedesign. But there are a lot of possibilities with the designed device. The device may need some research for different applications, though the principle of the designed system will remain as it is.

APPENDIX A

This Python script, Main.py, orchestrates an advanced facial recognition-based attendance system integrated with Firebase for data management. Here's a detailed description:

The script begins by importing necessary libraries such as OpenCV, Firebase, and face_recognition. These libraries provide functionalities for image processing, database management, and facial recognition.

It initializes Firebase Admin SDK using a service account key, establishing a connection to the Firebase database and storage services for storing student data and images.

Using OpenCV, the script captures video from the webcam and sets its dimensions for processing.

Background images and mode images are loaded from the local file system. These images are used for visualizing attendance information and system modes.

Pre-computed face encodings and corresponding student IDs are loaded from a pickle file. These encodings are essential for recognizing students' faces during attendance tracking.

The main loop continuously captures frames from the webcam feed and processes them for facial recognition.

Upon detecting a face in the frame, the script compares it with known face encodings to identify the student. If a match is found, it fetches the corresponding student information from the Firebase database.

The script updates attendance records based on certain conditions, such as the time elapsed since the last attendance entry. If a student is detected after a certain time interval, their attendance is marked, and the timestamp is updated in the database.

Various modes of operation are supported, such as displaying loading screens, student information, and images, based on the current state of the system.

The script continuously updates the display with attendance information and overlays using OpenCV's imshow function.

Overall, Main.py represents a sophisticated implementation of a facial recognition-based attendance system, seamlessly integrating with Firebase for efficient data management and providing a user-friendly interface for attendance tracking in educational environments.

Main.py

```
import os
import pickle
import numpy as np
import cv2
import face_recognition
import cvzone
import firebase_admin
```

```
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import numpy as np
from datetime import datetime

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "",
    'storageBucket': ""
})

bucket = storage.bucket()

cap = cv2.VideoCapture(1)
cap.set(3, 640)
cap.set(4, 480)

imgBackground = cv2.imread('Resources/background.png')

# Importing the mode images into a list
folderModePath = 'Resources/Modes'
modePathList = os.listdir(folderModePath)
imgModeList = []
for path in modePathList:
    imgModeList.append(cv2.imread(os.path.join(folderModePath, path)))
# print(len(imgModeList))

# Load the encoding file
```

```
print("Loading Encode File ...")
file = open('EncodeFile.p', 'rb')
encodeListKnownWithIds = pickle.load(file)
file.close()
encodeListKnown, studentIds = encodeListKnownWithIds
# print(studentIds)
print("Encode File Loaded")

modeType = 0
counter = 0
id = -1
imgStudent = []

while True:
    success, img = cap.read()

    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    faceCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)

    imgBackground[162:162 + 480, 55:55 + 640] = img
    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

    if faceCurFrame:
        for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown,
encodeFace)
```

```
            faceDis = face_recognition.face_distance(encodeListKnown,
encodeFace)
            # print("matches", matches)
            # print("faceDis", faceDis)

            matchIndex = np.argmin(faceDis)
            # print("Match Index", matchIndex)

            if matches[matchIndex]:
                # print("Known Face Detected")
                # print(studentIds[matchIndex])
                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
                bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
                imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)
                id = studentIds[matchIndex]
                if counter == 0:
                    cvzone.putTextRect(imgBackground, "Loading", (275, 400))
                    cv2.imshow("Face Attendance", imgBackground)
                    cv2.waitKey(1)
                    counter = 1
                    modeType = 1

        if counter != 0:

            if counter == 1:
                # Get the Data
                studentInfo = db.reference(f'Students/{id}').get()
                print(studentInfo)
```

```
            # Get the Image from the storage
            blob = bucket.get_blob(f'Images/{id}.png')
            array = np.frombuffer(blob.download_as_string(), np.uint8)
            imgStudent = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
            # Update data of attendance
            datetimeObject =
datetime.strptime(studentInfo['last_attendance_time'],
                                      "%Y-%m-%d %H:%M:%S")
            secondsElapsed = (datetime.now() - datetimeObject).total_seconds()
            print(secondsElapsed)
            if secondsElapsed > 30:
                ref = db.reference(f'Students/{id}')
                studentInfo['total_attendance'] += 1
                ref.child('total_attendance').set(studentInfo['total_attendance'])
                ref.child('last_attendance_time').set(datetime.now().strftime("%Y-
%m-%d %H:%M:%S"))
            else:
                modeType = 3
                counter = 0
                imgBackground[44:44 + 633, 808:808 + 414] =
imgModeList[modeType]

        if modeType != 3:

            if 10 < counter < 20:
                modeType = 2

            imgBackground[44:44 + 633, 808:808 + 414] =
imgModeList[modeType]
```

```
            if counter <= 10:
                cv2.putText(imgBackground, str(studentInfo['total_attendance']),
(861, 125),
                            cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 1)
                cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),
                            cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255),
1)
                cv2.putText(imgBackground, str(id), (1006, 493),
                            cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255),
1)
                cv2.putText(imgBackground, str(studentInfo['standing']), (910,
625),
                            cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100),
1)
                cv2.putText(imgBackground, str(studentInfo['year']), (1025, 625),
                            cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100),
1)
                cv2.putText(imgBackground, str(studentInfo['starting_year']),
(1125, 625),
                            cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100),
1)

                (w, h), _ = cv2.getTextSize(studentInfo['name'],
cv2.FONT_HERSHEY_COMPLEX, 1, 1)
                offset = (414 - w) // 2
                cv2.putText(imgBackground, str(studentInfo['name']), (808 +
offset, 445),
                            cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 50), 1)
```

```
                imgBackground[175:175 + 216, 909:909 + 216] = imgStudent

            counter += 1

            if counter >= 20:
                counter = 0
                modeType = 0
                studentInfo = []
                imgStudent = []
                imgBackground[44:44 + 633, 808:808 + 414] =
imgModeList[modeType]
    else:
        modeType = 0
        counter = 0
    # cv2.imshow("Webcam", img)
    cv2.imshow("Face Attendance", imgBackground)
    cv2.waitKey(1)
```

