GOVERNMENT POLYTECHNIC COLLEGE PERUMBAVOOR

Koovappady P.O Ernakulam-683 544 Kerala

Semester - VI

Computer Engineering 2022-23

A SEMINAR REPORT

on

FACIAL RECOGNITION BASED ATTENDANCE SYSTEM

Submitted by

ASLAM SUNEER

BIJU PETER	**GLAXY GEORGE**
HOD	**Lecturer in charge**

Internal Examiner	**External Examiner**

ABSTRACT

The Face Recognition Attendance System using Python is a project aimed at automating the attendance marking process in various organizations and institutions. This system leverages the power of computer vision and machine learning techniques to recognize and verify individuals based on their facial features. The objective of this project is to provide an efficient and accurate solution for attendance management, eliminating the need for manual processes and reducing human error.The system utilizes state-of-the-art face recognition algorithms and libraries in Python, such as OpenCV,facerecognition library and dlib, to detect and extract facial landmarks from images or video streams. These landmarks are then used to create a unique facial signature for each individual, which serves as their biometric identifier. To train the recognition model, a dataset of labeled images containing faces of registered individuals is used, allowing the system to learn and differentiate between different individuals.

In the attendance marking process, the system captures real-time video or images using a webcam or other suitable camera device. The captured frames are processed by the face recognition model to identify and verify the individuals present in the scene. Upon successful identification, the system records the individual's name and the timestamp of their attendance. The attendance records are stored in a CSV (Comma-Separated Values) file, ensuring easy integration with existing attendance management systems or further analysis. The implementation of the Face Recognition Attendance System provides numerous advantages over traditional attendance methods. It offers increased accuracy, as it relies on unique facial features that are difficult to forge or manipulate. Additionally, the system saves time and effort by automating the attendance marking process, enabling quick and efficient tracking of attendance data.

Acknowledgments

I am taking this opportunity to thank all who supported me with their valuable advice and co- operation throughout the entire project work.
I would like to express my profound thanks to our principal Dr.Aiju Thomas for having provided us with sufficient facilities at our college. I would like to express my profound gratitude to head of department Mr.Biju Peter for encouragement, support and blessing given always to me.
Thanks my parents for their kind co-operation and encouragement which help me in completion of this project. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time. My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Contents

1 INTRODUCTION **4**

1.1 Objectives . . . 4

1.2 Background . . . 5

1.3 Problem Statement . . . 6

1.4 Aims and Objectives . . . 7

1.5 Project Scope . . . 7

2 Requirement and Analysis **9**

2.1 System Feasibility . . . 10

2.1.1 Economic Feasibility . . . 10

2.1.2 Technical Feasibility . . . 10

3 Technologies Used **11**

3.1 Python . . . 11

3.1.1 OpenCV . . . 13

3.1.2 Numpy . . . 15

3.1.3 OS . . . 17

3.1.4 Concurrent.futures . . . 18

3.1.5 Facerecognition library . . . 20

3.1.6 DateTime . . . 22

3.2 PyCharm . . . 23

3.3 Dlib library . . . 25

4 Languages Used **28**

5 Flow Diagram **29**

6 Code **30**

6.1 Code Explanation . . . 32

6.2 Algorithm used 34
6.2.1 CNN(convolutional neural network) 34
6.2.2 Image Representation 34

7 Output **36**

8 CONCLUSION **38**

9 REFERENCES **39**

Chapter 1

INTRODUCTION

The face recognition based attendance system using Python and Google Sheets is a powerful tool that combines the capabilities of computer vision and cloud-based storage to streamline attendance tracking. By utilizing face recognition algorithms, the system can accurately identify individuals from images or video footage, and then record their attendance in a Google Sheets spreadsheet.
This system offers several advantages over traditional attendance methods. It eliminates the need for manual sign-in processes, reduces errors and fraudulent activities, and provides real-time attendance updates. Additionally, by leveraging Google Sheets as the backend storage, attendance data becomes easily accessible and can be further analyzed or integrated with other systems.

In this project, we will leverage Python libraries such as OpenCV and face recognition to perform face detection and recognition. The identified faces will be matched against a pre-trained model or a database of known individuals. Once the individual is recognized, the attendance will be logged in a Google Sheets spreadsheet using the Google Sheets API.By combining the power of face recognition and cloud-based storage, this system offers a robust and efficient solution for managing attendance in various settings, such as educational institutions, workplaces, or events.

1.1 Objectives

Attendance is prime important for both the teacher and student of an educational organization. So it is very important to keep record of the attendance. The problem arises when we think about the traditional process of taking attendance in class room.Calling name or roll number of the student for attendance is not only a problem of time consumption but also it needs energy. So an automatic attendance system

can solve all above problems.There are some automatic attendances making system which are currently used by much institution. One of such system is biometric technique and RFID system. Although it is automatic and a step ahead of traditional method it fails to meet the time constraint. The student has to wait in queue for giving attendance, which is time taking.
This project introduces an involuntary attendance marking system, devoid of any kind of interference with the normal teaching procedure. The system can be also implemented during exam sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions. In addition, the students have to register in the database to be recognized. The enrolment can be done on the spot through the userfriendly interface.

1.2 Background

Face recognition is crucial in daily life in order to identify family, friends or someone we are familiar with. We might not perceive that several steps have actually taken in order to identify human faces. Human intelligence allows us to receive information and interpret the information in the recognition process. We receive information through the image projected into our eyes, by specifically retina in the form of light. Light is a form of electromagnetic waves which are radiated from a source onto an object and projected to human vision. Robinson-Riegler, G., Robinson-Riegler, B. (2008) mentioned that after visual processing done by the human visual system, we actually classify shape, size, contour and the texture of the object in order to analyze the information. The analyzed information will be compared to other representations of objects or face that exist in our memory to recognize. In fact, it is a hard challenge to build an automated system to have the same capability as a human to recognize faces. However, we need large memory to recognize different faces, for example, in the Universities, there are a lot of students with different race and gender, it is impossible to remember every face of the individual without making mistakes. In order to overcome human limitations, computers with almost limitless memory, high processing speed and power are used in face recognition systems.

The human face is a unique representation of individual identity. Thus, face recognition is defined as a biometric method in which identification of an individual is performed by comparing real-time capture image with stored images in the database of that person (Margaret Rouse, 2012).
Nowadays, face recognition system is prevalent due to its simplicity and awesome performance. For instance, airport protection systems and FBI use face recognition for criminal investigations by tracking suspects, missing children and drug activities (Robert Silk, 2017). Apart from that, Facebook which is a popular social networking website implement face recognition to allow the users to tag their friends in the photo for entertainment purposes (Sidney Fussell, 2018). Furthermore, Intel Company allows the users to use face recognition to get access to their online account (Reichert, C., 2017). Apple allows the users to unlock their mobile phone, iPhone X by using face recognition (deAgonia, M., 2017).
The work on face recognition began in 1960. Woody Bledsoe, Helen Chan Wolf and Charles Bisson had introduced a system which required the administrator to locate eyes, ears, nose and mouth from images. The distance and ratios between the located features and the common reference points are then calculated and compared. The studies are further enhanced by Goldstein, Harmon, and Lesk in 1970 by using other features such as hair colour and lip thickness to automate the recognition. In 1988, Kirby and Sirovich first suggested principle component analysis (PCA) to solve face recognition problem. Many studies on face recognition were then conducted continuously until today (Ashley DuVal, 2012).

1.3 Problem Statement

Traditional student attendance marking technique is often facing a lot of trouble. The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking technique such as calling student names or checking respective identification cards. There are not only disturbing the teaching process but also causes distraction for students during exam sessions. Apart from calling names, attendance sheet is passed around the classroom during the lecture sessions. The lecture class especially the class with a large number of students might find it difficult to have the attendance sheet being passed around the class.

