DESIGNING AN INTEGRATED AIOT SYSTEM FOR TRACKING CLASS ATTENDANCE

KUAK XUAN REN

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Bachelor of Engineering (Honours) in Electronic Systems

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

September 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : KUAK XUAN REN

ID No. : 210AGB06806

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DESIGNING AN INTEGRATED AIOT SYSTEM FOR TRACKING CLASS ATTENDANCE"** was prepared by **KUAK XUAN REN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) in Electronic Systems at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr. Lee Han Kee

Date : 23/9/2024

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Prof. Dr. Lee Han Kee for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would like to extend my deepest appreciation to my friends Boey Zhi Xuan, Edward Yeoh Hong Enn, Chew You Hui, Cheah You Qing, and Tan Cheng Yong for the support they have provided me during the entire scope of work of this project. They are the kind people that volunteered their faces to assist me in capturing faces for my face recognition system dataset. I would personally like to thank those individuals who have helped me in beginning and even more in maintaining the project. They proved to be highly useful in the course of doing my work.

DESIGNING AN INTEGRATED AIOT SYSTEM FOR TRACKING CLASS ATTENDANCE

ABSTRACT

The project aims to develop an automatic facial recognition system that supports artificial intelligence (AI) and the Internet of Things (IoT) for attendance system. The system is able to capture the students' real facial feature data, and uses it as a tool to achieve high-accuracy student identification. Basics and ensures that the software is more secure than the previous traditional attendance method using roll-less slides. For face detection the system uses a YOLO (You Only Look Once) algorithm, which allows quick and efficient recognition of student faces in the classroom context. For facial recognition, deep metric learning methods which involve face encoding are used, and the system can highly match student faces with relevant confidence level of 0.75 or above. In the current work, face recognition is carried out using the ResNet-34 deep convolutional neural network to produce a 128 -dimensional face vectors for the identification. Records for attendance control are kept in the Excel, and this cuts down the time taken to record attendance since Excel has inbuilt facilities for calculating the percentage attendance of each individual. The system shows optimal performance as well as scalability with the overall achievement of the goals that include automation, accuracy and efficiency in attendance tracking. Further enhancements including integrating of night vision cameras and research on face recognition algorithms that utilizes GPU can improve the system performance. The objective of this project is accomplished in the creation of a reliable, scalable and user-friendly system for facial recognition attendance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
YOLO	You Only Look Once
VSCode	Visual Studio Code
IoT	Internet of Things
UI	User Interface
CNN	Convolutional Neural Network
dlib	Deep Metric Learning Approach
CUDA	Compute Unified Device Architecture
IDE	Integrated Development Environment
OpenCV	Open-Source Computer Vision
NLP	Natural Language Processing
CPU	Central Processing Unit
GPU	Graphic Processing Unit
ID	Identification
F. O. V	Field of View
IoU	Intersection over Union

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

As part of digitalization and the integration of modern technology, AI has greatly influenced the aspect of the IoT in several industries including education. One of the most encompassing areas where the concept of smart automation can be used is in the area of operation of a classroom attendance system since this is a better system, efficient, accurate, and secure than other traditional methods of doing it. Some of the current methods of attending classes include signing a student list or scanning a QR code, which the following drawbacks – cheating, or missing those arriving late, and time consumption to do it because of students' interaction.

Some of the challenges observed with the manual attendance systems have been solved by integrated with the automated attendance systems using AI & IoT. Using such systems not only brings about an added bonus of reducing the time that the clerks take to input the data into the system, but also increases security and reduces the probability of error in maintaining records of attendance. Automated systems may improve the overall efficiency of classes, ensure proper keeping of registers and eradicating default that is associated with manual systems.

There has been a huge change in the strategies and ideas used in face recognition for identification over the years. In the 1960s, semi-automated were develop this system by which major facial points such as the eyes, ears, and mouths were manually marked, distances and ratios calculated for comparison. Such systems by the 1970s included the Goldstein, Harmon and Lesk system, which uses 21 subjective markers but very hard to automate because they involved measurements in a very general and subjective way (Goldstein, Harmon and Lesk, 1971). Fisher & Elschlagerb proposed the world standard map of facial feature points yet how did not have enough distinguishability to allow the identification of the adult facial images (Fischler and Elschlager, 1973).

Then Connectionist approach used pattern recognition and neural networks to classify the human faces, however it required large training data sets that hamper its usage (S.S.R. Abibi, 2002). The first fully automated system used general pattern matching by comparing faces to the general face model reducing data with histogram and grayscale values (Cui et al., 2009). The above developments helped shape the current advanced AI based facial recognition technological solutions that are now essential especially in automating functions such as students' attendance.

In recent years, the face recognition technologies have improved due to the development of deep learning along with the real time object detection algorithms such as YOLO, thus making it more accurate and efficient to apply this to generate auto-attended systems in schools. Thus, integrating these technologies does not only overcome the limitations of manual systems, but also offers a secure, large-scale distinctive solution that is in line with the digitization concept.

1.2 Problem Statements

Traditional attendance tracking methods, such as manual roll call, scan QR code or paperwork sign-in, are error-prone, time-consuming, and may not accurately reflect an individual's actual presence. The proposed intelligent face recognition system solves these problems by introducing a reliable, efficient, and automated attendance management method.

1.3 Aims and Objectives

The objectives of the thesis are stated below:

- To design an AI-based face recognition system for class attendance.
- To analyze the data collected from the designed class attendance system.
- Project streamlined management in the class attendance system.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, the basic concepts about AI, OpenCV, TensorFlow, Deep Learning, Machine learning, CNN, ResNet34, dlib and CUDA will be described with a detailed overview of the ideas. Also, the analysis of the selected topic will be enriched with the help of articles from other journals identified during the literature review.

2.2 Artificial Intelligence (AI)

AI means ability of a digital computer or a computer-controlled robot as an intelligent creature to solve problems of an intelligent nature. This term is normally used when attempting to refer to the creation of procedures that may replicate abstract human thought, including reasoning, finding meaning in something, generation of one concept from another, or learning from past events. Digital computers have been designed since the period of 1940s and 1950s to solve very intricate problems, for instance, solve a logical theorem or play a game of chess.

However, judging by the processing speed and the memory volume of AI devices, there are no programs that have complete open-endedness of all more or less specialized AI approaches to certain domains or that are capable of performing routine

extensive practical tasks that require daily knowledge. Since, some computer programs incorporate AI and have acquired an expertise in accomplishing specific tasks, and hence their effectiveness in other areas of application including medical diagnosis, search engines, voice, handwriting and face recognition, and also the chatbots.

2.3 Open-Source Computer Vision (OpenCV)

OpenCV or also known as Open-Source Computer Vision Library is a well-known open-source computer vision and machine learning library. As a framework intended to undergird various computer vision applications and to facilitate the deployment of machine perception into manufactured goods, OpenCV is under the Apache 2 license so that businesses may conveniently employ and adapt the code. It contains 2500 optimized algorithms for the most popular computer vision algorithms in the industry and some of the newer ones as well falling under machine learning. These algorithms can be used in tasks as face detection or recognition, object identification, action classification in videos, tracking camera and objects, 3D model extraction, image stitching, red-eye elimination, augmented reality etc.

OpenCV has been adopted by more than 47,000 users and downloaded more than 18 million times and it is used by companies, research groups, and departments of government. Some of its notable users include google, Microsoft, Intel and Sony together with new companies like Applied minds and VideoSurf, where OpenCV is

used to stitch street view images, monitor and detect accidents in mining equipment and in swimming pools respectively. The library is compatible with many operating systems, for example, Windows, Linux, Android, and MacOS. The programming languages which are supported are C++, Python, Java, and MATLAB. OpenCV is optimized for real time vision and uses architectures such as MMX and SSE with active development of full featured CUDA and OpenCL support. OpenCV has C++ interface and heavy use of templates, and it plays well with STL containers, has more than 500 algorithms, and thousands of utility functions.

2.4 TensorFlow

TensorFlow is machine learning framework coded in Python by Google and announced in 2015, intended primarily for the deep learning but in addition supports typical machine learning tasks as well. Developed for big scale number computing, TensorFlow is not originally designed for deep learning but its success in the field made Google decide to release it via open-source license. TensorFlow uses multi-dimensional arrays which are called tensors and are perfect for dealing with big data.

TensorFlow's workflow works around data flow graphs where each computation is predetermined by nodes and edges. This graph-based execution model allows TensorFlow to arrange the computation through a cluster of computers and use GPUs for performance boost. This was makes TensorFlow highly suitable for managing complex models and large-scale machine learning application.

2.5 Machine Learning

AI, which includes sub-discipline referred to as machine learning is the science whereby machines are given the capacity to improve their performance by using

certain knowledge in data or prior experiences relating to a specific problem such that the machine is able to predict outcomes without heavier human intervention (IBM, 2023). AI entails that computers work on their own without even being programmed to do so. New data is provided to the applications of machine learning, and they can learn on their owns, evolve, even change, and define themselves. Machine learning is a technique of obtaining information from big data through the use of algorithms, which strengthens the learning process in a cyclical manner. Information from the data is fed into the specific computational methods and the outputs, which are the machine learning algorithms, are not based on any fixed equation that can be considered as the model.

Machine learning algorithms perform with increasing effectiveness in the 'learning' processes when more samples are present and fed into the models. For instance, deep learning is a subset of a more general field termed as machine learning that enables computers to mimic natural aspects of human learning such as learning from samples. It gives improved performance parameters than the other traditional machine learning algorithms. Machine learning in its basic terms has been around since the World War II when the first device to use prescribed learning was called the Enigma Machine.

2.6 Deep Learning

Currently, deep learning, also known as representation learning, is used in many applications. Increased interest in deep and distributed learning is due to the increasing

volume of data and tremendous improvements in hardware technologies. Although deep learning stems from conventional neural networks, it is more effective than them in comparison. Converting and graph technologies are utilized by deep learning to build context dependent multiple layer learning models. The deep learning technology used provides excellent performance in various application fields such as audio and speech processing, visual processing, and natural language processing (NLP).

The structure of the deep learning models includes the input layer, a number of hidden layers, and the output layer. The data is passed through each layer encoded and fed to the subsequent layer hence hierarchal victimization of the input data. This structure of layers makes it easier for the model to execute functions such as classification and pattern identification all without having to write code to extract features.

Similar to other methods of learning new skills where one has to keep repeating it, the deep learning also needs large amounts of data in order to train an effective neural network. Deep learning is flexible insofar as it has enough data feeding into it since it can then be used in a lot of different ways.

2.7 YOLO

YOLO (You Only Look Once) is a real-time objective detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015. It is a one-stage object detector that predicts the bounding boxes and class probabilities of objects in input photos using a convolutional neural network (CNN). Initially, YOLO was implemented with the Darkent framework. The YOLO algorithm segments an input image into a grid of cells and predicts bounding box coordinates and the likelihood of an object being present in each cell. It also predicts the category of the object. YOLO is faster and more efficient than two-stage object detectors such as R-CNN and its variants because it processes the entire image at once (Redmon et al., 2016).

YOLO has developed several versions, including YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8. Each new version builds on the previous one, offering more features including greater accuracy, faster processing speeds and better handling of small objects. YOLO is widely utilised in many different applications, including surveillance systems and self-driving vehicles. Additionally, real-time object identification tasks like real-time video analysis and real-time video surveillance make extensive use of it (Redmon et al., 2016). In this project, YOLO will use the CNN as its underlying architecture. The relationship between YOLO and CNN is foundational, as YOLO relies on a CNN to perform real-time object detection as shown in Figure 2.1 (Viswanatha, Chandana R and Ramachandra, 2022).