EncodeGenerator.py is a Python script designed to generate and store facial encodings for a collection of student images. Below is a breakdown of its functionality:

Firebase Initialization: The script initializes the Firebase Admin SDK using a service account key, establishing connectivity to the Firebase database and storage services.

Importing Student Images: It retrieves student images from a specified folder (Images). Each image is read using OpenCV and added to a list (imgList). Additionally, it extracts student IDs from the filenames and stores them in a separate list (studentIds).

Uploading Images to Firebase Storage: For each image, the script uploads it to Firebase Storage using the corresponding student ID as the filename. This ensures that the images are securely stored in the cloud.

Face Encoding Generation: The script defines a function findEncodings() to generate facial encodings for the images using the face_recognition library. It iterates through the list of images, converts each image to RGB format (as required by the library), and computes the facial encoding using face_recognition.face_encodings(). The resulting encodings are stored in a list (encodeList).

Encoding Process: The script initiates the encoding process by calling the findEncodings() function with the list of student images. It then combines the list of encodings (encodeList) with the list of student IDs (studentIds) into a single list (encodeListKnownWithIds).

Saving Encodings: The combined list (encodeListKnownWithIds) is saved to a binary file (EncodeFile.p) using the pickle module. This file serves as a serialized representation of the facial encodings and corresponding student IDs.

Overall, EncodeGenerator.py automates the process of generating facial encodings for student images, uploading them to Firebase Storage, and saving the encoded data for future use in facial recognition-based applications, such as attendance tracking.

EncodeGenerator.py

```
import cv2
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import  storage

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "",
    'storageBucket': ""
})



# Importing student images
folderPath = 'Images'
```

```
pathList = os.listdir(folderPath)
print(pathList)
imgList = []
studentIds = []
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath, path)))
    studentIds.append(os.path.splitext(path)[0])

    fileName = f'{folderPath}/{path}'
    bucket = storage.bucket()
    blob = bucket.blob(fileName)
    blob.upload_from_filename(fileName)



    # print(path)
    # print(os.path.splitext(path)[0])
print(studentIds)



def findEncodings(imagesList):
    encodeList = []
    for img in imagesList:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)

    return encodeList
```

```
print("Encoding Started ...")
encodeListKnown = findEncodings(imgList)
encodeListKnownWithIds = [encodeListKnown, studentIds]
print("Encoding Complete")

file = open("EncodeFile.p", 'wb')
pickle.dump(encodeListKnownWithIds, file)
file.close()
print("File Saved")
```

AddDatatoDatabase.py is a Python script responsible for adding data to a Firebase Realtime Database. Here's a breakdown of its functionality:

Firebase Initialization: The script initializes the Firebase Admin SDK using a service account key, establishing connectivity to the Firebase Realtime Database.

Defining Database Reference: It defines a reference to the 'Students' node in the Firebase database. This reference will be used to access and manipulate data within that node.

Defining Data: The script defines a dictionary named data, where each key represents a unique student ID and its corresponding value is a nested dictionary containing various student attributes such as name, major, starting year, total attendance, standing, year, and last attendance time.

Adding Data to Database: It iterates over each key-value pair in the data dictionary and sets the corresponding child node under the 'Students' node in the Firebase database. This effectively adds the student data to the database.

Overall, AddDatatoDatabase.py automates the process of populating the Firebase Realtime Database with student information, facilitating efficient data management and retrieval for applications such as attendance tracking systems.