Thus, face recognition attendance system is proposed in order to replace the manual signing of the presence of students which are burdensome and causes students get distracted in order to sign for their attendance. Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

The paper proposed by Zhao, W et al. (2003) has listed the difficulties of facial identification. One of the difficulties of facial identification is the identification between known and unknown images. In addition, paper proposed by Pooja G.R et al. (2010) found out that the training process for face recognition student attendance system is slow and time-consuming. In addition, the paper proposed by Priyanka Wagh et al. (2015) mentioned that different lighting and head poses are often the problems that could degrade the performance of face recognition based student attendance system.

Hence, there is a need to develop a real time operating student attendance system which means the identification process must be done within defined time constraints to prevent omission. The extracted features from facial images which represent the identity of the students have to be consistent towards a change in background, illumination, pose and expression. High accuracy and fast computation time will be the evaluation points of the performance.

1.4 Aims and Objectives

The objective of this project is to develop face recognition attendance system.
Expected achievements in order to fulfill the objectives are:
i. To detect the face segment from the video frame.
ii. To extract the useful features from the face detected.
iii. To classify the features in order to recognize the face detected.
iv. To record the attendance of the identified student in Google sheet.

1.5 Project Scope

The scope of the project involves the development of a Face Recognition Attendance System with two modules. The first module focuses on capturing and storing face images, while the second module involves marking the detected faces in a Google Sheet along with the student's name and the time of detection. Here's a breakdown of each module's scope:

1. First Module: Face Image Capture and Storage

- Utilize the laptop camera to capture face images.
- Implement face detection and recognition algorithms to identify faces in real-time.
- Compress and store the captured face images in a file named "training images."
- Associate each face image with the respective student's name as a label.

2. Second Module: Attendance Marking in Google Sheet

- Use the face recognition algorithm to detect faces from the captured images or real-time video stream.
- Compare the detected faces with the stored face images to identify students.
- Retrieve the student's name corresponding to the recognized face.
- Record the student's name and the timestamp of face detection in a Google Sheet.
- Update the Google Sheet with each new face detection, adding new rows for attendance records.

Chapter 2

Requirement and Analysis

The requirement and analysis for a face recognition-based attendance system using Python with Google Sheets involves several key aspects. The goal of the system is to provide an accurate and efficient way to track attendance using face recognition technology while integrating with Google Sheets for data storage and management. To begin with, the system needs to incorporate reliable face detection and recognition algorithms. This involves selecting and implementing appropriate techniques such as Haar cascades or deep learning-based models to detect and identify faces from images or video footage. The accuracy and efficiency of the chosen algorithms are crucial for ensuring reliable attendance tracking.

To achieve accurate face recognition, a robust training dataset is required. This dataset should include a diverse set of images or video samples for each individual whose attendance will be tracked. The dataset should cover different lighting conditions, angles, and facial expressions to improve the system's ability to recognize individuals accurately.The system should also include pre-processing and image enhancement techniques. These techniques help optimize the input images and improve recognition accuracy. Operations such as image resizing, normalization, and enhancement can be applied to ensure consistent and high-quality input for the face recognition algorithm.

The attendance logging functionality is an essential component of the system. Once an individual's face is recognized, their attendance should be logged by recording relevant information such as their name, date, and time in the appropriate columns of the Google Sheets spreadsheet. This process should be automated and real-time to provide up-to-date attendance records.

Scalability and performance are important factors for a system that may need to handle a large number of individuals and attendance records. Optimizations such

as parallel processing or distributed computing can be explored to enhance performance when dealing with significant data volumes.
Comprehensive documentation should be provided to guide users in setting up, configuring, and using the system effectively. Additionally, a support mechanism should be available to address user queries and troubleshoot any issues that may arise during implementation and usage.

2.1 System Feasibility

The system feasibility can be divided into the following sections:

2.1.1 Economic Feasibility

The project is economically feasible as the only cost involved is having a computer with the minimum requirements are dlib library,facerecoginition library and mainly opencv. Python is the language used.

2.1.2 Technical Feasibility

OperatingEnvironments: Windows10,ExcelSheet,PyCharm,AnacondaPrompt,cmake

Chapter 3

Technologies Used

3.1 Python

Python is a powerful programming language widely used in various domains, including computer vision and data management, making it an excellent choice for developing a face recognition-based attendance system integrated with Google Sheets.
Python provides a vast ecosystem of libraries and frameworks that simplify the implementation of complex tasks. For face recognition, libraries such as OpenCV and face recognition offer robust algorithms and methods to detect and recognize faces from images or video footage. These libraries provide pre-trained models or the ability to train custom models on specific datasets, enabling accurate face identification.Additionally, Python's simplicity and readability make it easy to understand and maintain the codebase. It offers a wide range of data manipulation and analysis libraries like NumPy and Pandas, which can be leveraged for data preprocessing, handling attendance records, and interacting with Google Sheets.
Python's versatility extends to its ability to interface with Google Sheets. The Google Sheets API provides a straightforward way to interact with Google Sheets spreadsheets, allowing attendance data to be recorded and retrieved seamlessly. Libraries such as gspread simplify the integration process by providing a Pythonic interface to access and modify Google Sheets data.Python's rich ecosystem also offers various supporting libraries that enhance the functionality of the attendance system. For example, Flask or Django can be used to develop a web application for the system's user interface, allowing users to interact with the system through a web browser. These frameworks provide robust routing, template rendering, and authentication features to create a user-friendly experience.Furthermore, Python's extensive community support and documentation make it easier to find solutions to common problems or address specific challenges during the development process.

Online resources, forums, and tutorials are readily available, enabling developers to leverage existing knowledge and collaborate with others in the Python community. Python's versatility, extensive libraries, ease of use, and integration capabilities with Google Sheets make it an excellent choice for developing a face recognition-based attendance system. Its robustness, community support, and wide range of tools contribute to a smooth development process and the creation of a reliable and efficient solution.

Uses of Python:
Python is extensively used in face recognition applications due to its rich ecosystem of libraries and tools. Here are some of the key uses of Python in face recognition:
1. Python provides libraries like OpenCV and dlib, which offer pre-trained models and algorithms for face detection. These libraries use computer vision techniques to locate and extract facial regions from images or video frames.
2. Python's libraries, such as OpenCV and face recognition, enable accurate facerecognition by leveraging deep learning algorithms. These libraries provide pre-trained models that can identify individuals by comparing facial features or embeddings.
3. Python's data manipulation libraries, such as NumPy and Pandas, are used for preprocessing face images. This involves tasks like resizing, cropping, normalizing, and enhancing the images to improve the accuracy of face recognition algorithms.
4. Python's machine learning libraries, such as TensorFlow and PyTorch, are used to train custom face recognition models. These libraries allow developers to build, train, and fine-tune deep learning networks using large datasets of labeled face images.
5. Python's versatility allows it to integrate with other technologies and systems. For instance, Python can be used to connect face recognition systems with external databases, APIs, or cloud services. Integration with platforms like Google Sheets can facilitate attendance tracking and data management.
6. Python's extensive libraries, ease of use, and flexibility make it an ideal choice for developing face recognition applications. Its ecosystem supports various tasks ranging from face detection and recognition to data preprocessing, model training, user interface development, integration with external systems, and deployment of scalable systems.