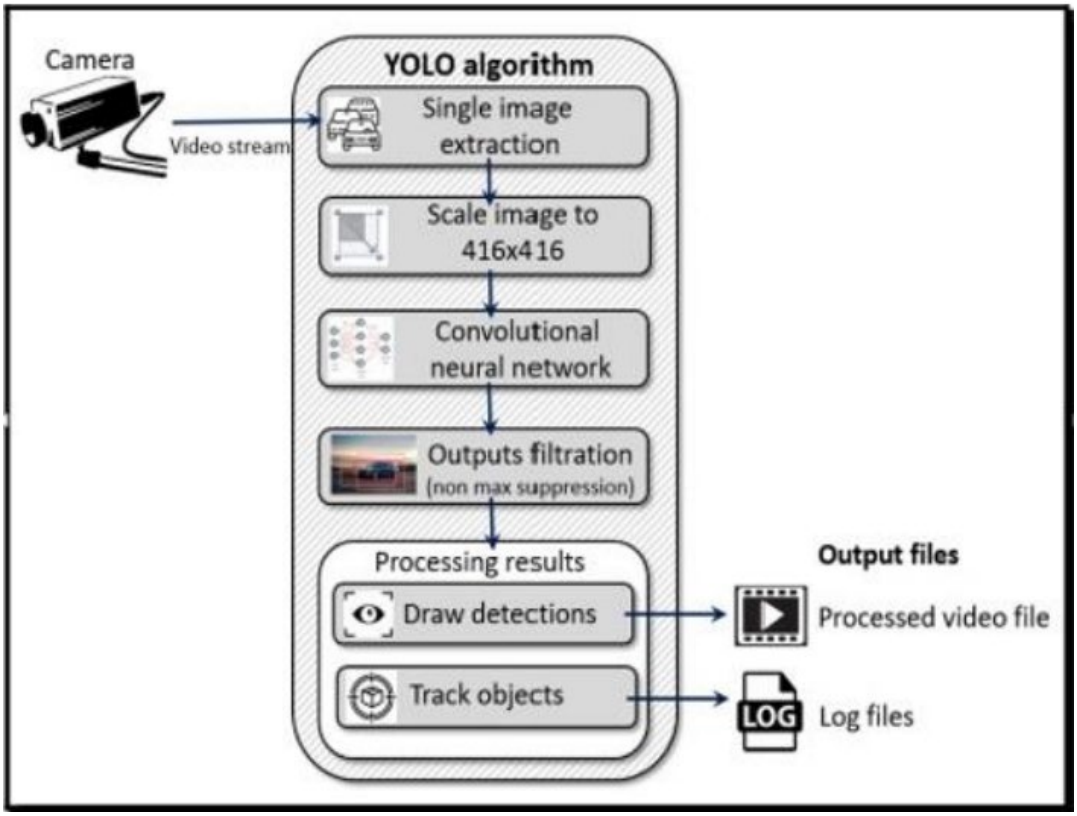


Figure 2.1: YOLO algorithm using CNN (Viswanatha, Chandana R and Ramachandra, 2022)

2.7.1 Why YOLO?

YOLO because of its accuracy and speed, and because it is used in real-world projects some of the reasons that mark its main strength. First it gives real-time detection of the objects something which is cooperate for those applications which needs continuous detection like video surveillance systems and self-driving cars. Secondly, because there is only one detection method, it is relatively fast as compared to other methods that take several steps or passes and which works on the detection network on the whole image at a time. Nonetheless, speed in YOLO's object detection is achieved through the use of a unique approach that combines CNN while also serving as a speedy approach (Mirkhan, 2023).

Moreover, YOLO is effective in identifying the small objects within the image due to grid-based approach. Last but not the least; anchor boxes enhances its versatility

of handling items at different scales within one image and makes it adequate to several datasets and applications. Combined, these characteristics make YOLO a robust and flexible solution for practical tasks related to object recognition and, due to this, YOLO is in high demand and applied in many areas (Mirkhan, 2023).

2.7.2 YOLOv7 vs YOLOv8

The efficiency of YOLOv8 and YOLOv7 is evaluated on different key parameters, such as speed in terms of frames per second (fps), and accuracy of bounding box prediction, mean average precision (mAP), and the model size. Real-time object detection benchmarks prove that YOLOv8 is the way to go since it offers convincible gains over YOLOv7 in several critical categories. Another benefit for using YOLOv8 over YOLOv7 is that it has a higher frame-per-second (FPS) rate than the former. According to the optimised algorithms and better architectural model, is also the enhanced object detection.

This means that for the same given amount of time YOLOv8 can analyze more frames than other models, thanks to this it is perfect for real-time applications. Due to the fact that YOLOv8 has a superior model structure then by the factor of object detection it can be more accurate and reliable. In addition, YOLOv8 has better mean average precision (MAP) scores compared with YOLOv7 for all cases studied. The prime measure that holds significant importance in object detection systems is accuracy and precision, and these two factors are responsible to quantify the ability of the model to locate along with identifying every item contained in a particular image or a video. Due to increases in these results, YOLOv8 is able to offer a greater and more accurate range of uses that may be applied in the real world.

To enhance the architecture of YOLOv8 more enhancements have been done on its ability of detecting the objects. YOLOv8 design cancels the need for anchor boxes hence making it an anchor-free architecture. Such design enhances the model’s

ability to detect objects with variations in size and aspect ratio while at the same time reducing its training complexity (Abramov, 2023). Further, multi-scale prediction is embedded in YOLOv8, which means that the model is capable of predicting at different scales. This multi-scale method improves the accuracy of the item detection as well as this model's capability to capture things at different scales. The following figures are the comparison of YOLOv8 with other versions (Augmented A. I., 2023).

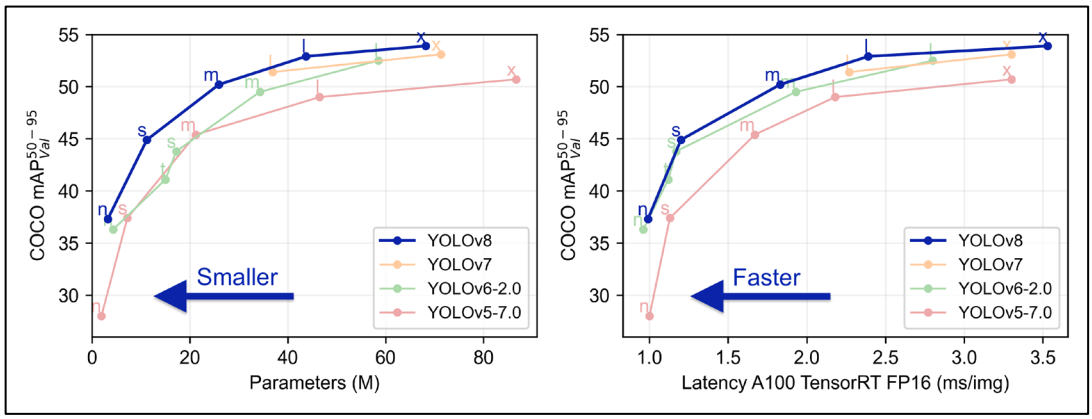


Figure 2.2: Comparison between YOLOv8 and other YOLO versions (Augmented A.I., 2023)

YOLOv8 is the latest model of all the YOLO models that are based on deep learning for the identification of objects of substantial size. As a result, it comes with a relatively simple model with no horizontal anchor, prediction at multiple scales, and an enhanced backbone network that makes it an ideal choice for real-time object detection. This makes YOLOv8 as a reliable solution for many who are in search of an efficient and effective object detection program.

2.8 Convolution Neutral Network (CNN)

In general, there is a clear structure of the CNN models despite the many modifications in the models available for use. This framework comprises of an input layer, layers with convolution as well as pooling, one or more fully connected layers, activation functions and an output layer. The first part of the network is the feature extractor which is made of several convolutional and pooling layers arranged in succession. This feature extractor then extracts the feature representations from the given raw input data set and makes it more abstract and at a higher level. The features are then extracted through fully connected layers after which these layers and the activation functions are utilised for the classification or regression of data as required. To improve the CNN performance, some regulatory aspects such as batch normalization and dropout are included in the layers; different mapping functions (Bouvrie 2006). Figure 2.3 shows the architecture of a CNN broken down into its sections (Zhao et al., 2024). These are the components that have to be configured in order to create new architectures and to increase the performance. Knowledge of these components and their unique implementation is crucial for comprehending the progress made in CNN architecture in the study of computer vision (Bhatt et al. 2021).

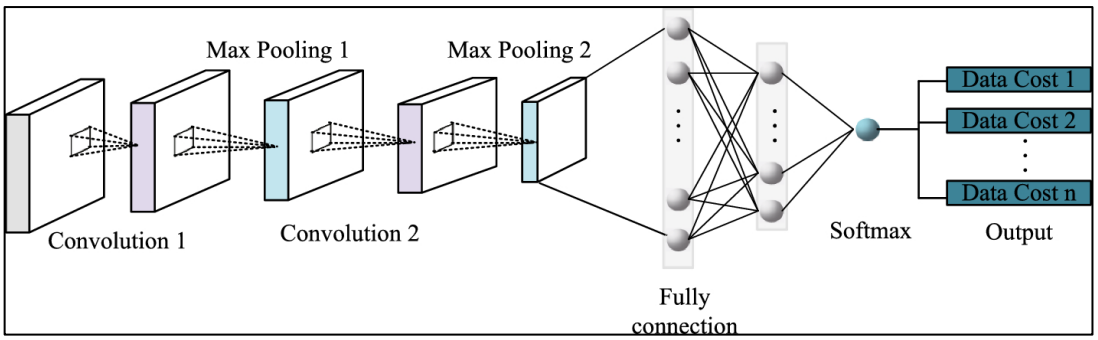


Figure 2.3: The Structure of CNN (Zhao et al., 2024)

In the CNN designed for n-class classification problem, on the first layer, the raw data is subjected to two convolutional layers such as Convolution 1 and Convolution 2 and followed by two pooling layers namely Max Pooling 1 and Max Pooling 2. After this the data is passed to the fully connected layer of the network for further analysis. Lastly for the output layer, the outputs going into it are passed through the SoftMax function which scales the outputs between 0 and 1. Thus, the vector outcome called Data Cost 1 represents the probabilities of the data belonging to the defined n categories and the higher data cost value indicates a higher possibility of the data to be introduced into the certain unique category.

CNNs need data preprocessing before one feed the primitive data into them to be analysed or interpreted. Some of the preprocessing steps include homogenization (Stepanov et al, 2023), normalization (Huang et al, 2023) as well as PCA (Uddin et al, 2021). The homogenization is a process that consist in the subtraction of the average value calculated from the entire training set from all dimensions of the input data aiming to centre the data at zero. Normalization scales the data difference between maximum and minimum and sets the new values of the data within zero and one. On the same note, as is the case with all normalization techniques, PCA can normalize the input data for each feature independently and thus also address the problem of high dimensionality and high correlation between the parameter dimensions.

2.9 ResNet 34

ResNet-34 is a state-of-the-art image classification model which is used in ResNet-34, a 34-layer convolutional neural network introduced in the paper Deep Residual Learning for Image Recognition (He et al., 2016a). As opposed to any other conventional neural networks, this model uses the output of every layer, passed to succeeding connected layers as residuals. This approach is similar to residual connections used in text prediction model reducing the problems such as vanishing gradient and allowing the networks to be trained deeper.

The ResNet-34 has been trained on the ImageNet dataset, which has more than 100,000 images split in 200 classes. The amount of pre-training achieved here is more, it allows the model to identify as many classes of images as possible. Apart from the ImageNet, a dataset containing more than 130,000 images of 200 classes is Tiny ImageNet, which also enhance the model. For the purpose of gaining from the strengths of this pre-trained model, a process known as “transfer learning” is employed in this, the pre-trained ResNet-34 checkpoint is fine-tuned to suit the requirements of our task. This approach enables us to expand upon the tremendous amount of previous work carried out with the ImageNet dataset and improve upon the existing model for new, related tasks.

The architecture of ResNet-34 is illustrated in Figure2.4 on the right side. It explains the way in which the network is constructed with 34 layers and highlights one of its key features that is a residual connection in ResNet. These residuals pass from layer to layer giving room for skipping some layers on the way to subsequent layers by the networks. This architecture has greatly helped in avoidance of problems like vanishing gradients as well as enabling more efficient training of deeper neural networks. In addition, these connections are essential input-output lines for the flow of data through an all-other network that make it perform better during training and converge more quickly.