AddDatatoDatabase.py

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db

cred = credentials.Certificate("serviceAccountKey.json")
```

```
firebase_admin.initialize_app(cred, {
    'databaseURL': ""
})

ref = db.reference('Students')

data = {
    "321654":
        {
            "name": "Murtaza Hassan",
            "major": "Robotics",
            "starting_year": 2017,
            "total_attendance": 7,
            "standing": "G",
            "year": 4,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
    "852741":
        {
            "name": "Emly Blunt",
            "major": "Economics",
            "starting_year": 2021,
            "total_attendance": 12,
            "standing": "B",
            "year": 1,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
    "963852":
        {
```

```
            "name": "Elon Musk",
            "major": "Physics",
            "starting_year": 2020,
            "total_attendance": 7,
            "standing": "G",
            "year": 2,
            "last_attendance_time": "2022-12-11 00:54:34"
        }
}

for key, value in data.items():
    ref.child(key).set(value)
```

References

[1] Yazdani, Hamid Reza, and Ali Reza Shojaeifard. ”Facial recognition system using eigenfaces and PCA.” Mathematics and Computational Sciences 4.1 (2023): 29-35.

[2] Najafi Khanbebin, Saeed, and Vahid Mehrdad. ”Local improvement approach and linear discriminant analysis-based local binary pattern for face recognition.” Neural Computing and Applications 33 (2021): 7691- 7707.

[3] Kas, Mohamed, et al. ”A comprehensive comparative study of handcrafted methods for face recognition LBP-like and non LBP operators.” Multimedia Tools and Applications 79 (2020): 375-413. [4] Rehman, Sami Ul, et al. ”DRA-Net: densely residual attention based low-light image enhancement.” Fourteenth International Conference on Graphics and Image Processing (ICGIP 2022). Vol. 12705. SPIE, 2023.

[5] Serengil, Sefik Ilkin, and Alper Ozpinar. ”Lightface: A hybrid deep face recognition framework.” 2020 innovations in intelligent systems and applications conference (ASYU). IEEE, 2020.

[6] Kezebou, Landry, et al. ”TR-GAN: Thermal to RGB face synthesis with generative adversarial network for cross-modal face recognition.” Mobile Multimedia/Image Processing, Security, and Applications 2020. Vol. 11399. SPIE, 2020.

[7] Martinez-Diaz, Yoanna, et al. ”Benchmarking lightweight face architectures on specific face recognition scenarios.” Artificial Intelligence Review (2021): 1-44. Shiddiqi, Ary Mazharuddin, Edo Dwi Yogatama, and Dini Adni Navastara. ”Resource-aware

[8] video streaming (RAViS) framework for object detection system using deep learning algorithm.” MethodsX 11 (2023): 102285.

[9] Wang, Xiaofei, et al. ”Convergence of edge computing and deep learning: A comprehensive survey.” IEEE Communications Surveys & Tutorials 22.2 (2020): 869-904.

[10] Masi, I., Tran, A. T., Hassner, T., Leksut, J. T., & Medioni, G. (2016). Do We Need to Collect Millions of Faces for Effective Face Recognition? European Conference on Computer Vision (ECCV).

[11] M. Arsenovic, S. Sladojevic, A. Anderla and D. Stefanovic, ”FaceTime — Deep learning based face recognition attendance system,” 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2017, pp. 000053-000058, doi: 10.1109/SISY.2017.8080587.

[12] B. P. Prayogo, Hendrawan, E. Mulyana and W. Hermawan, ”A Novel Approach for Face Recognition: YOLO-Based Face Detection and Facenet,” 2023 9th International Conference on Wireless and Telematics (ICWT), Solo, Indonesia, 2023, pp. 1-6, doi: 10.1109/ICWT58823.2023.10335263.

[13] S. Khalili and A. Shakiba, ”A face detection method via ensemble of four versions of YOLOs,” 2022 International Conference on Machine Vision and Image Processing (MVIP), Ahvaz, Iran, Islamic Republic of, 2022, pp. 1-4, doi: 10.1109/MVIP53647.2022.9738779.

[14] V. A, R. R. Krishna and R. V. U, ”Facial Recognition System for Automatic Attendance Tracking Using an Ensemble of Deep-Learning Techniques,” 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9580098.

[15] V. Bittal, V. Jagdale, A. Brahme, D. Deore and B. Shinde, ”Multifarious Face Attendance System using Machine Learning and Deep Learning,” 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 387-392, doi: 10.1109/ICICCS56967.2023.10142759.

[16] M. Suman Menon, A. George and N. Aswathy, ”Implementation of a Multitudinous Face Recognition using YOLO.V3,” 2021 Fourth International Conference on Microelectronics, Signals & Systems (ICMSS), Kollam, India, 2021, pp. 1-6, doi: 10.1109/ICMSS53060.2021.967360

Publications

[1] K. Painuly, Y. Bisht, H. Vaidya, A. Kapruwan and R. Gupta, "Efficient Real-Time Face Recognition-Based Attendance System with Deep Learning Algorithms," 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bangalore, India, 2024, pp. 1-5, doi: 10.1109/IITCEE59897.2024.10467743. keywords: {YOLO;Deep learning;Training;Electric potential;Face recognition;Employment;Real-time systems;CNNs;YOLO;Computer Vision;Deep learning},