3.1.1 OpenCV

OpenCV (Open Source Computer Vision) is a powerful computer vision library that provides a comprehensive set of tools and algorithms for image and video processing in Python. It is widely used by researchers, developers, and enthusiasts for a variety of computer vision applications.With OpenCV in Python, developers can leverage its rich functionality to perform tasks such as image manipulation, filtering, and enhancement. The library offers numerous functions for resizing, cropping, rotating, and adjusting the brightness and contrast of images. It also provides advanced features like morphological operations, edge detection, and contour extraction.
OpenCV's capabilities extend to object detection and tracking, which are crucial in many computer vision applications. Developers can utilize pre-trained models or train custom models to detect objects in images or video streams. OpenCV's algorithms, such as Haar cascades and HOG (Histogram of Oriented Gradients), enable the identification and tracking of objects based on their visual characteristics.Additionally, OpenCV in Python supports feature extraction and matching, making it useful for tasks like image recognition and image-based search. Developers can use algorithms like SIFT (Scale-Invariant Feature Transform) and ORB (Oriented FAST and Rotated BRIEF) to identify key points in images and match them across multiple images or frames.
OpenCV's integration with Python offers numerous advantages. Python's simplicity and readability make it easy to write and understand code for computer vision tasks. Python also provides access to a wide range of libraries and frameworks that can complement OpenCV, such as NumPy for numerical computations and Matplotlib for data visualization.Furthermore, Python's extensive community support ensures that developers can find resources, tutorials, and examples to assist them in utilizing OpenCV effectively. The availability of online forums and documentation facilitates collaboration and knowledge sharing among developers working on computer vision projects. OpenCV in Python is a versatile and powerful tool for computer vision applications. Its extensive range of functions, coupled with the simplicity and flexibility of Python, enables developers to build sophisticated image and video processing systems with case.

Features of OpenCV:
1. OpenCV offers a comprehensive set of functions for image manipulation, filtering, and enhancement. It includes operations like resizing, cropping, rotating, flipping, and adjusting brightness, contrast, and color levels. It also supports advanced techniques such as morphological operations, edge detection, and contour extraction.
2. OpenCV provides functionalities for reading, writing, and processing video streams. It allows developers to extract frames, perform real-time video analysis, apply filters and effects, and encode and decode videos using different codecs.
3.OpenCV includes various algorithms and techniques for object detection and tracking. It supports popular methods like Haar cascades, HOG (Histogram of Oriented Gradients), and deep learning-based approaches. These algorithms enable the detection and tracking of objects in images and video streams.
4. OpenCV provides tools for camera calibration, which is essential for accurate 3D reconstruction and computer vision applications. It includes functions for camera calibration, pose estimation, stereo vision, and depth estimation using stereo cameras or depth sensors.

Disadvantages:
1. OpenCV has a complex API and can be challenging for beginners to grasp. Understanding and effectively utilizing its numerous functions and algorithms requires a solid understanding of computer vision concepts and programming skills.
2. OpenCV's documentation can be incomplete or outdated for certain functions and algorithms. This can make it difficult for developers to find accurate and up-to-date information, leading to potential issues or challenges when working with specific features of the library.
3. While OpenCV provides optimized implementations of many algorithms, it may not always achieve the best performance on certain hardware platforms or in specific scenarios. Fine-tuning and optimizing OpenCV code for specific use cases and hardware configurations may require additional effort and expertise.
4. Although OpenCV integrates with deep learning frameworks like TensorFlow and PyTorch, its deep learning capabilities are relatively limited compared to specialized deep learning libraries. OpenCV's deep learning support is not as extensive or

flexible as frameworks dedicated solely to deep learning tasks.

3.1.2 Numpy

NumPy, short for Numerical Python, is a fundamental Python library that plays a critical role in numerical computing. It provides powerful tools and data structures for efficient manipulation and computation with large, multi-dimensional arrays and matrices. The centerpiece of NumPy is the ndarray (n-dimensional array) object, which allows for efficient storage and operations on homogeneous data.
NumPy offers an extensive collection of mathematical functions that operate element-wise on arrays. These functions encompass a wide range of operations, including basic arithmetic operations, statistical functions, linear algebra routines, trigonometric functions, and more. The ability to apply these operations directly to entire arrays, without the need for explicit looping, makes NumPy highly efficient and convenient for numerical computations.One of NumPy's notable features is broadcasting, which enables operations between arrays with different shapes and dimensions. It automatically aligns the dimensions of the arrays, allowing for element-wise operations to be performed smoothly. This simplifies the code and eliminates the need for explicit looping or resizing of arrays.
NumPy also provides advanced indexing and slicing techniques, allowing for selective access and manipulation of elements or subsets of an array. This feature enables efficient extraction and modification of data, enhancing data processing capabilities.In addition to its core functionalities, NumPy seamlessly integrates with other Python libraries, such as SciPy and Matplotlib, forming a powerful ecosystem for scientific computing and data analysis. The combination of these libraries provides a comprehensive suite of tools for various numerical computing tasks.
NumPy is known for its efficient numerical computations, thanks to its C-based implementation and optimized functions. It leverages low-level libraries and numerical routines, resulting in faster execution times compared to pure Python code. NumPy is widely used in scientific and numerical computing domains, such as data analysis, machine learning, image processing, simulation, and optimization. Its array-based computing, extensive mathematical functions, and efficient memory utilization make it an essential tool for performing complex numerical computations in Python.

Features of Numpy:

1. N-dimensional Array Objects: NumPy introduces the ndarray (n-dimensional array) data structure, which allows efficient storage and manipulation of homogeneous data. It enables operations on arrays of various dimensions, from simple 1D arrays to complex multidimensional arrays.
2. Array Operations: NumPy provides a wide range of mathematical and logical operations that can be applied element-wise on arrays. These operations include arithmetic operations, statistical functions, linear algebra operations, trigonometric functions, and more. The ability to perform these operations on entire arrays efficiently is one of the primary advantages of using NumPy.
3. Broadcasting: NumPy's broadcasting feature enables operations between arrays with different shapes and dimensions. It automatically adjusts the dimensions of the arrays to match, eliminating the need for explicit looping and greatly simplifying computations.
4. Array Indexing and Slicing: NumPy supports advanced indexing and slicing techniques to access specific elements or subsets of an array. This allows for efficient extraction and manipulation of data from arrays.
5. Integration with Other Libraries: NumPy seamlessly integrates with other Python libraries, such as SciPy (Scientific Python) and Matplotlib (plotting library), forming a powerful ecosystem for scientific computing and data analysis.
6. Efficient Numerical Computing: NumPy is implemented in C and provides highly optimized functions, resulting in faster execution times compared to pure Python code. It leverages the capabilities of low-level libraries and numerical routines, making it suitable for high-performance computing tasks.
7. Memory Efficiency: NumPy's ndarray uses contiguous blocks of memory, resulting in efficient storage and access of large datasets. It also allows for more compact storage of data compared to standard Python lists.
8. Data Manipulation: NumPy offers a variety of functions for reshaping, sorting, filtering, and manipulating array data. These operations enable efficient data preprocessing and manipulation for tasks such as data cleaning, feature engineering, and data transformation.

3.1.3 OS

The "os" module in Python provides a way to interact with the operating system. It offers functions for performing various operating system-related tasks, such as file and directory operations, process management, environment variables, and more.

features of os:
1. File and Directory Operations: The "os" module provides functions for working with files and directories. You can create, delete, rename, and check the existence of files and directories. It allows for navigating directory structures, getting file attributes, and changing file permissions.
2. Process Management: The module includes functions for managing processes. You can launch new processes, terminate existing processes, get information about running processes, and handle process-related events. This allows for programmatic control and interaction with system processes.
3. Environment Variables: The "os" module provides functions to access and modify environment variables. You can retrieve values of environment variables, set new variables, and modify existing ones. This is useful for obtaining information about the system environment or configuring the runtime environment for your Python program.
4. Path Manipulation: The module offers functions for manipulating file paths. It allows for joining and splitting paths, extracting directory names and filenames, and checking the validity of paths. These functions simplify working with file and directory paths across different operating systems.
5. System Information: The "os" module provides functions to retrieve system-related information. You can get details such as the current working directory, the operating system name, the host name, and the user name. This information can be useful for system administration tasks or program behavior based on the underlying system.
6. File System Operations: The module includes functions for interacting with the file system. You can change the current working directory, create and remove directories recursively, traverse directory trees, and obtain file system-related information.