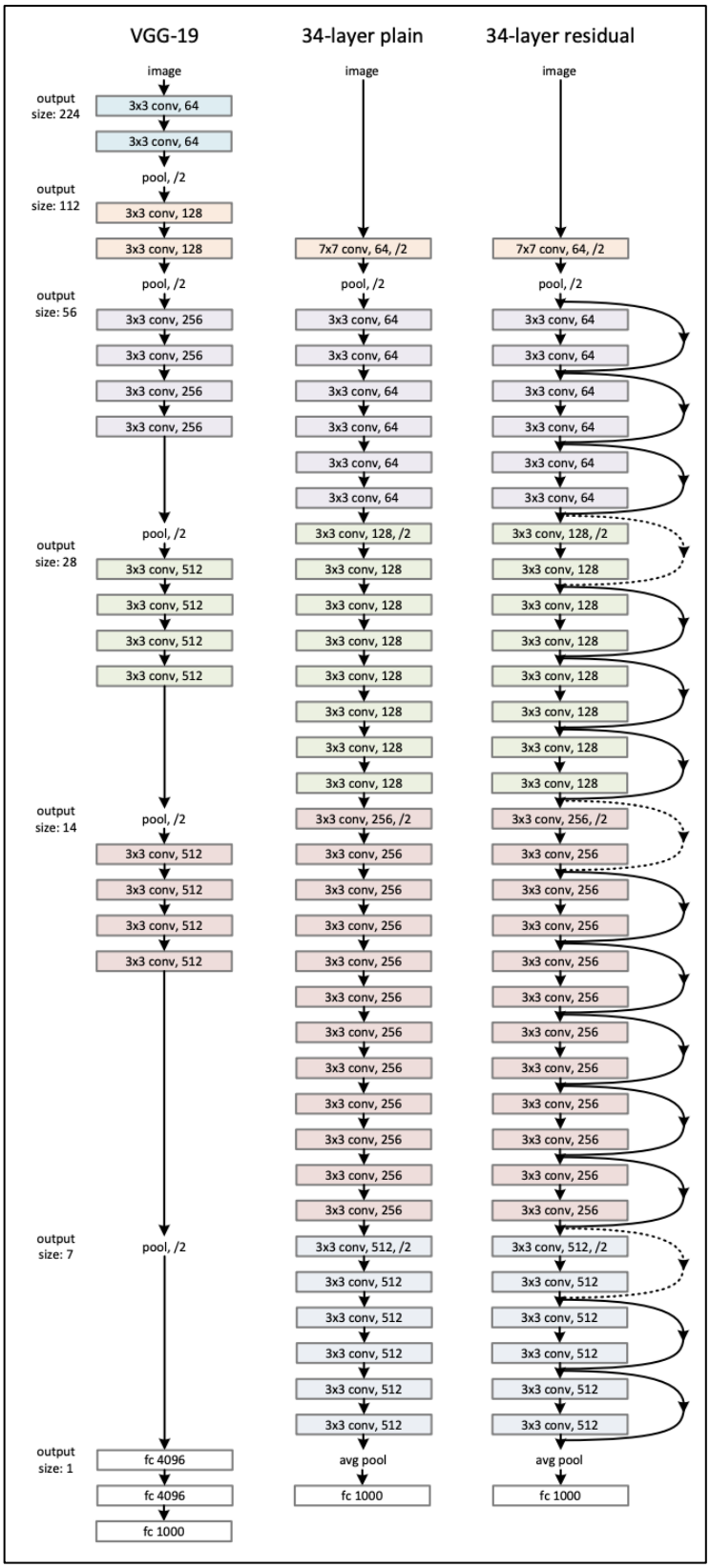


Figure 2.4: The Resnet34 layer architecture (He et al., 2016a)

2.10 Deep Metric Learning Approach (Dlib)

The face recognition module in this system uses Deep Metric Learning, a learning approach. This method intends to compare the dissimilarities in the various samples thereby finding the optimal distance in the application which is a Euclidean distance. Unlike most models that compute only one of the labels such as the object coordinates or the bounding boxes, the model returns a real valued feature space of the face in the image. Face recognition is done using the machine learning library, Dlib. The built-in face recognition model in Dlib is formulated on the ResNet-34 model, which was trained by Davis King using a number of images around 3 million (Rosebrock, 2018). The output of this ResNet model is 128 vector of real values (embedding) which describes a face. The output vector enables the system to represent every face as 128 numbers which represents distinct features of a face (elifezgisen, 2023).

The process starts from the feeding of a face image into ResNet 34 model. This is then followed by the model generating a 128-dimensional vector as the final output. When it comes to comparing the 2 faces, the system derives 2 such vectors from the face in questions. The system used the controlled vocabulary to compare the two faces to determine if they are of the same person The Euclidean distance or the Angle between the two vectors is then computed. Here, the measure of similarity that is adopted is the Euclidean distance; the similarity cut-off point being 0.6. If face distance is less than this value it means two face vectors represent faces of the same person. If this distance is higher than threshold, the faces are considered different ones (elifezgisen, 2023).

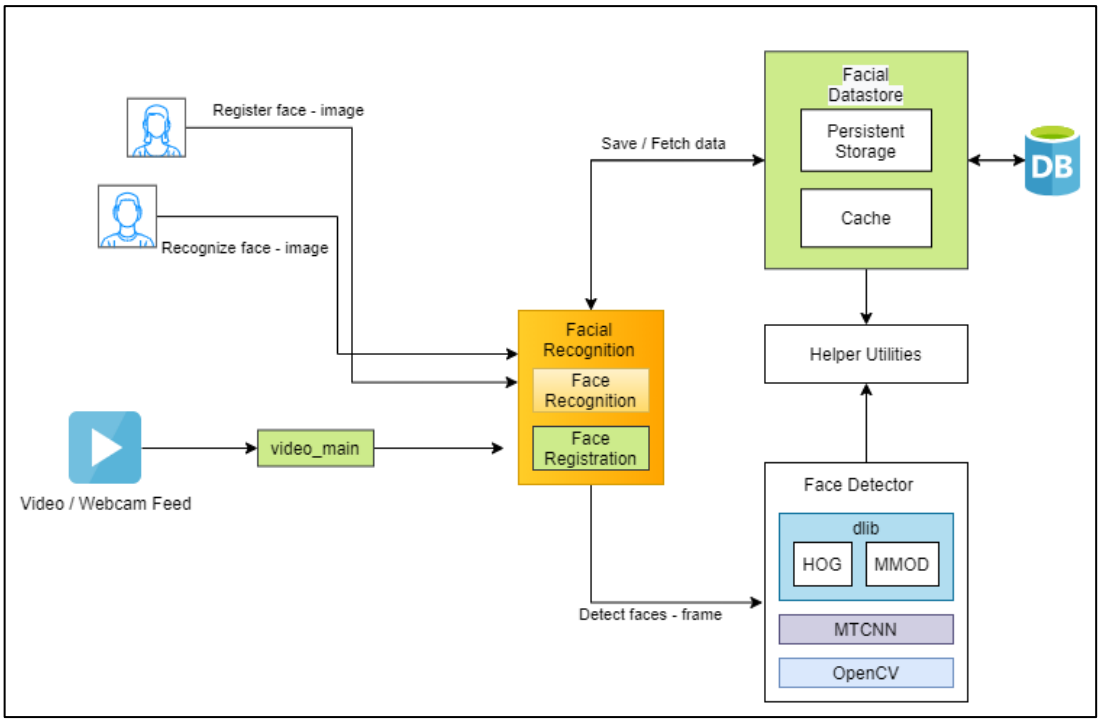


Figure 2.5: Face recognition flow (E King, 2009)

The face recognition system follows a structural workflow that involves various components in order to obtain accurate facial recognition. The process starts with acquiring media input which has to be processed frame by frame. Every frame of the image is then subjected to face detection in a bid to identify the position of face within the image. Once the target faces are found the faces are further defined into Facial Regions of Interest or Facial ROI, which in simple terms are portions of the face region which are of particular interest. These ROIs are then passed through a Neural Network yielding a 128 dimensions facial encoding for each face. This encoding contains the areas of the face, which are distinctive, and they are sent for comparison as well as to identify.

The system consisting of several components that are considered to be crucial. The face detection component was implemented with four kinds of different face detectors for better cropping options. The face recognition submodule is a one-stop entity responsible for all the functions relatable to faces including facial registration and processing. Cache services and permanent storage are also offered by the storage system, while cache is based on Python's built-in structures and permanent storage uses JSON. Users get an option of extending these abstract classes so that they are able to incorporate more elaborate storage systems in case it is necessary. Finally, the utilities component holds methods for image or video operations and validations as well as other operations necessary for the operation of the system. Altogether, these components enable effectiveness of the face recognition system and allow identification of facial data with maximum accuracy.

2.11 Compute Unified Device Architecture (CUDA)

CUDA for short of Compute Unified Device Architecture is a hardware and software architecture developed by NVIDIA Company that revealed on 15 February 2007. It allows concurrent computations on the GPU and supports all NVIDIA GPUs starting from the G8x series including GeForce, Quadro and Tesla. The entry of the CUDA platform in GPU changed the way programmers used it by providing a high parallelism computing platform that was also flexibly designed. CUDA, through its API issue a C-like interface for software developers to write programs that may implement both on the CPU and GPU without specialized knowledge on computing graphics.

Program developers can use CUDA in the form of CUDA libraries that can be incorporated into programs, OpenACC directives located in source codes, C, C++, Fortran, and other programming languages as extensions. It offers a more direct access to GPU's figured instruction set, memory and parallel processing element. This provides developers a tremendous amount of computational capability as CUDA is suitable for general purpose parallel processing which involves the use of GPUs for carrying out multiple threaded processes. CUDA facilitates fine-grained parallelism which results into efficient utilization of the inherent parallelism of the GPUs which

makes it a useful tool for computation intensive problems in research or software development (Paramjeet Kaur and Nishi, 2014).

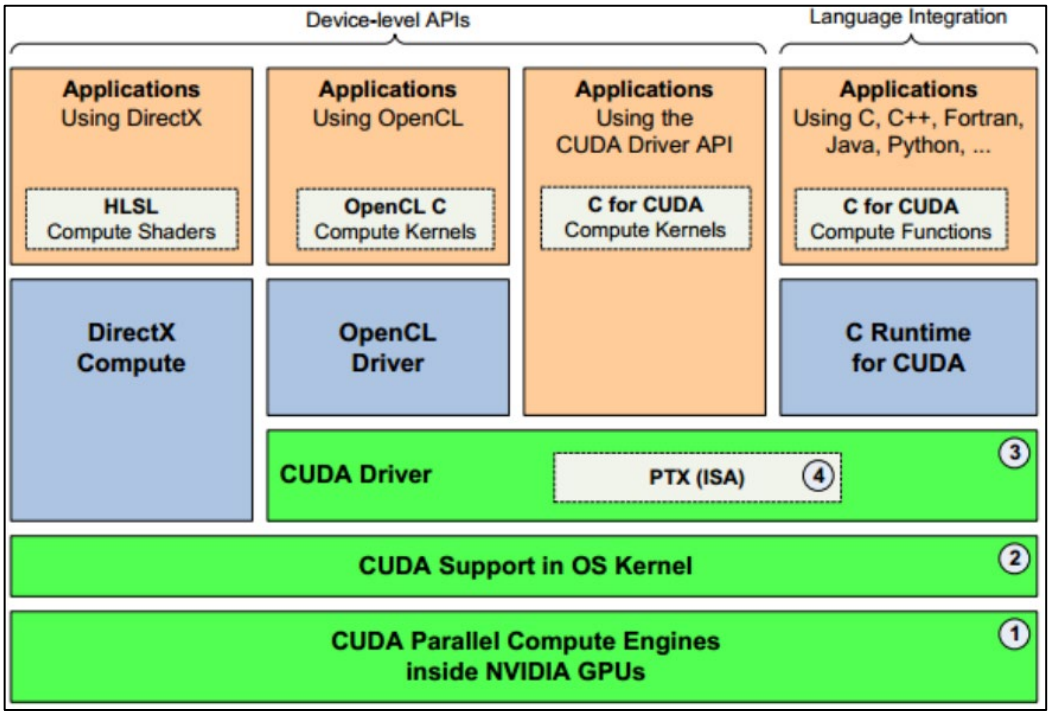


Figure 2.6: Components of CUDA architecture

CUDA also allows GPUs to employ a parallel throughput operation with an aim of completing many threads at a comparatively slower rate as compared to CPU that targets a single thread. First, there are parallel compute engines within each NVIDIA GPU where these immense parallel workloads are solved. Second, the architecture provides support at operating system kernel level for the operations like system hardware initialization and configuration. Third, it presents a user-mode driver which gives developers device-level access which in turn allows them to feed code in a way that interacts with the GPU. Finally, the CUDA programming model employs the PTX (Parallel Thread Execution) instruction set architecture, which encompasses parallel computing kernels and functions to help the developer to manage and perform the parallel operation in the GPU (Kirk, 2007).

2.12 Journal Reviews

According to the research paper written by Mirkhan (2023), YOLO, algorithm is famous for real-time object detection, high accuracy and speed. YOLO is effective in applications like video surveillance and self-driving vehicles since it identifies and locates objects with a single feed through the network for optimum speed. It is designed in such a way that it works with multiple object scales, efficient for small objects besides, it has fully convolutional network architecture which is friendly on GPU. However, YOLO is not without limitations for instance it was mainly designed to perform object recognition hence if applied to other uses like image or instance segmentation it may not perform optimally. It is not very accurate as some other two shot detection methods, performs poorly with very small objects and does not include a tracking mechanism, which is a disadvantage for tracking the object through time.

According to the research paper by Redmon et al. (2016), YOLO is the new approach in the object detection since it handles it as regression not using the classifiers. YOLO predicts bounding boxes and class probabilities without the need for scanning the whole image through different transformations and thus can be trained in an end-to-end manner by considering detection performance. This unified architecture allows one to have high-speed of processing, the base model YOLO model can achieve real-time of 45 FPS, and the smaller Fast YOLO model can achieve 155 FPS with the double of the mAP of other real-time detectors. While making more localization errors compared to some state-of-the-art systems, YOLO is less sensitive to false positives from the background and demonstrates consistently high performance in translating object representations from one context to another including natural images and artwork.