3.1.4 Concurrent.futures

The ”concurrent.futures” module in Python provides a high-level interface for asynchronously executing code and managing concurrent tasks. It offers a simple and efficient way to parallelize and distribute work across multiple threads or processes. The module includes two main classes: ”ThreadPoolExecutor” and ”ProcessPoolExecutor,” which implement the concurrent execution of tasks in a thread pool or a process pool, respectively.With ”concurrent.futures,” you can submit functions or callable objects for execution asynchronously. The module takes care of managing the worker threads or processes and returning the results once the tasks are completed. It provides a convenient abstraction that allows you to focus on the logic of your tasks rather than low-level thread or process management.
The ”ThreadPoolExecutor” class creates a pool of worker threads, which can execute tasks concurrently. You can submit tasks using the ”submit” method, which returns a ”Future” object representing the result of the computation. By using ”Future” objects, you can track the progress of the tasks, cancel them if needed, and retrieve their results when they are finished.Similarly, the ”ProcessPoolExecutor” class creates a pool of worker processes, allowing for parallel execution of tasks across multiple processes. It provides the same interface as the ”ThreadPoolExecutor” class, making it easy to switch between thread-based and process-based execution depending on the requirements of your code.
The ”concurrent.futures” module abstracts the complexities of managing threads or processes and provides a high-level interface for concurrent execution. It automatically handles thread or process creation, synchronization, and resource management, making it easier to write scalable and efficient concurrent code. It is particularly useful for tasks that are I/O-bound or can benefit from parallel processing, such as making network requests, performing data processing, or executing computationally intensive operations.By leveraging the ”concurrent.futures” module, you can take advantage of the available computing resources and maximize the utilization of your CPU cores or system threads. This can result in significant performance improvements and faster execution times for your applications.

Features of Concurrent.futures:
1. Asynchronous Execution: concurrent.futures enables asynchronous execution of

tasks, allowing for parallel processing and efficient utilization of system resources.
2. Executor Classes: The module provides two executor classes, ThreadPoolExecutor and ProcessPoolExecutor, which manage pools of worker threads or processes respectively. These executors handle the creation and management of worker units to execute tasks concurrently.
3. Future Objects: When submitting tasks for execution, the module returns Future objects that represent the results of the computations. Future objects allow you to track the progress of tasks, cancel them if needed, and retrieve their results when they are completed.
4. Task Submission and Execution: Tasks can be submitted for execution using the submit() method, which accepts callable objects or functions as input and returns a Future object representing the task. The tasks are executed concurrently by the worker threads or processes in the pool.
5. Exception Handling: concurrent.futures provides mechanisms to handle exceptions raised by the executed tasks. The exception() method of the Future object allows you to catch and handle exceptions appropriately.
6. Controlling Parallelism: The module offers control over the level of parallelism by allowing you to specify the maximum number of worker threads or processes. This flexibility allows you to adjust the concurrency level based on available system resources or the nature of the tasks being executed.
7. Synchronization and Ordering: concurrent.futures provides synchronization primitives such as Lock and Condition that can be used to coordinate tasks and ensure proper ordering of operations when necessary.
8. Integration with Iterables: The module includes map() and as completed() functions that simplify processing iterables of tasks. These functions return iterators that yield results as tasks are completed, making it easy to work with collections of tasks.
9. Graceful Shutdown: The module provides methods to gracefully shut down the executor and wait for all running tasks to complete. This ensures proper termination of worker threads or processes and prevents any potential resource leaks.
10. Context Managers: concurrent.futures supports context managers, allowing you to conveniently manage the lifecycle of executor instances and handle resource cleanup automatically.

3.1.5 Facerecognition library

The facerecognition library is a popular Python package that offers simplified face recognition capabilities. Built on top of the powerful dlib library, it provides a high-level interface for performing various face-related tasks. One of its key features is face detection, using dlib's highly accurate face detection algorithm. The library can identify and locate faces in images or video frames.In addition to face detection, the facerecognition library enables facial landmark detection, which involves identifying specific points on a face, such as the eyes, nose, and mouth. This information can be used for tasks like face alignment and emotion analysis.
One of the main functions of the facerecognition library is face encoding. It generates a numerical representation, or encoding, of a face based on its unique features. These encodings can be compared to determine if two faces belong to the same person or to identify a face against a known database of faces.The facerecognition library also provides convenient utilities for common face recognition tasks, such as face clustering (grouping similar faces together), facial attribute analysis (e.g., age and gender estimation), and face recognition in real-time video streams.With its easy-to-use interface and integration with dlib's robust algorithms, the facerecognition library has gained popularity among developers for its effectiveness in implementing face recognition applications. It offers a convenient solution for a range of use cases, including identity verification, access control, surveillance systems, and personalized experiences in various domains.

Features of Facerecognition library:
The facerecognition library provides a range of features for face recognition tasks in Python. Some of its notable features include:
1. Face Detection: The library utilizes dlib's face detection algorithm to locate faces in images or video frames. It can accurately detect and identify multiple faces within an image.
2. Facial Landmark Detection: It offers the capability to detect facial landmarks, which are specific points on a face, such as the eyes, nose, and mouth. This information can be used for tasks like face alignment, emotion analysis, and feature extraction.

3. Face Encoding: The library generates a numerical representation, or encoding, of a face based on its unique features. These encodings can be used to compare faces and determine if they belong to the same person. The encoding process captures distinctive facial features, allowing for efficient face matching.
4. Face Comparison: The library provides functions for comparing face encodings and determining the similarity between faces. This enables tasks such as face verification (matching a face against a known reference) and face identification (matching a face against a database of known faces).
5. Face Clustering: It includes utilities for clustering similar faces together based on their facial features. This allows for the organization and grouping of faces with similar attributes or identities.
6. Real-time Face Recognition: The library supports real-time face recognition in video streams, allowing for the identification and tracking of faces in real-time applications.
7. Facial Attribute Analysis: It provides functionality for estimating facial attributes such as age, gender, and facial expressions, enabling additional insights and analysis of detected faces.
8. Integration with Python: The facerecognition library is designed to be easy to use and integrates well with Python, making it accessible to a wide range of developers.

Facerecognition library is required:
The facerecognition library is required when you want to incorporate face recognition functionality into your Python project. Here are some key points highlighting its importance:
1. Face Detection: The library offers robust face detection capabilities, allowing you to locate and identify faces within images or video frames.
2. Facial Landmark Detection: It provides the ability to detect facial landmarks, enabling tasks such as face alignment, emotion analysis, and feature extraction.
3. Face Encoding: The library generates unique numerical representations (encodings) of faces based on their distinctive features, which can be used for face comparison and identification.
4. Face Comparison and Recognition: With facerecognition, you can compare face encodings to determine if two faces belong to the same person or identify faces against a known database.

5. Face Clustering: The library allows you to cluster similar faces together based on their facial features, facilitating organization and grouping of faces with similar attributes or identities.
6. Real-time Face Recognition: It supports real-time face recognition in video streams, making it useful for applications that require immediate face identification and tracking.
7. Facial Attribute Analysis: The library provides functionality for estimating facial attributes such as age, gender, and expressions, adding additional insights to face-related tasks.

3.1.6 DateTime

In Python, you can work with dates and times using the built-in datetime module. The datetime module provides classes for manipulating dates and times, including functions for formatting and parsing dates.
To get the current date and time, you can use the datetime class from the datetime module. Here's an example:

```
from datetime import datetime
currentdatetime = datetime.now()
print(currentdatetime)
```

This will output the current date and time in the default format, which is something like: 2023-05-22 14:30:45.123456.
If you want to format the date and time in a specific way, you can use the strftime method to format the datetime object into a string. Here's an example:

```
from datetime import datetime
currentdatetime = datetime.now()
formatteddatetime = currentdatetime.strftime(”print(formatteddatetime)
```

This will output the current date and time in the specified format: 2023-05-22 14:30:45.
You can also create specific date and time objects using the datetime class. Here's an example of creating a datetime object for a specific date and time:

```
from datetime import datetime
specificdatetime = datetime(2023, 5, 22, 14, 30, 0)
```

print(specificdatetime)

This will output the specified date and time: 2023-05-22 14:30:00.

These are some basic operations for working with dates and times in Python using the datetime module. There are additional functions and classes available in the module for more advanced operations, such as timedelta calculations and time zone conversions.

3.2 PyCharm

PyCharm is a powerful Integrated Development Environment (IDE) specifically tailored for Python development. Developed by JetBrains, PyCharm provides an extensive array of features and tools to enhance productivity and streamline the development process.At the core of PyCharm is its intelligent code editor, which offers syntax highlighting, code completion, and code navigation. It provides code analysis, error checking, and suggestions to help developers write clean and error-free code. The IDE supports various Python versions and offers a seamless coding experience with its user-friendly interface.

PyCharm excels in code refactoring and debugging capabilities. It offers a range of options for code restructuring and improving code quality, enabling developers to easily rename variables, extract code snippets into functions, and more. The advanced debugging tools allow for setting breakpoints, inspecting variables, and stepping through code to identify and fix issues efficiently.Managing Python projects is made easier with PyCharm's project management features. It enables the creation, organization, and navigation of project files and directories. The IDE integrates with version control systems such as Git, facilitating collaborative development and code management. It provides a smooth workflow for committing, pulling, and pushing changes to repositories.

PyCharm supports the creation and management of virtual environments, isolating project dependencies. It seamlessly integrates with popular package managers like pip and conda, making it effortless to install, update, and manage Python packages within projects.Testing is made convenient with PyCharm's support for various testing frameworks, including unittest, pytest, and doctest. It offers test runners,

code coverage analysis, and the ability to generate reports to ensure comprehensive testing and code quality.Web development is another area where PyCharm shines. It provides extensive support for web frameworks like Django, Flask, and Pyramid. Features include template editing, server configuration, and deployment tools, simplifying web application development.
PyCharm also offers built-in database tools for connecting to and working with databases. It supports popular database systems such as MySQL, PostgreSQL, and SQLite, allowing developers to interact with databases directly within the IDE. PyCharm can be extended and customized through plugins and integrations. It supports integration with tools like Jupyter Notebook, Docker, and Anaconda, expanding its functionality and adapting to specific project requirements., PyCharm provides a comprehensive and feature-rich environment for Python developers. Its intelligent code editor, debugging capabilities, project management tools, and integration with frameworks and tools make it a preferred choice for efficient and productive Python development.

Features of PyCharm:
1. Intelligent Code Editor: PyCharm provides a powerful code editor with features like syntax highlighting, code completion, and code navigation. It offers intelligent code analysis, error checking, and suggestions to help developers write clean and error-free code.
2. Code Refactoring and Debugging: PyCharm offers a range of refactoring options to improve code structure and maintainability. It also includes advanced debugging tools, allowing developers to set breakpoints, inspect variables, and step through code for efficient debugging.
3. Project Management: PyCharm supports managing Python projects of various sizes. It provides tools for creating, organizing, and navigating project files and directories. It also offers integration with version control systems like Git, allowing for seamless collaboration and code management.
4. Virtual Environments and Package Management: PyCharm facilitates the creation and management of virtual environments, which isolate project dependencies. It supports popular package managers like pip and conda, making it easy to install and manage Python packages within projects.
5. Testing and Test Coverage: The IDE includes features for writing and running

unit tests. PyCharm integrates with testing frameworks such as unittest, pytest, and doctest, providing test runners and generating coverage reports to ensure code quality.
6. Web Development Support: PyCharm extends its capabilities beyond Python with integrated support for web development frameworks like Django, Flask, and Pyramid. It provides tools for template editing, server configuration, and deployment to streamline web application development.
7. Database Support: PyCharm offers built-in support for working with databases. It includes database tools for connecting to, querying, and managing various databases, such as MySQL, PostgreSQL, and SQLite, directly within the IDE.

3.3 Dlib library

The dlib library is a powerful C++ library used for machine learning and computer vision tasks, offering a broad range of tools and algorithms for tasks such as object detection, facial recognition, image segmentation, and more.
The dlib library is a highly regarded C++ library renowned for its capabilities in machine learning and computer vision. With a wide range of tools and algorithms, dlib empowers developers to tackle various tasks in these fields with ease. Its machine learning capabilities include support for popular algorithms like support vector machines (SVM), k-nearest neighbors (k-NN), and deep learning tools such as convolutional neural networks (CNN). This allows for tasks like classification, regression, and clustering to be efficiently performed. In the realm of computer vision, dlib provides powerful features like image processing, feature extraction, object tracking, and geometric transformations. It includes implementations of algorithms like the Histogram of Oriented Gradients (HOG) and its own face detector, enabling tasks such as object detection and facial recognition. The library's versatility and comprehensive nature make it a valuable resource for developers aiming to build advanced applications in machine learning and computer vision.
The dlib library is widely known for its extensive capabilities in machine learning and computer vision, providing developers with a comprehensive set of tools and algorithms for tasks such as classification, regression, clustering, image processing, object tracking, and geometric transformations, making it a versatile and valuable

resource for building advanced applications in these domains.

Features of dlib library:
The dlib library offers a rich set of features for machine learning and computer vision tasks. Some of the notable features include:
1. Machine Learning Algorithms: dlib provides various machine learning algorithms, including support vector machines (SVM), k-nearest neighbors (k-NN), decision trees, and ensemble methods. These algorithms can be applied to classification, regression, and clustering problems.
2. Deep Learning Support: dlib supports deep learning through its implementation of convolutional neural networks (CNN). It allows developers to build and train CNN models for tasks such as image classification, object detection, and facial recognition.
3. Computer Vision Functionality: The library offers a comprehensive set of computer vision functionalities. It includes image processing operations like resizing, cropping, and filtering. It also provides tools for feature extraction, image alignment, object tracking, and geometric transformations.
4. Face Detection and Recognition: dlib incorporates robust face detection and recognition capabilities. It includes its own face detector based on the Histogram of Oriented Gradients (HOG) method. Additionally, dlib provides a face landmark detection model for facial feature localization, which can be used for tasks like face alignment and emotion analysis.
5. Tools for Image Segmentation: dlib includes algorithms for image segmentation, such as graph cuts and superpixels. These tools enable partitioning an image into meaningful regions, which is useful for tasks like object segmentation and scene understanding.
6. Optimization and Numerical Methods: The library offers optimization algorithms and numerical methods, including solvers for linear and quadratic programming problems, numerical integration, and matrix computations.
7. Cross-Platform Support: dlib is designed to be portable and works across different operating systems, including Windows, macOS, and Linux. It supports multiple programming languages, including C++ and Python, making it accessible to a wider range of developers.

How dlib Work?:

At a high level, dlib works by providing a set of pre-implemented algorithms and tools that can be utilized to perform machine learning and computer vision tasks. Here is an overview of how dlib typically works:

1. Data Representation: Before using dlib, data must be properly represented in a format compatible with the library. For example, images may need to be loaded and converted into appropriate data structures that dlib can process, such as matrices or image arrays.