According to the research Augmented A.I. (2023), YOLOv8 provides several enhancements over YOLOv7 that has been listed in the features above. The improvements of YOLOv8 include such parameters as a greatly boosted speed, improved accuracy, and absence of anchors, prediction of different scales, and an advanced backbone network. YOLOv8 yields much better results than YOLOv7 with a boost on real-time object detection with higher FPS, especially with smaller models such as the YOLOv5 Nano. This model also comes with a relatively higher mean

average precision (MAP) of 53.7, therefore indicating an improvement in precision especially for small objects. Specifically, YOLOv8 eliminates the need for anchors for setting up the model, makes the training of models easier across the different datasets, and has improved the prediction for scale objects through a multi-scale prediction mechanism. Also, there is enhanced architecture called YOLOv8 with a better backbone network and the extra improvements in the model structure, including pose estimation models that make this architecture more effective and versatile. As an improvement from the previous version, YOLOv8 has a better architecture design with better I/O since it is easy to use and implement hence easy to make modification of the model when it is going to be deployed to different object detection tasks.

According to the research paper by Bhatt et al. (2021), computer vision is evolving particularly in the image processing sector since automation of object recognition is increasingly becoming essential. This has been made possible by CNNs which has also produced remarkable performance on different domains like video processing, object recognition, image classification and segmentation, natural language processing and speech recognition. Advancements in the amount of data available and cheap and readily available hardware has further boosted the research into CNN, thus leading to the development and exploration of other ideas such as different activation functions, different methods of reducing overfitting, optimisation of parameters, among others as well as architectural enhancements. It provides a comprehensive survey of recent developments in CNN architectures, categorizing them into 8 distinct groups: spatial exploitation, multi-path, depth, breadth, dimension, channel boosting, feature-map exploitation and attention based CNNs. The primary findings of the paper are as follows, a comparative review of the changes that have taken place in CNN architecture. The key innovations and their strengths and weaknesses. It also provides details on various parts of CNNs, pro and cons of different types of CNNs, literature that has not been covered yet, different uses of CNNs and potential future work.

From the work of Huang et al. (2023) on the use of DNNs for training, normalization methods are important in increasing the rate of convergence as well as the ability of DNNs to generalize. The paper surveys normalization techniques and presents a common view on the optimization objectives that underlie them. It

introduces a taxonomy to clarify the similarities and differences between various approaches, breaking down the pipeline of prominent normalizing activation methods into three components, circulation subdivision of the normalization zone, the processes of normalization, and the restoration of the representation of normalization. This helps in understanding the design of other normalization techniques which are being put in place. The paper also provides a survey of recent progress in the study of normalization approach and the real-world usage of these methods as examples is also provided to show how these methods solve some of the problems in the given problems.

According to the research paper of He et al. (2016a). The present research outlines ResNet-34, which is a convolutional neural network with 34 layers for images classification. The original document entitled “Deep Residual Learning for Image Recognition” introduced this concept. Unlike the conventional networks, it takes advantage of residual connections to facilitate information flow between different layers thus mitigating challenges such as vanishing gradients ultimately making it possible to train deeper neural networks effectively. ResNet-34 is based on pre-trained model using large-scale ImageNet dataset that comprises of more than 100k images distributed into 200 classes. Through transfer learning, this model can also be fine-tuned from its original state and consequently perform better on some particular tasks.

According to the research paper of Viswanatha, Chandana R, and Ramachandra (2022), the survey is on YOLO algorithm and its new developments in the area of real-time objects detection. From the two models, YOLO and CNNs are stressed due to their efficiency in achieving general object descriptions with less quantitative loss as compared to the remaining models. Thus, the paper demonstrates how effective implementation of CNN architectures can improve object recognition and solve the various application, including deformity diagnosis and educational applications. The survey provides remarks on the process of developing YOLO, its utility in the spheres of finance and other, as well as concern with feature extraction in specific and focused visual data; methods for identifying targets and features are also discussed.

Table 2.1: Summary of the Journal Review

Author(s) & Year	Tittle	Summary
Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016)	You Only Look Once: Unified, Real-Time Object Detection	The paper under discussion calls YOLO a new approach to object detection that breaks with the need for multiple scanning of the image and unites the detection process with regression. YOLO also directly predicts bounding boxes and class probabilities which make it convenient to be trained end to end and also very fast. The base YOLO model runs at 45 FPS, and even further leveraging, the Fast YOLO model attains 155 FPS, which is quite efficient than most of the real-time detectors in terms of mean average precision. While compared with some state-of-art systems, YOLO has more localization error, it has less false positive from the background and performs well in different scenarios including the natural image domain and artwork image domain.
Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K. and Ghayvat, H. (2021)	CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope	This paper focus on the tremendous development of computer vision especially in image processing due to new developments in Deep Convolutional Neural Networks (CNNs). CNNs are well implemented in areas that include video processing, object recognition and image classification and more due to availability of data and cheap hardware. The paper reviews recent CNN architecture developments, highlighting eight key

		categories: Spatial exploitation, multi-path, depth, breadth, dimension, channel boosting, feature-map exploitation and attention-based CNN. The paper gives a comparative analysis of architectural modifications, innovations, their advantages and limitations and gaps in the existing literature, possible future research, and many applications of CNNs.
Huang, L., Qin, J., Zhou, Y., Fang, Y., Liu, L. and Shao, L. (2023)	Normalization Techniques in Training DNNs: Methodology, Analysis and Application	This paper focuses on the need to enhance the normalization techniques to have faster convergence rates and high level of generalization in Deep Neural Networks (DNNs). For developing the better understanding of the analysed techniques, the paper provides the taxonomy of the normalization methods. It breaks down normalization methods into three components: circulation subdivision, normalization processes and representation restoration. It also reviews latest improvement made to normalization techniques and how such methods applicable towards solving actual world issues.
Viswanatha, V., Chandana R, K. and Ramachandra, A.C. (2022)	Real Time Object Detection System with YOLO and CNN Models: A Review	This paper presents an overview of the YOLO algorithm and its developments in real time object detection. Among others, YOLO and CNN known for their great results in object recognition with less quantitative loss are outlined in the paper. It includes how the appropriate CNN architectures can aid the development of

		object recognition for deformity diagnosis and education. The described survey concerns the peculiarities of the YOLO algorithm, its application in the sphere of finance and other domains, as well as feature extraction and target selection techniques.
He, K., Zhang, X., Ren, S. and Sun, J. (2016)	Deep Residual Learning for Image Recognition.	This article examines Deep Metric Learning, Dlib's ResNet-34 model has a face recognition system that generates 128-dimensional vectors for faces. The comparison of these vectors takes place using Euclidean distance. If two vectors are separated by less than 0.6 then the facial images match in terms of identity. Applied to three million photographs, this technique guarantees precise identification through low distances among individuals and high among dissimilar ones.

CHAPTER 3

METHODOLOGY

3.1 Introduction:

This chapter has the purpose to describe in progressing detail how the system is going to be implemented starting with the planning process and ending with the implementation process. The methodology for this project is structured around four key phases, each integral to the successful development and evaluation of the system: They are: (1) Planning, (2) System Design, (3) Implementation, (4) Evaluation. These phases however are illustrated in the figure below which shows the systematic sequence of development of the project.

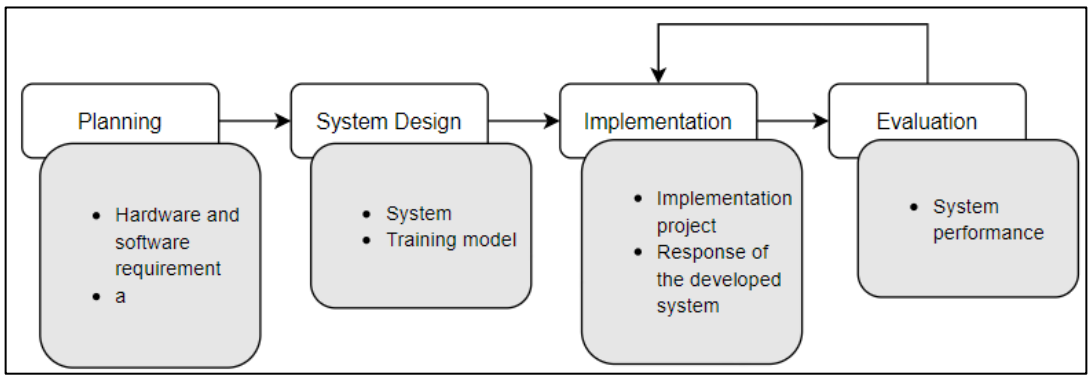


Figure 3.1: The phases of Methodology

3.2 Planning Phase:

One of the key functions within the planning phase is to decide on the specific hardware and application software that would help in the accomplishment of the objectives of the system. This encompasses identification and choosing of appropriate real-time face recognition cameras, identification of the right AI models for implementing face detection as well as face recognition and identification of correct software tools for managing data and tracking attendance. Besides, in the planning phase, a timetable is also developed with stages that include system design, implementation and testing.

3.2.1 Hardware and Software Equipment

Table 3.1 shows the list of equipment that was used for the system. The equipment can be separated into two categories, which is hardware and software. For the hardware, its purpose is to act as a face detection tools to detect the human faces from time to time. For the software side system, it used to do the face recognition if successful recognise then it will go to record to the attendance system.

Table 3.1: Hardware and software equipment selected for the system

Hardware	Software
• Rapoo C280 Webcam 2K HD	• Windows 11 • Roboflow • YOLOv8 • CUDA 11.8.0 • Visual Studio Code • Windows 11 • Excel

3.2.2 Proposed System

When implementing the real-time face recognition system for attending classes, it is crucial to clearly outline certain characteristics and specifications in the system, meeting the needs of a specific educational facility. Starting with the identification of requirements, possible constraints include the number of students that a specific institution enrols as well as the actual physical environment available at the institution. It is also important to define the system's extent, as the system's simpleness and capability are proportional to the variety of data it is expected to process for example, students to be identified at once.

Also, there is a need to define the required level of accuracy and reliability of the attendance tracking system. This helps in maintaining order in the operations of the system in that, it achieves the set goals of the institution without compromising on detail in taking attendance records. The optimal direction must also find a way of reaching high accuracy while at the same time being reasonable enough in terms of error rates to meet operational needs of the institution.

3.2.3 Proposed System Block Diagram

Block diagram of the proposed system as shown in Figure 3.2 below highlight the various processes that take place in the real-time face recognition for attendance tracking. If an input is issued, for instance initiating a class session, the system switches on the camera for visioning and recognition of student faces. The system then maps the identified faces against a registered database of student details for an identity check on people in the classroom.

When the faces are identified to match with the registered data, then the system marks the attendance for the particular student. This attendance data is further stored in an Excel file for the smooth running of the attendance records. The data so saved can then be used to validate absentees as well as determine attendance rates. Furthermore, it will monitor students that have attended less than 80% of lessons and the effectiveness of student attendance and compliance with rules.

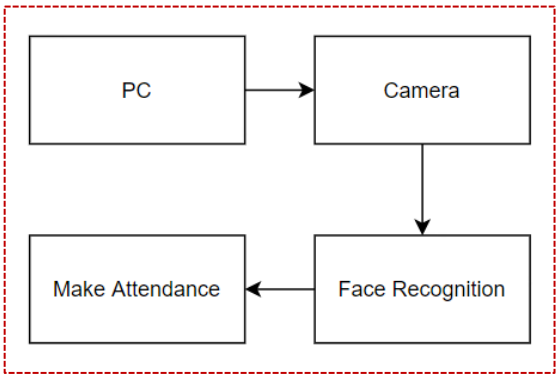


Figure 3.2: Block diagram of the proposed system

3.3 Design Phase

In this phase, most of the work is dedicated to establishing the hardware support, as well as the software requirements for the functioning of the system. The first procedure involves the implementation of the YOLO model to train the system to detect faces of students, to enable it to enrol them into the database. After identifying the faces, features vectors are generated with face encoding methods from each of the given faces. This individual and highly specific identity marker is used as the main differentiator for each of the students. If a face is captured during a class, then the system compares the face encoding to a locally stored database of students' encoding and is then able to mark the student's attendance. Figure 3.3 shows the overview of the process in this phase:

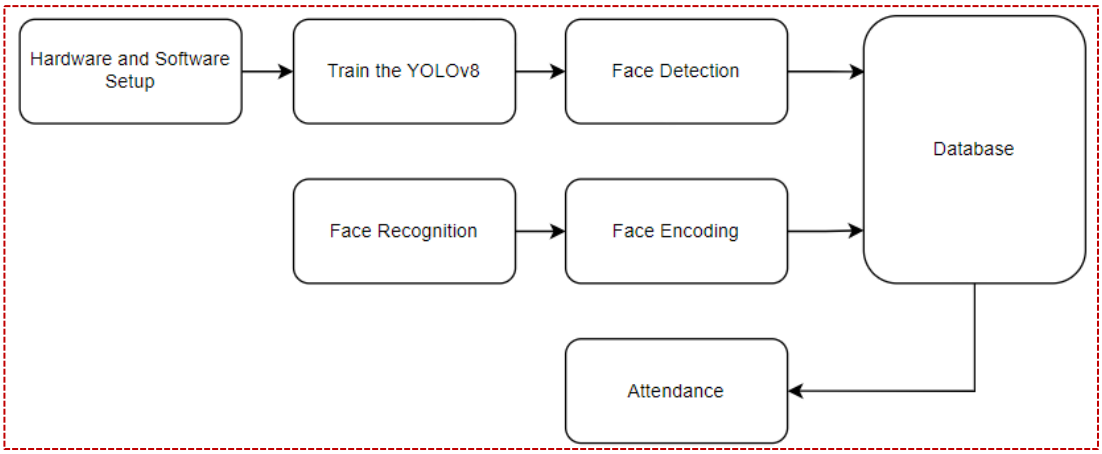


Figure 3.3: Overview of the process in this phase

3.3.1 Hardware and Software Setup

This starts with getting the right camera for an appropriate price that will capture the video in a manner that the faces of the individuals can be easily recognized. When the camera is used and will be placed at a classroom or another relevant location in the course of the experiment and can effectively recognize the people. Also, other basic programs that will enable in setting up of the system in terms of programming must also be downloaded. For example, by employing an electronic integrated environment system (IDE) like Visual Studio Code (VSCode), effective coding for the project can be made so that the software parts of the system are effective and harmoniously integrate with the hardware.