2. Algorithm Selection and Configuration: Depending on the specific task at hand, developers choose the appropriate algorithm or tool from the dlib library. This may involve selecting a machine learning algorithm (e.g., SVM or k-NN) for classification, a CNN architecture for deep learning tasks, or a specific computer vision algorithm (e.g., face detection or image segmentation).

3. Training or Initialization: For certain tasks, such as training a machine learning model or initializing a deep learning network, dlib provides mechanisms to prepare the algorithm for the specific problem. This may involve providing labeled training data for supervised learning tasks or configuring the network architecture and hyperparameters for deep learning tasks.

4. Data Processing and Feature Extraction: Once the algorithm is set up, dlib allows developers to process data and extract relevant features. For example, in computer vision tasks, this may involve performing image preprocessing, applying filters, or extracting descriptors from images.

5. Algorithm Execution: With the data and features ready, developers can execute the selected algorithm using the provided functions or methods. dlib takes care of the underlying computations and optimizations to perform the desired task, such as classification, regression, clustering, object detection, or facial recognition.

6. Evaluation and Analysis: After the algorithm execution, developers can evaluate the results and analyze the performance of the applied algorithm. This may involve calculating metrics, visualizing outputs, or comparing against ground truth or validation data.

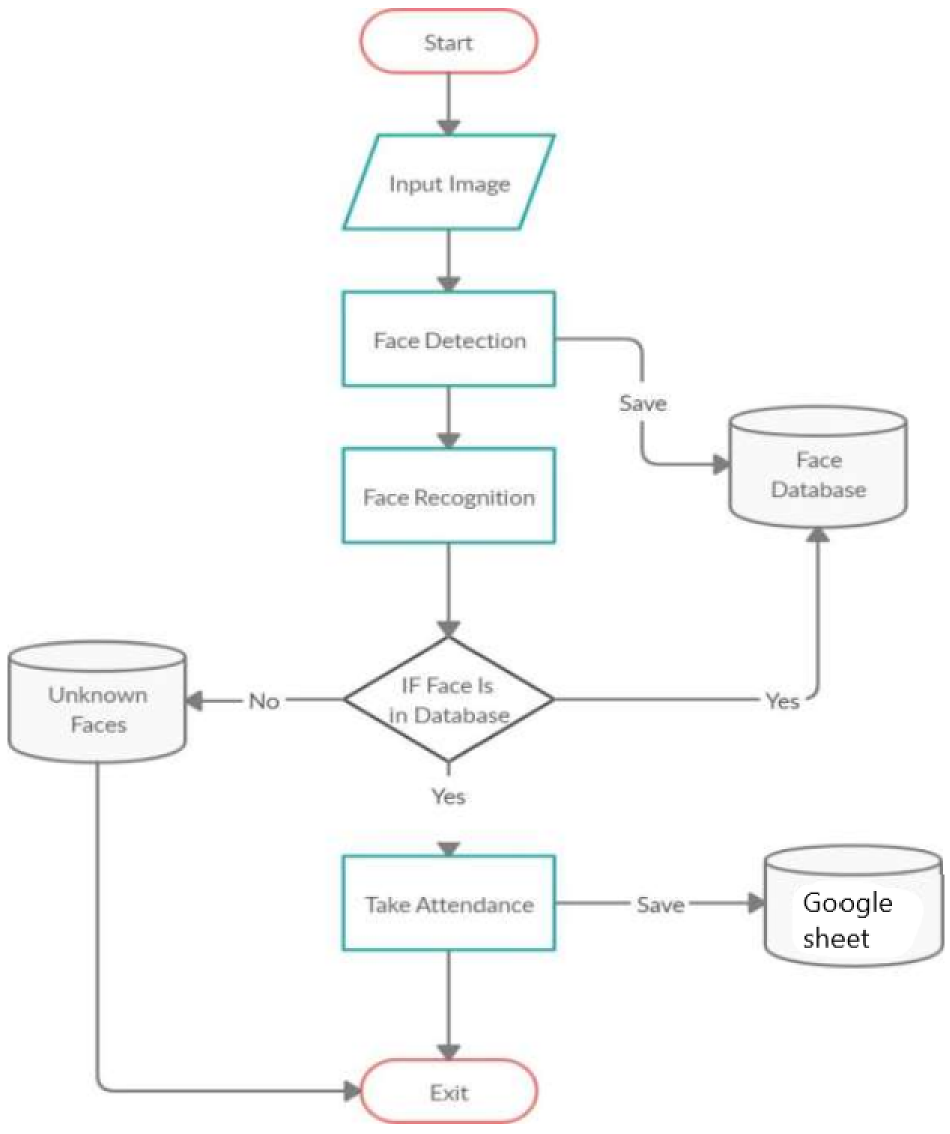
Chapter 4

Languages Used

- Python
- C++
- c

Chapter 5

Flow Diagram



Chapter 6

Code

import cv2

import numpy as np

import face$_r ecognition$

$importos$

$fromdatetimeimportdatetime$

$importconcurrent.futures$

$path =' Training_i images'$

$classNames = []$

$attendance = set()$

$Loadimagesandclassnames$

$forclinos.listdir(path) :$

$img = cv2.imread(os.path.join(path, cl))$

$ifimgisnotNone :$

$classNames.append(os.path.splitext(cl)[0])$

$Findencodingsforallimages$

$deffindEncodings(images) :$

$encodeList = []$

$forimginimages :$

$rgb = cv2.cvtColor(img, cv2.COLOR_B GR2RGB)$

$encode = face_r ecognition.face_e ncodings(rgb)$

$iflen(encode) > 0 :$

$encodeList.append(encode[0])$

$returnencodeList$

$images = [cv2.imread(os.path.join(path, cl))forclin$

$os.listdir(path)]$

$withconcurrent.futures.ThreadPoolExecutor()asexecutor:$

$encodeListKnown = list(executor.map(findEncodings,$
images

))[0]

$Startvideocapture$

$cap = cv2.VideoCapture(0)$

$Setupfacerecognition$

$whileTrue:$

$ret, img = cap.read()$

$ifnotret:$

$break$

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

$imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)$

$facesCurFrame = face_recognition.face_locations(imgS)$

$encodingsCurFrame =$

$face_recognition.face_encodings(imgS, facesCurFrame)$

$Usemulti-threadingtospeedupfacerecognition$

$withconcurrent.futures.ThreadPoolExecutor()as$

$executor:$

$matches =$

$list(executor.map(face_recognition.compare_faces,$
encodeListKnown

$*len(encodingsCurFrame),$

$encodingsCurFrame))$

$faceDis =$

$list(executor.map(face_recognition.face_distance,$
encodeListKnown

$*len(encodingsCurFrame),$

$encodingsCurFrame))$

$Identifyfacesandmarkattendance$

$fori, (match, faceDis)inenumerate(zip(matches,$

$faceDis)):$

$matchIndex = np.argmin(faceDis)$

$ifmatch[matchIndex] :$

$name = classNames[matchIndex].upper()$

$ifnamenotinattendance : Skipmarking$

$presentifalreadymarked$

$attendance.add(name)$

$y1, x2, y2, x1 = facesCurFrame[i]$

$y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4$

$cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)$

$cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)$

$cv2.putText(img, name, (x1+6, y2-6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)$

$Saveattendancetoafile$

$withopen('Test101.csv',' a')asf :$

$now = datetime.now().strftime(''f.write(f'name, now')$

$cv2.imshow('Webcam', img)$

$ifcv2.waitKey(1) == ord('q') :$

$break$

$cap.release()$

$cv2.destroyAllWindows()$

6.1 Code Explanation

The provided code is a Python script that performs real-time face recognition using the webcam feed and marks the attendance of recognized faces. Here's a step-by-step explanation of the code:

1. First, the necessary libraries are imported: cv2 for computer vision operations, numpy for numerical computations, facerecognition for face recognition algorithms, os for interacting with the operating system, and datetime for working with date and time.

2. The script defines some variables such as path, which represents the directory path where the training images are stored, classNames, an empty list to store the

names of individuals, and attendance, a set to store the names of individuals who are marked present.
3. The script loads the images from the path directory and extracts the class names (names of individuals) from the image file names.
4. A function named findEncodings is defined, which takes a list of images as input and returns a list of face encodings. It iterates over each image, converts it to RGB format, and obtains the face encodings using the facerecognition library. The obtained encodings are added to the encodeList list.
5. The script reads the images from the path directory, and using concurrent.futures.ThreadPoolExecu
it executes the findEncodings function in parallel for the images. The resulting encodings are collected in the encodeListKnown list.
6. Video capture is initiated using the webcam (index 0).
7. The script enters a loop where it reads each frame from the video capture. If the frame is not successfully captured, the loop breaks.
8. The captured frame is resized and converted to RGB format. The face locations and encodings are then obtained using the facerecognition library.
9. Multi-threading is used to speed up the face recognition process. The face encodings from the known images (obtained earlier) are compared with the encodings from the current frame using the facerecognition.comparefaces and facerecognition.facedistance functions.
10. The script iterates over each match and corresponding face distance. The closest match is identified, and if the match is above a certain threshold, the person's name is determined using the classNames list. If the person's name is not already in the attendance set, it is added, and a bounding box with the name is drawn on the image.
11. The attendance information is saved to a file named ”Test101.csv” in the format ”name,time”. The current time is obtained using datetime.now().strftime(”12. The processed image with bounding boxes and names is displayed in a window named ”Webcam”. The loop continues until the user presses ’q’ to quit.
13. Once the loop is terminated, the video capture is released, and all windows are closed. The code performs real-time face recognition using a webcam feed, compares the detected faces with the known images, and marks the attendance of recognized individuals. It uses multi-threading to improve the processing speed and saves the attendance information to a CSV file.

6.2 Algorithm used

6.2.1 CNN(convolutional neural network)

A convolutional neural network (CNN) is a type of deep learning neural network that is commonly used for image recognition. CNNs are able to learn to identify patterns in images, even when the images are rotated, scaled, or translated. This makes them well-suited for tasks such as face recognition, object detection, and image classification.

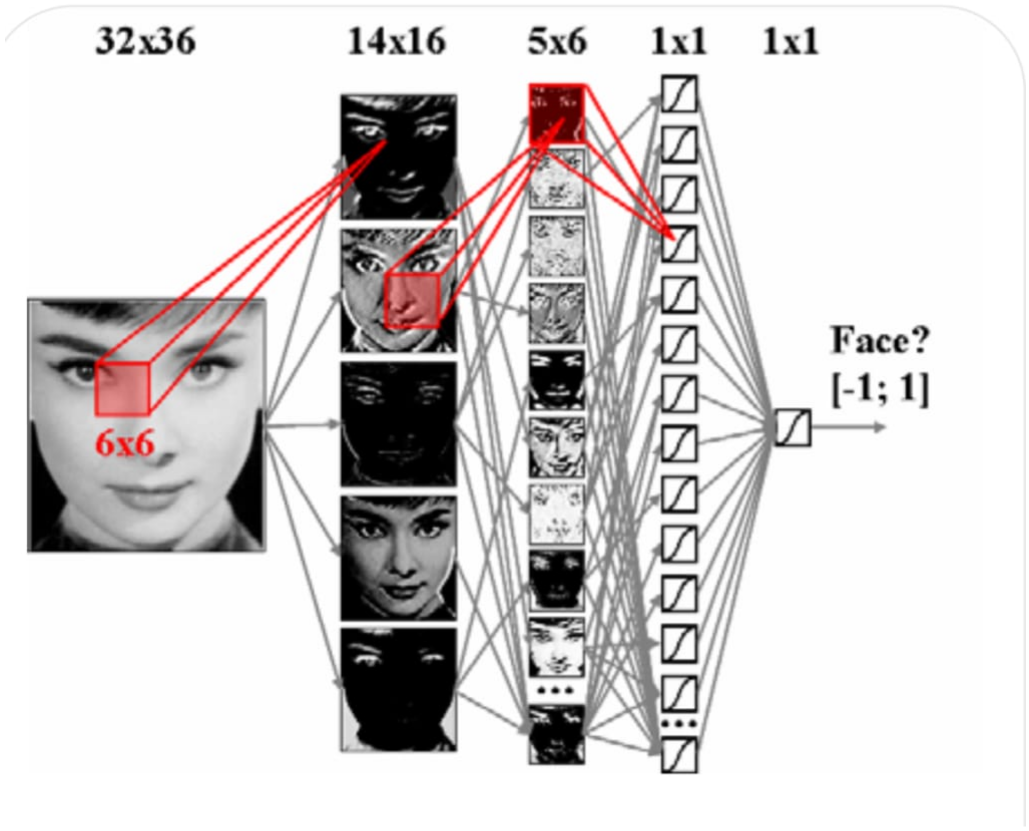
CNNs work by applying a series of convolution operations to the input image. A convolution operation is a mathematical operation that takes two functions as input and produces a third function. In the case of CNNs, the two functions are the input image and a filter. The filter is a small, rectangular array of numbers that is used to extract features from the input image.The convolution operation is applied repeatedly to the input image, moving the filter over the image one pixel at a time. As the filter moves over the image, it extracts features from the image and creates a new feature map. The feature maps are then used to classify the image.CNNs are able to learn to identify patterns in images because they are able to learn the weights of the filters. The weights of the filters are learned through a process called backpropagation. Backpropagation is an iterative process that adjusts the weights of the filters so that the CNN can better classify the images in the training dataset. CNNs are a powerful tool for image recognition. They are able to learn to identify patterns in images, even when the images are rotated, scaled, or translated. This makes them well-suited for tasks such as face recognition, object detection, and image classification.

6.2.2 Image Representation

An image is a 2-Dimensional light intensity function

f (x,y) = r (x,y) $\times$ i (x,y) -(2.0)

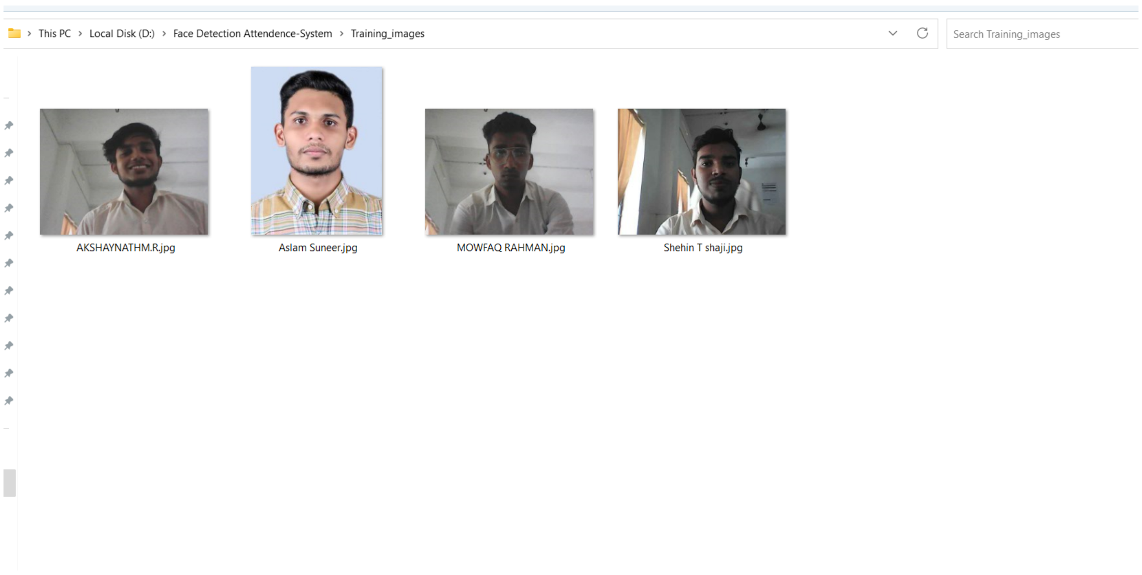
Where, r (x, y) is the reflectivity of the surface of the corresponding image point. i (x,y) Represents the intensity of the incident light. A digital image f(x, y) is discretized both in spatial co-ordinates by grids and in brightness by quantization. Effectively, the image can be represented as a matrix 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 256 × 256, elements, 640 × 480 pixels or 1024 × 1024 pixels. Quantization of these matrix pixels is done at 8 bits for black and white images and 24 bits for colored images.