3.3.2 Face Detection and Recognition

This function can be considered as the core of the real-time attendance. The process starts with the training of a YOLO model with respect to face detection, which enables the system to detect and locate faces in the live video stream. Otherwise, pre-trained YOLO models can be employed to crop out the faces of the students, which would in turn be used to compile a vast database of facial images of the students. This dataset is complemented by respective names and IDs to create a database without which identification cannot be precise.

The next step crucial after defining the dataset is to extract an identity vector for all the face using a face encoding. This involves using dlib which drawn out from the Resnet architecture to obtain a face encoding vector for every student. These face encodings are used to represent the students' face in the database based on their facial landmarks. The last of these is to match given face encoding to the database of face encodings, allowing the system to recognise or in fact authenticate the person in real time for making attendance records accurate.

3.3.3 Attendance Logging

The attendance logging procedure therefore ensures that the face recognition system works hand in hand with the attendance software to offer between fully automated system of documenting the attendance records. In cases where a student face is detected and their identity is recognized, the system records their attendance even to the minute. The considered system has to work in rather complicated conditions, for example, the identification of more student faces during the lecture's record and guarantee that facial recognition algorithms will not only point to each of the students in the lecture, but also link it to their profiles. If, after perusing through every inch of the face, the face cannot be recognized, then the face is tagged with the "Unknown" so that no confusion arises in the attendance record. Further, the system also determines the accumulated total time for the student based on the attendance in order to generate the efficient way of tracking the student presence in the class in the long run.

3.3.4 Training YOLO Model

Below is a figure that gives a feel of this training process, demarcating the stages of preparing the model for deployment.

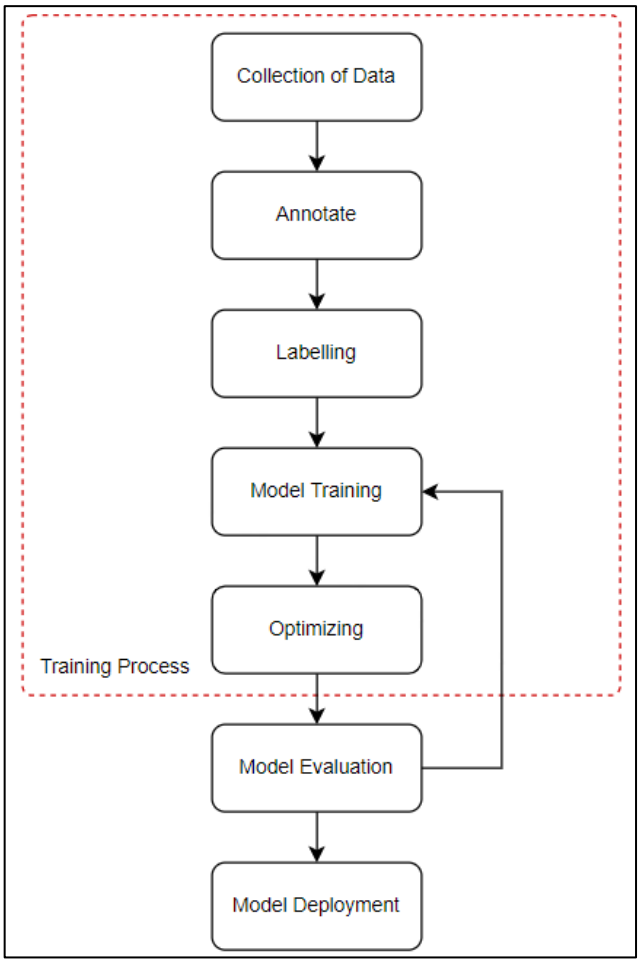


Figure 3.4: Overview of training YOLOv8 for face detection

The training of the YOLOv8 model consists of exposing the model to a dataset that has been chosen with caution to train the model on what it is expected to identify in the imported data. This process is important in face detection and recognition in real-time operations that the system may be exposed to. The training data usually comprises of images labelled according to the face classes, helping the model learn the distinctions between various faces. This way as the model begins to take in more data its detection capability increases making face recognition to be more accurate.

3.4 Implement Phase

This section describes the whole detection, recognition and logging attendance systems. Figure 3.5 shows the overview of the implementation phase of this project.

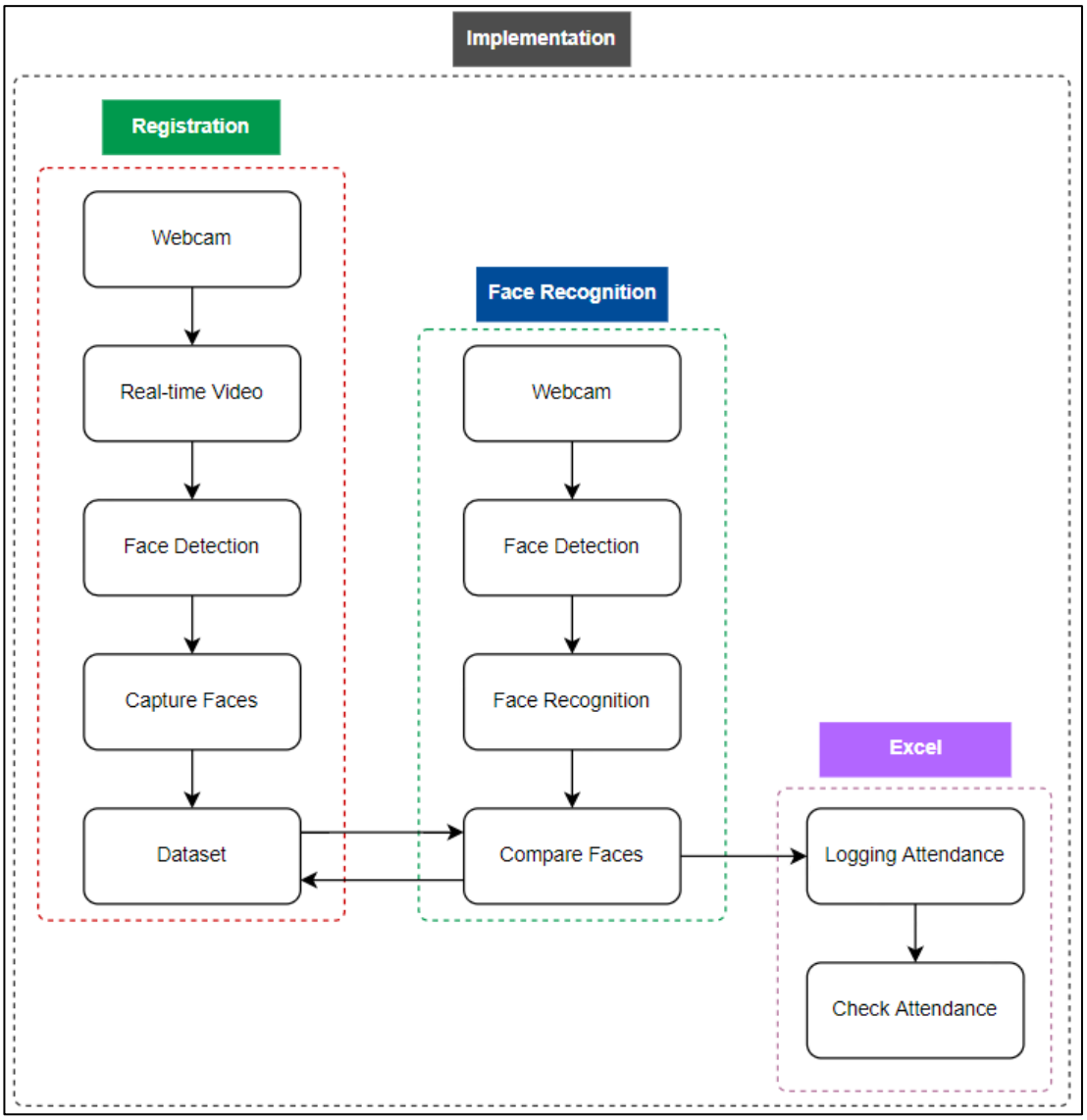


Figure 3.5: Overview of Implementation Phase

After the YOLO model has been successfully trained, the system proceeds to the next crucial stages; Registering the Faces, Real-time face detection and logging the attendance into the Excel sheet. In the Registration phase, it allows opening the webcam to capture human faces using pre-trained YOLOv8 model for the face detection. In the process of recording, the camera takes photos of the student faces and even if one student's photo is distorted, the camera takes a new photo and stores in a separate dataset, so that a clear picture of faces is taken for subsequent recognition. This process is crucial in the development of the facial database because allows the system to recognize each students' face in subsequent class sessions.

This is followed by the Real-time Face Recognition where the system opens the webcam again to cover for the students' faces during the class. With the help of YOLO pre-trained model, which has real time face detection, the system detects all the faces present in the frame. After detection, each face is converted to identity vector where it is matched against the database of student faces acquired during the registration process. If the match is found, then it knows who the student is and then logs his/her presence in real time. This makes the process smooth such that no human intervention is required for the taking of attendance by the students themselves or instructors.

Last of all, the system proceeds to the Attendance Logging phase in which the recognized students' attendance is recorded in an Excel sheet. This is to indicate whether or not the student was present in the session and give him/her a timestamp to indicate that he/she attended the session. In addition, the system also takes into account students' attendance percentage for the particular class session that he/she was present or absent. This percentage is then used to determine how the student has done in total over multiple class sessions to offer the most complete and intricate report on the student's attendance. Through the implementation of these processes, the system not only helps in tracking of attendance but also improves on the quality, speed and reliability of the class processes.

3.5 Analysis Phase

This phase signifies the last steps of the methodology and is significant in identifying the performance of the system. It requires the determination of the best way through which the developed system runs effectively in addressing the objectives of the project. Through this type of performance evaluation, there will be confirmation that the system works correctly and is effective in achieving the intended objectives within the areas of accuracy, credibility, and efficiency. This step is crucial in evaluating the success of the system and to look out for areas that may need to be developed in the future.

3.6 Project Management

The Table 3.2 shows the Gantt Chart of Final Year Project 1, whereas Table 3.3 shows the Gantt Chart of Final Year Project 2.

Table 3.2: Gantt Chart for Final Year Project 1

Task	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project Tittle Decision														
Planning														
Introduction														
Literature Review														
Methodology														
Initialize Hardware & Software														
Training Model														

Table 3.3: Gantt Chart for Final Year Project 1

Task	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Face Detection														
Face Registration														
Create Dataset Format														
Build Real-Time Code														
Encoding Faces														
Create Excel to Record Attendance														
Debugging & Testing														
Result														
Discussion														
Conclusion & Recommendation														

3.7 Cost Estimation

This section focuses at estimating costs that are likely to be incurred to realize the system at the end of the project. The main purpose of this section is to estimate, the overall costs needed for developing and implementing the system. This way setting a financial framework helps in controlling and managing resources and preventing future over expenditure on project. The estimated fund required to build the project is RM 118.48.

Table 3.4: Cost Estimation of Project Materials

Item	Cost (RM)
Hardware:	
Rapoo C280 Webcam 2K HD	118.48
Total Estimated Cost	118.48

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This section of the project describes the results of the experiment and a discussion of different development stages. They involve image collection, face detected results, hardware configuration, model training and performance, encoding of faces and lastly logging attendance into an Excel sheet and check the overall attendance. At each level the results are used for evaluation of the system and its accuracy.