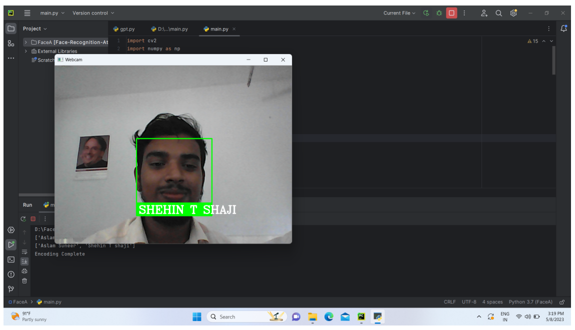
Chapter 7

Output

Training images:

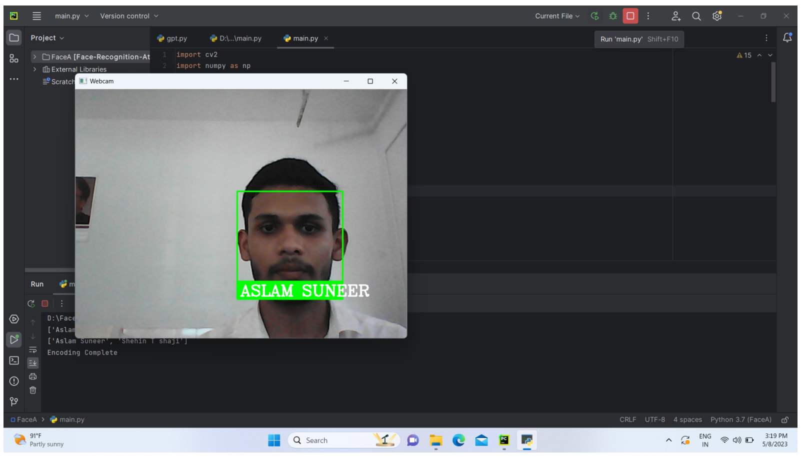


Face Detection:



```
import cv2
import numpy as np
```

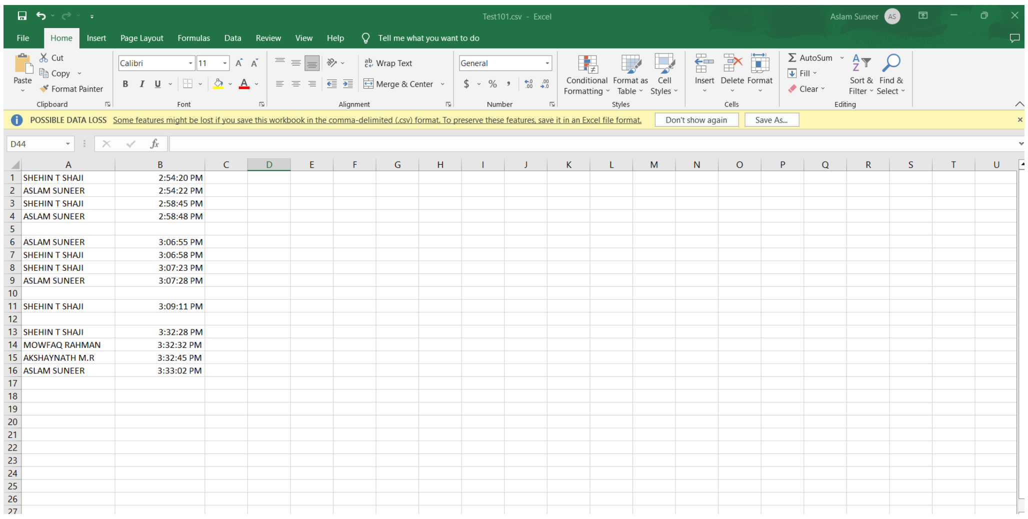
```
D:\Face
['Aslam
['Aslam Suneer', 'Shehin T shaji']
Encoding Complete
```



```python
import cv2
import numpy as np
```

```
D:\Face
['Aslam
['Aslam Suneer', 'Shehin T shaji']
Encoding Complete
```

Attendance in ExcelSheet:



Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B
1	SHEHIN T SHAJI	2:54:20 PM
2	ASLAM SUNEER	2:54:22 PM
3	SHEHIN T SHAJI	2:58:45 PM
4	ASLAM SUNEER	2:58:48 PM
5		
6	ASLAM SUNEER	3:06:55 PM
7	SHEHIN T SHAJI	3:06:58 PM
8	SHEHIN T SHAJI	3:07:23 PM
9	ASLAM SUNEER	3:07:28 PM
10		
11	SHEHIN T SHAJI	3:09:11 PM
12		
13	SHEHIN T SHAJI	3:32:28 PM
14	MOWFAQ RAHMAN	3:32:32 PM
15	AKSHAYNATH M.R	3:32:45 PM
16	ASLAM SUNEER	3:33:02 PM

Chapter 8

CONCLUSION

Face recognition systems are part of facial image processing applications and their significance as a research area are increasing recently. Implementations of system are crime prevention, video surveillance, person verification, and similar security activities. The face recognition system implementation can be part of Universities. Face Recognition Based Attendance System has been envisioned for the purpose of reducing the errors that occur in the traditional (manual) attendance taking system. The aim is to automate and make a system that is useful to the organization such as an institute. The efficient and accurate method of attendance in the office environment that can replace the old manual methods. This method is secure enough, reliable and available for use. Proposed algorithm is capable of detect multiple faces, and performance of system has acceptable good results.
The face recognition attendance system project has successfully leveraged the power of face recognition technology to automate the attendance process. The system utilizes a Convolutional Neural Network (CNN) algorithm to detect and recognize faces in real-time video frames. By comparing the detected faces with a database of known faces, the system accurately identifies individuals and marks their attendance. The project offers several benefits such as improved efficiency, accuracy, and convenience compared to traditional attendance methods. It eliminates the need for manual entry or paper-based systems, reduces administrative workload, and provides a reliable and secure way to track attendance. Overall, the face recognition attendance system project showcases the potential of face recognition technology in revolutionizing attendance management processes in various domains, including educational institutions, workplaces, and events.

Chapter 9

REFERENCES

1 OpenCvDocumentation -https://opencv.org

2 Numpy - https://numpy.org

3 “RFID based Student Attendance System” (Hussain, Dugar, Deka, Hannan, 2014)

4 Design of a Face Recognition System (PDF Download Available). Available from:https://www.researchgate.net/publication/262875649DesignofaFaceRecognitionSystem