4.2 Experimental Results

In this section, the results of the developed system are thoroughly discussed, focusing on the performance of YOLOv8 for face detection and dlib for face encoding, comparison and matching. The system successfully recognized different student faces in real-time for the attendance tracking system, with all data being logged to an Excel sheet.

Additionally, the process of training the face detection model is explained, highlighting how the system compares real-time detections with the pre-collected student face dataset. This enables accurate classification and identification of students during live detection for efficient attendance management.

4.2.1 Image Collection for Registration

```
Please Enter Your Course Name (e.g., UGEB3016 Final Year Project 1): UGEB3016 Final Year Project 1
Course UGEB3016 Final Year Project 1 already exists. Adding new student to this course.
Please Enter Your Name: student a
Name corrected to: Student A
Please Enter Your Student ID: 20AAA12345
Wait to start capturing your face...
```

Figure 4.1: Student registration

Figure 4.1 describes the flow of student registration. Firstly, the student types the name of the course he or she wants to enrol in, then the student's full name along with their student ID. If it is written in wrong format such as having a small letter in the beginning of each word like for example "kuak xuan ren" the system will study and convert it to right format "Kuak Xuan Ren" respectively for the student ID format it should be within this format "01ABC12345".

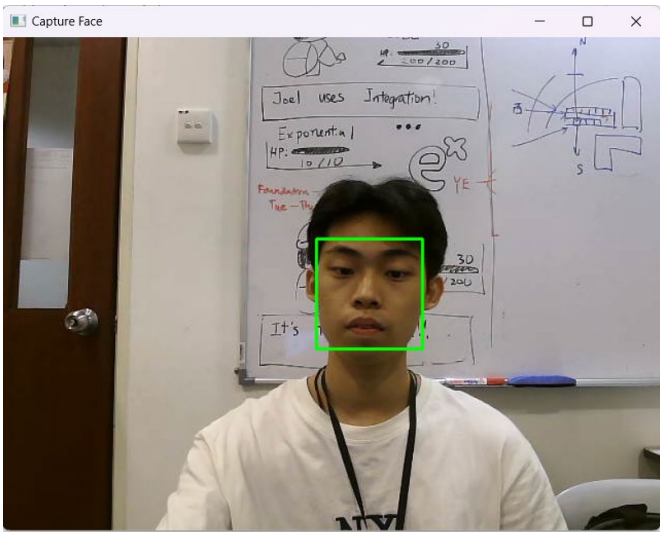


Figure 4.2: Image Collecting Process

Figure 4.2 illustrates the ways through which the face data is collected by the system. The student's face is then taken and saved in a file under the particular course name that the student entered during the time of registration. This guarantees the storage of each student face data and assignment of it to the correct course to be used later in the attendance tracking system.

Next, the figures below illustrate how the students' collected faces are stored in folders, with their name and student ID labelled. When collecting the face dataset of a student, it first prompts the name of the course to which their dataset should be put under-in other words, the course folder and then the dataset folder of their individual face images.

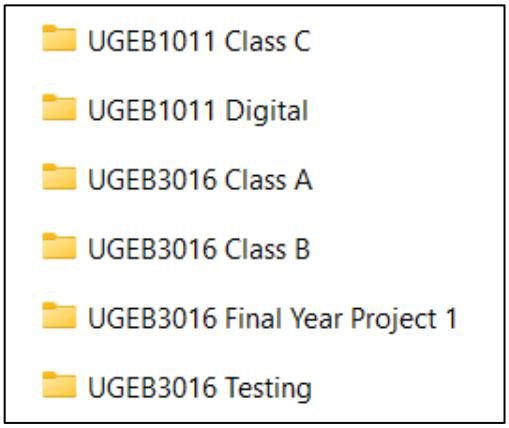


Figure 4.3: The courses folder

Figure 4.4: The dataset of the students

Inside each course folder that in Figure 4.4, the student's face datasets are stored, with each course containing its own set of student-specific datasets. This can make sure that the face dataset is grouped according to the course.

```
[
    {
        "name": "Kuak Xuan Ren",
        "student_id": "21AGB06806"
    },
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Chew You Hui",
        "student_id": "20ADB04796"
    },
    {
        "name": "Boey Zhi Xuan",
        "student_id": "20AGB05745"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    }
]
```


Figure 4.5: The json file style

The Figure 4.5 shows that when the students register their faces, the program not only stores the images in correspondent folders but also saves all the students' data in JSON format. It contains basic information on the student including their name, student ID and the courses which they are registered for. With the help of JSON, the system structurally facilitates the problems of searching and processing of the field values containing student information for each course during the real-time detection. This approach leads to improved approaches to data access and integration, which can effectively combine the face recognition results with the right students' record for attendance tracking and reporting.

4.2.2 Face Detection

This section provides all the detected faces of the students within the class and for each detection, the student's name, student ID, and the confidence level of the detection on top of the bounding box surrounding the face.

4.2.2.1 Face Detection in Class

The below figures indicate the result having been captured from 6 classes these include UGEB3016 Final Year Project, UGEB3016 Testing, UGEB1011 Digital, UGEB3016 Class A, UGEB3016 Class B, and UGEB1011 Class C relying on the reflection of real-time detection result as recorded during the session.

UGEB3016 Final Year Project:

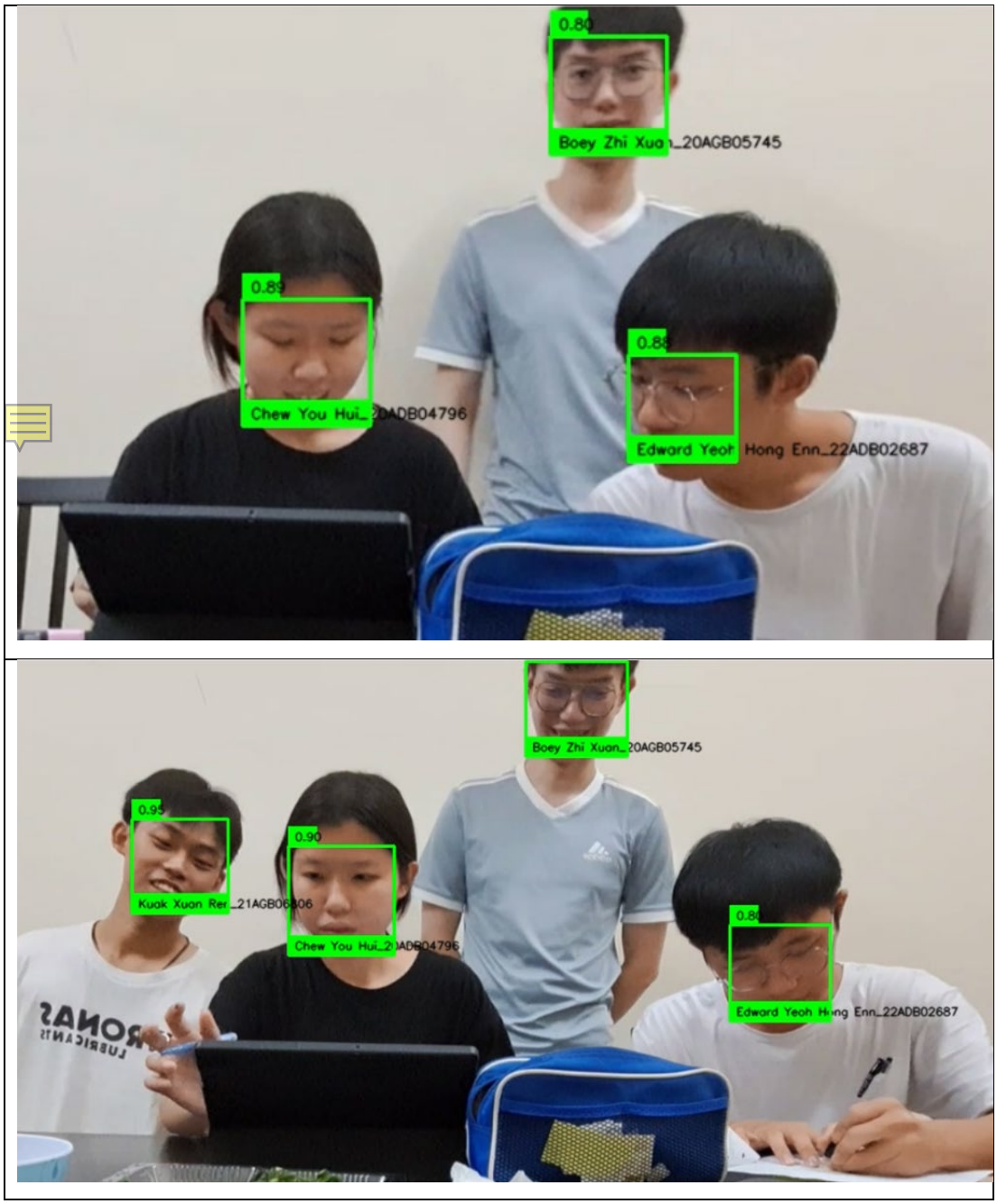


Figure 4.6: Two class for UGEB3016 Final Year Project

UGEB3016 Testing:

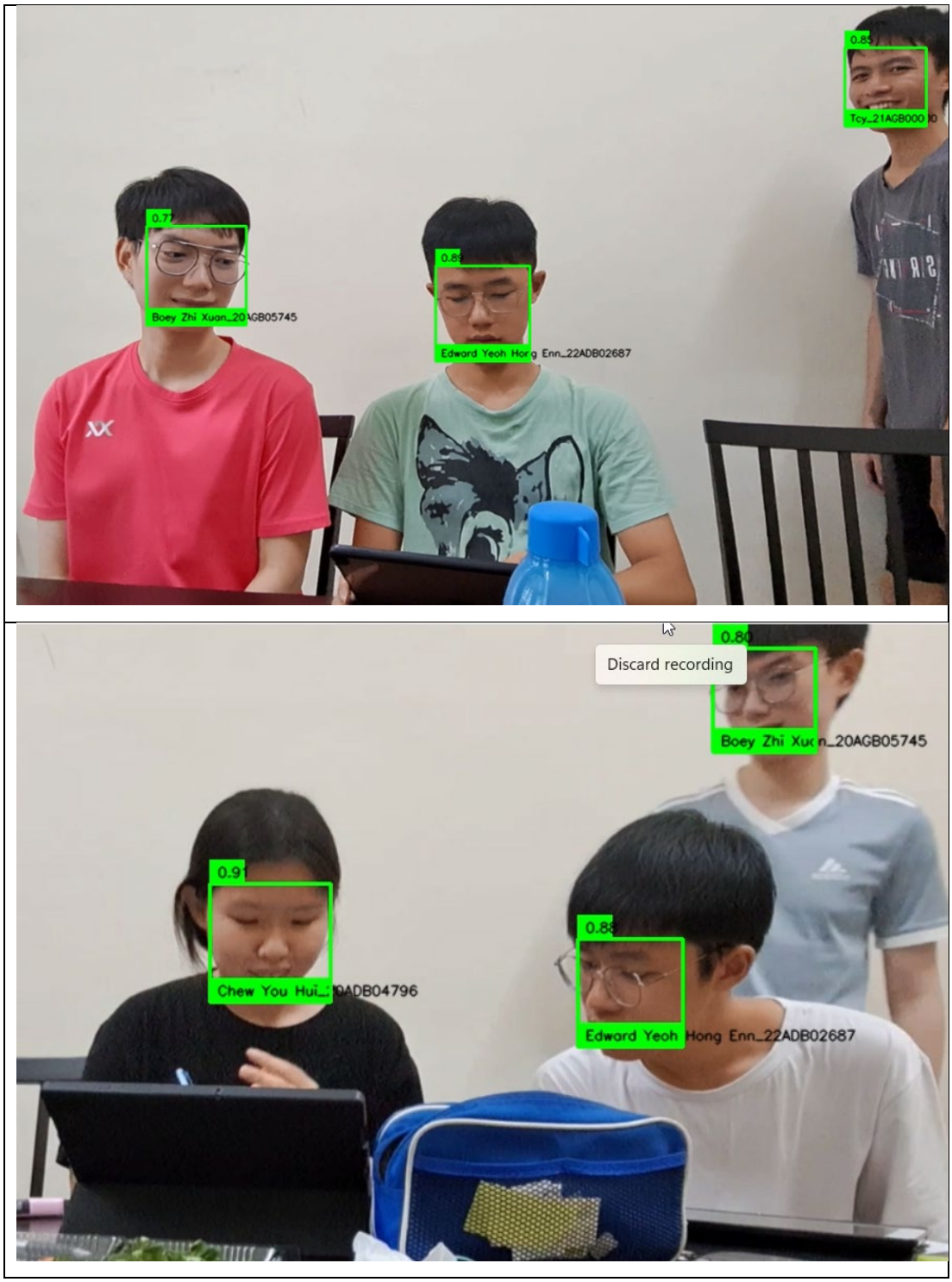


Figure 4.7: Two class for UGEB3016 Testing

UGEB1011 Digital:

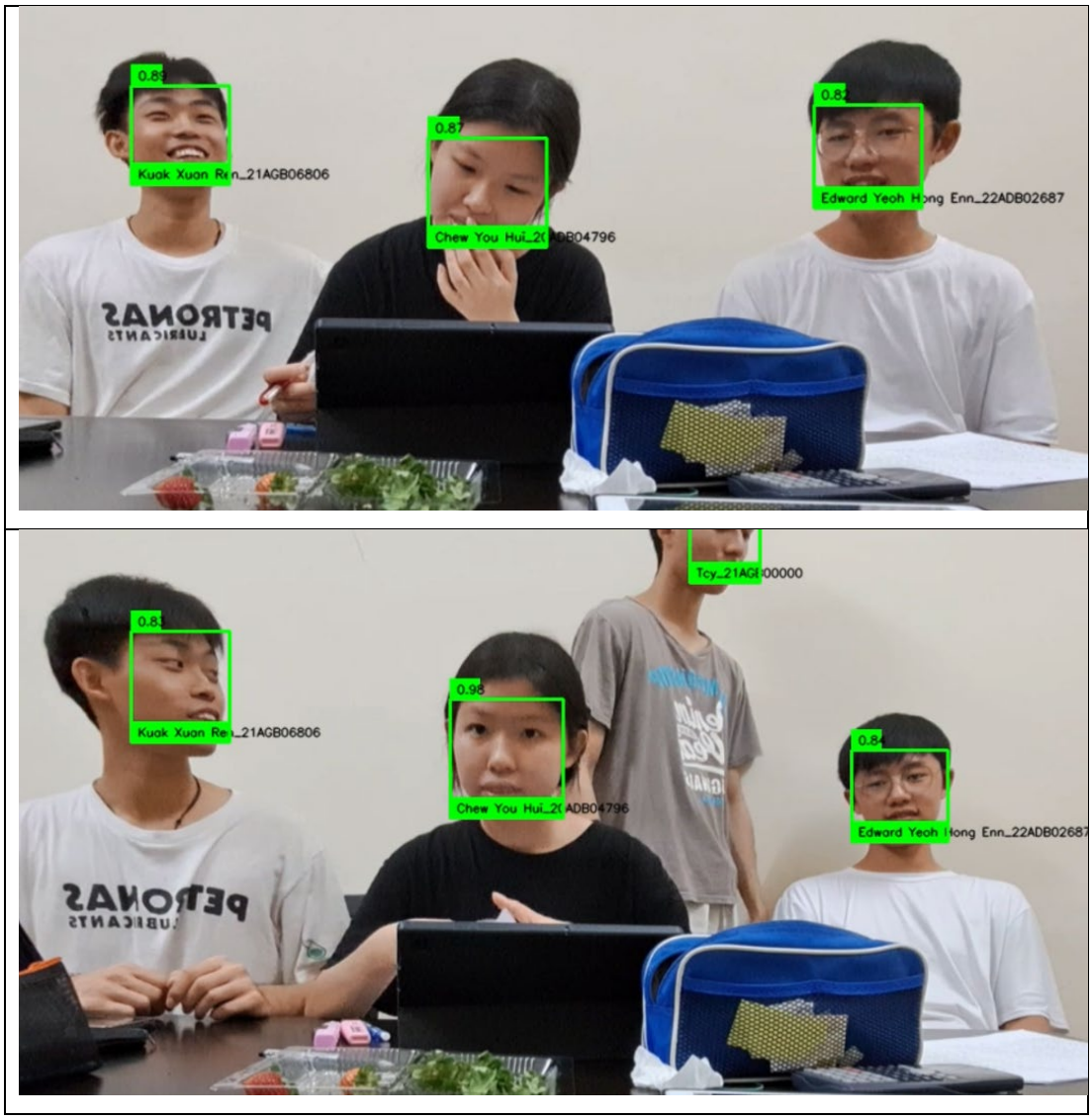


Figure 4.8: Two class for UGEB1011 Digital

UGEB3016 Class A:

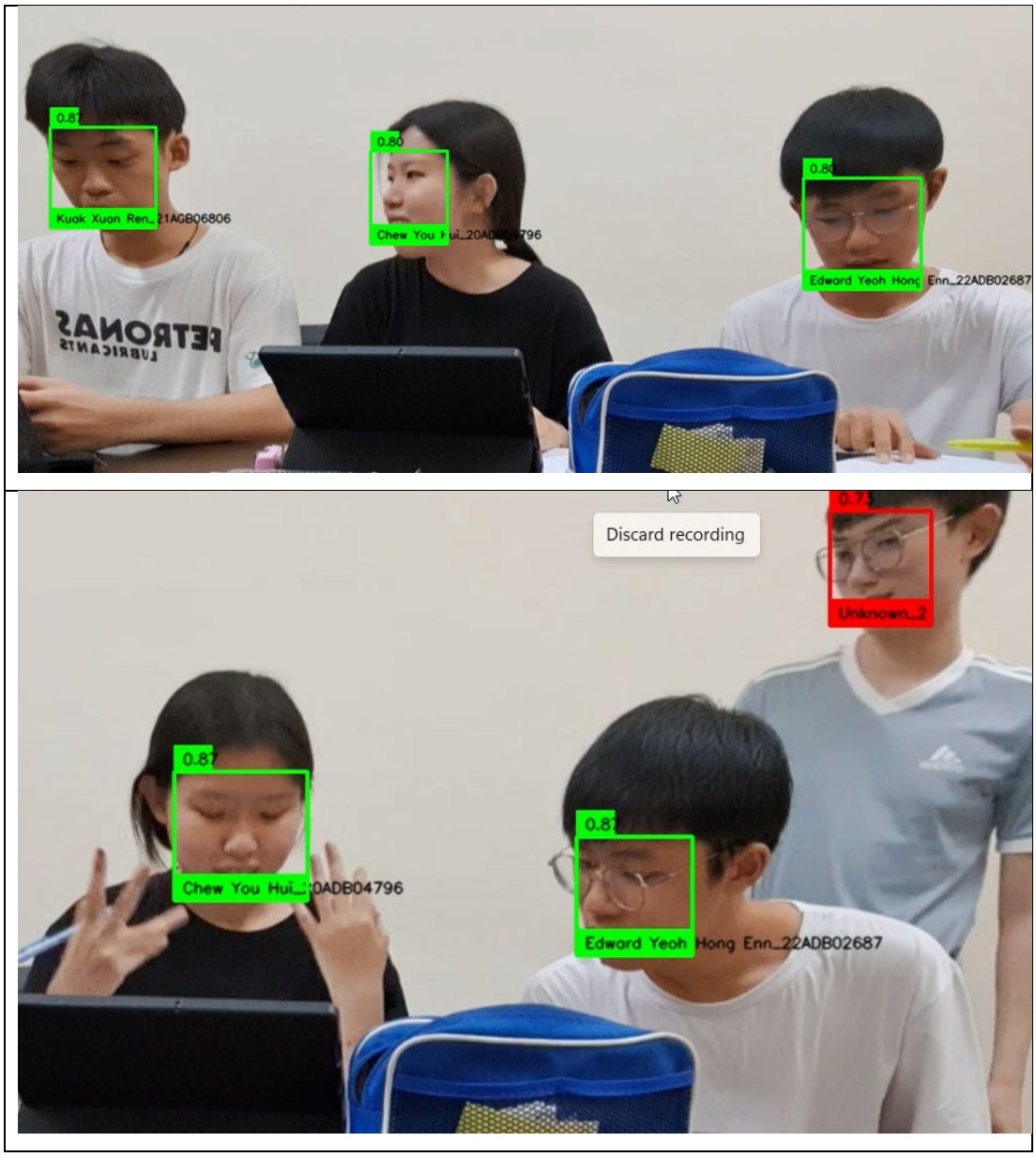


Figure 4.9: Two class for UGEB3016 Class A

UGEB3016 Class B:

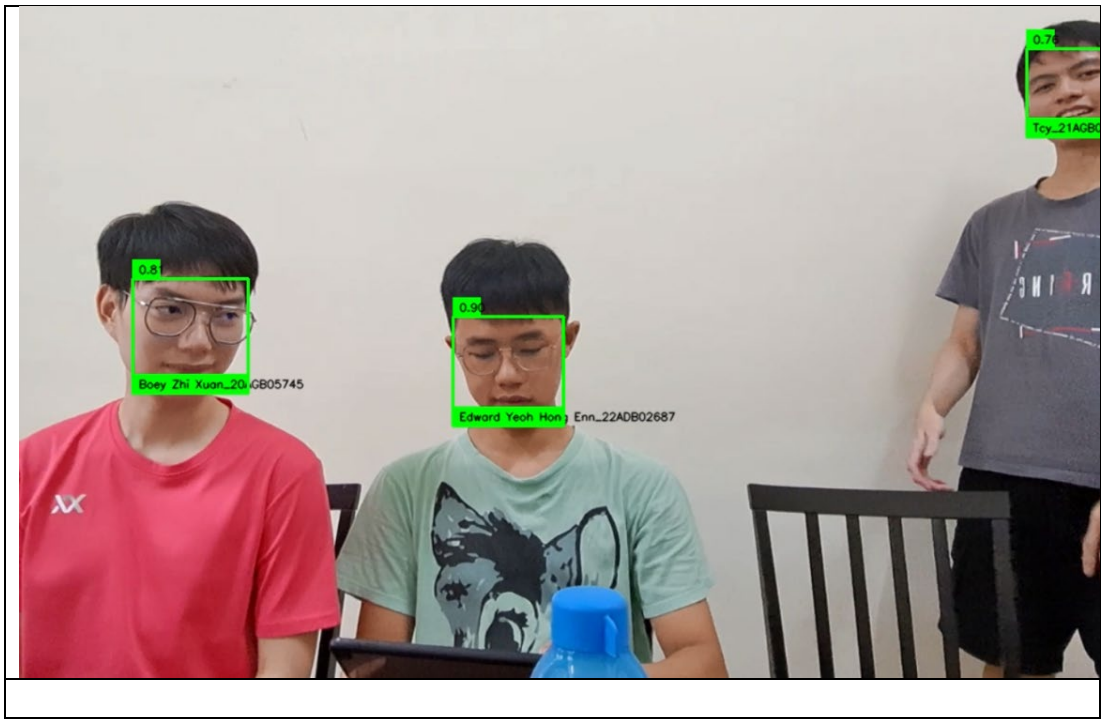
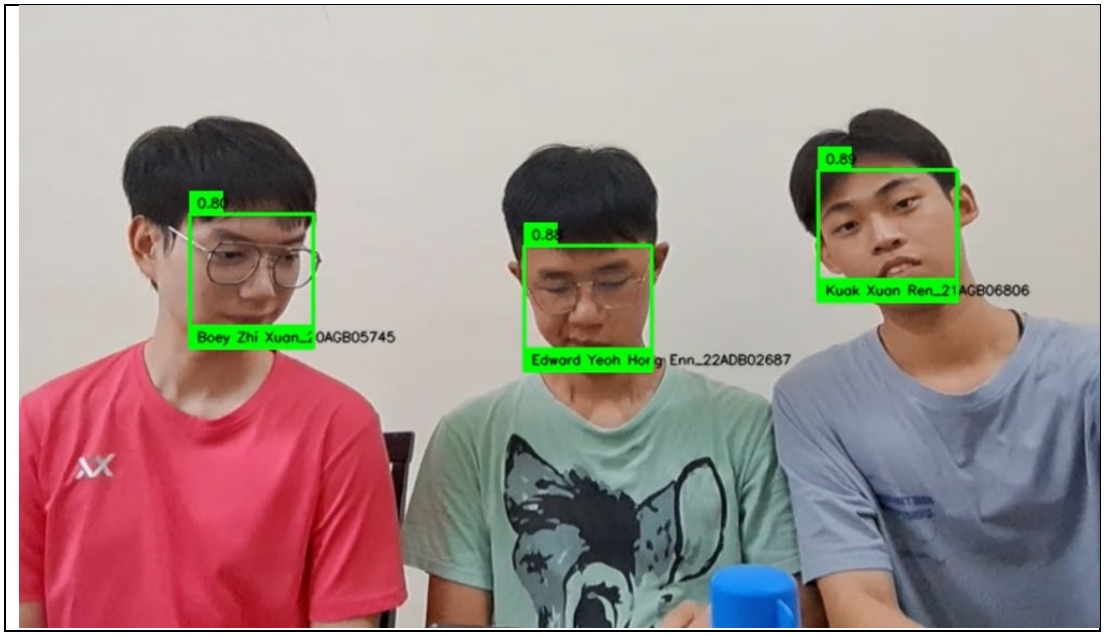


Figure 4.10: One class for UGEB3016 Class B

UGEB1011 Class C:



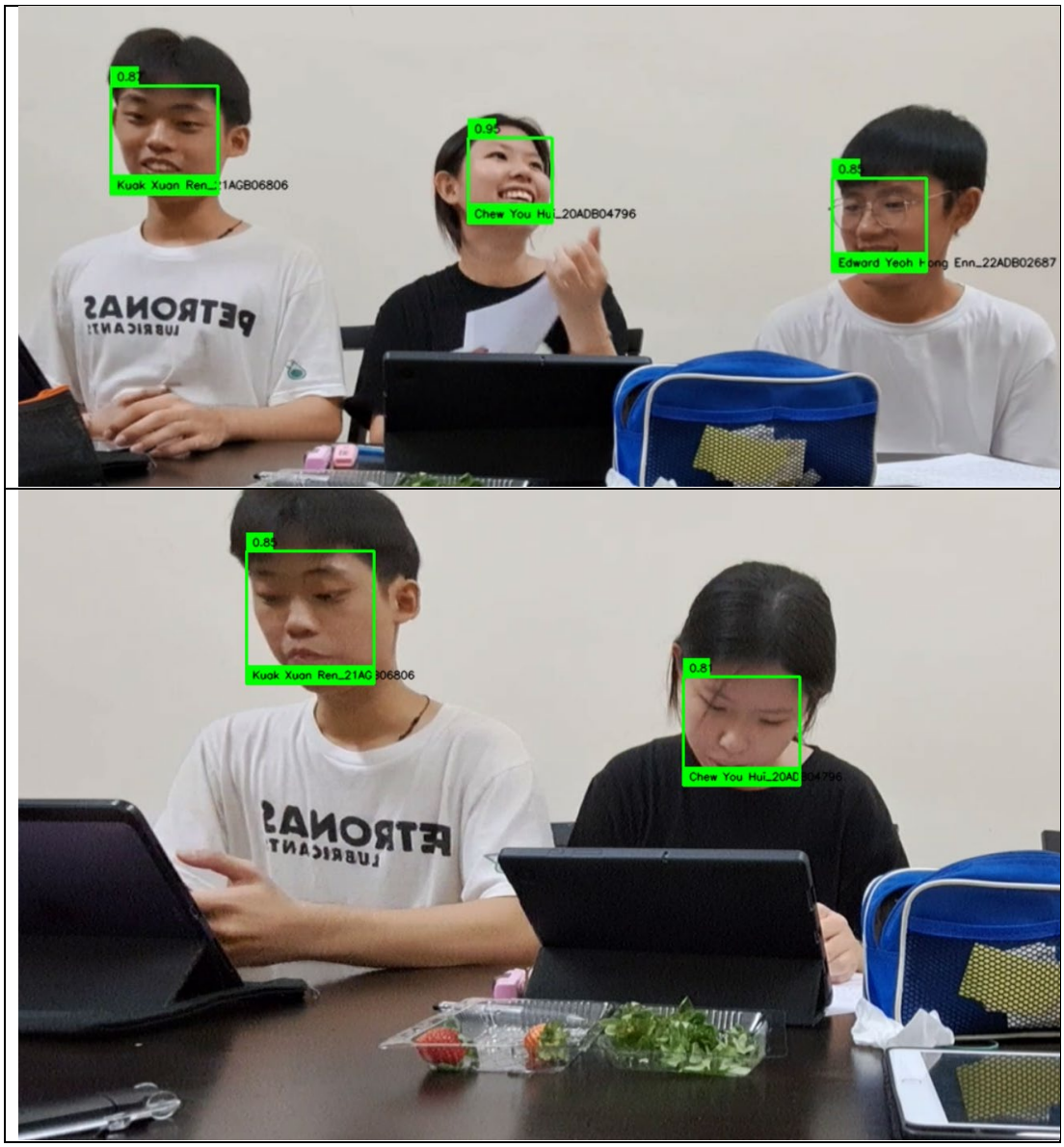


Figure 4.11: Three class for UGEB1011 Class C

Figures 4.12 to 4.16 show the confidence level captured by the face recognition system for each student, but in the form of graphs instead of text logs. It can be seen from the above graphs that for all students an average of each mark weighs above 80% and therefore the results show a high accuracy of the system.

In Figure 4.12, student Chew You Hui (ID: 20ADB04796) had an average confidence score of 0.89 this was the highest among all participants showing the high accuracy of this system. Figure 4.13 the student's name Boey Zhi Xuan (ID: 20AGB05745), whose average confidence level is 0.80, potentially influenced by lighting conditions during face capture. In Figure 4.14, Kuak Xuan Ren (ID：21AGB06806), figure registered an average of 0.88 and once again proving that the developed system is quite accurate.

Meanwhile, Figure 4.15 shows Edward Yeoh Hong Enn (ID: 22ADB02687) with an average confidence level of 0.85, which maintains a high standard of detection accuracy. Lastly, for the Figure 4.16 student Tcy (ID: 21AGB00000), who averaged 0.80. Although the dataset for this student is smaller, as he attended only three classes, his confidence level for the first class reached 0.85. However, in the other two classes, the confidence levels were slightly lower at 0.79 and 0.76.

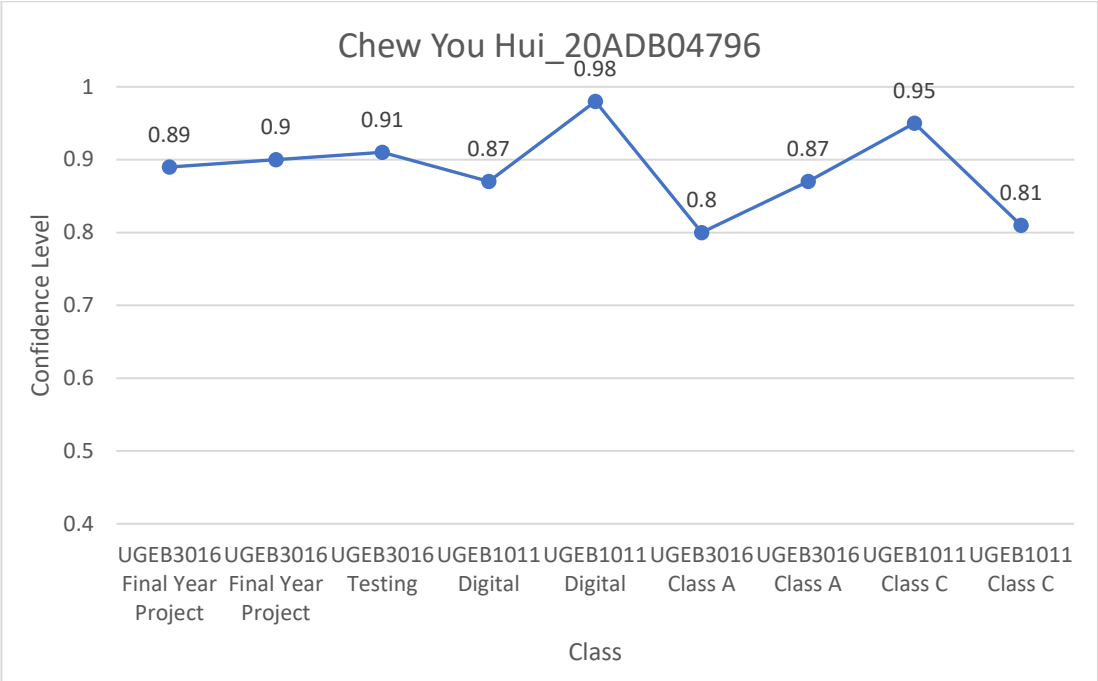


Figure 4.12: Confidence level for Chew You Hui_20ADB04796

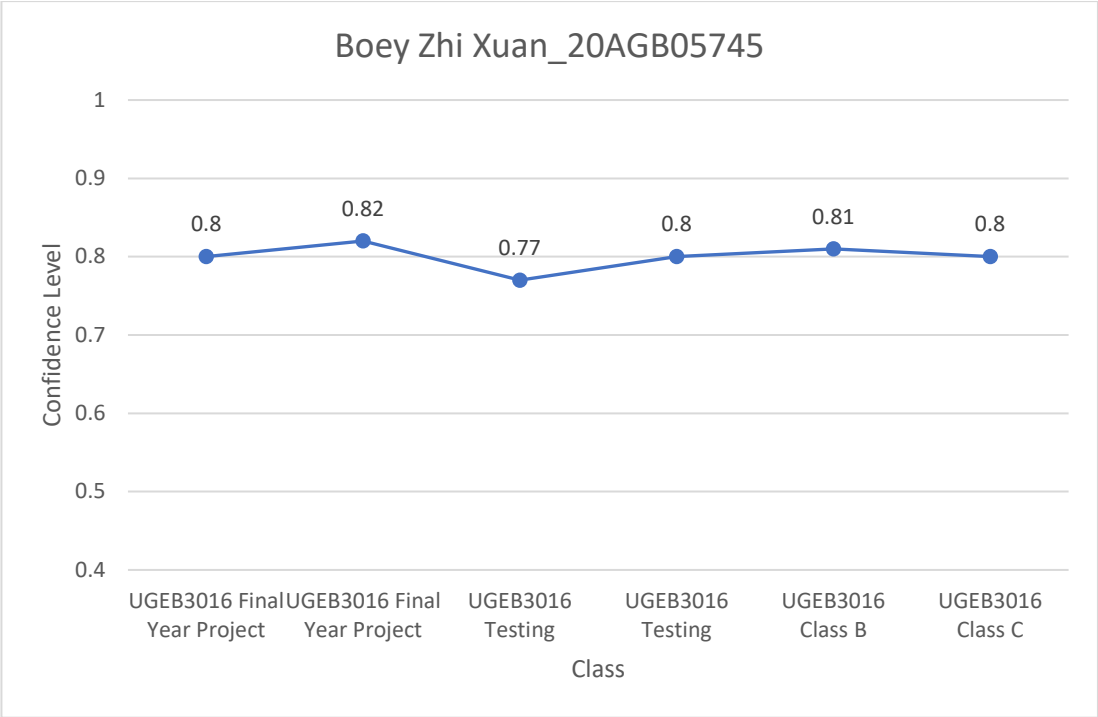


Figure 4.13: Confidence level for Boey Zhi Xuan_20AGB05745

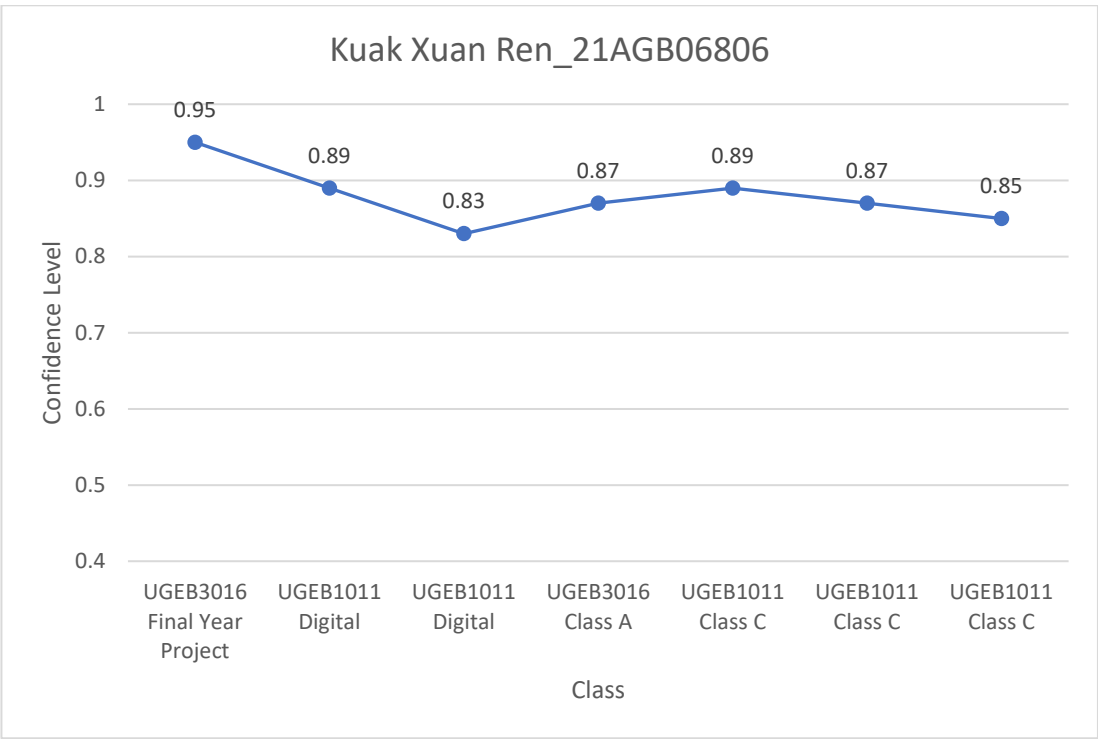


Figure 4.14: Confidence level for Kuak Xuan Ren_21AGB06806

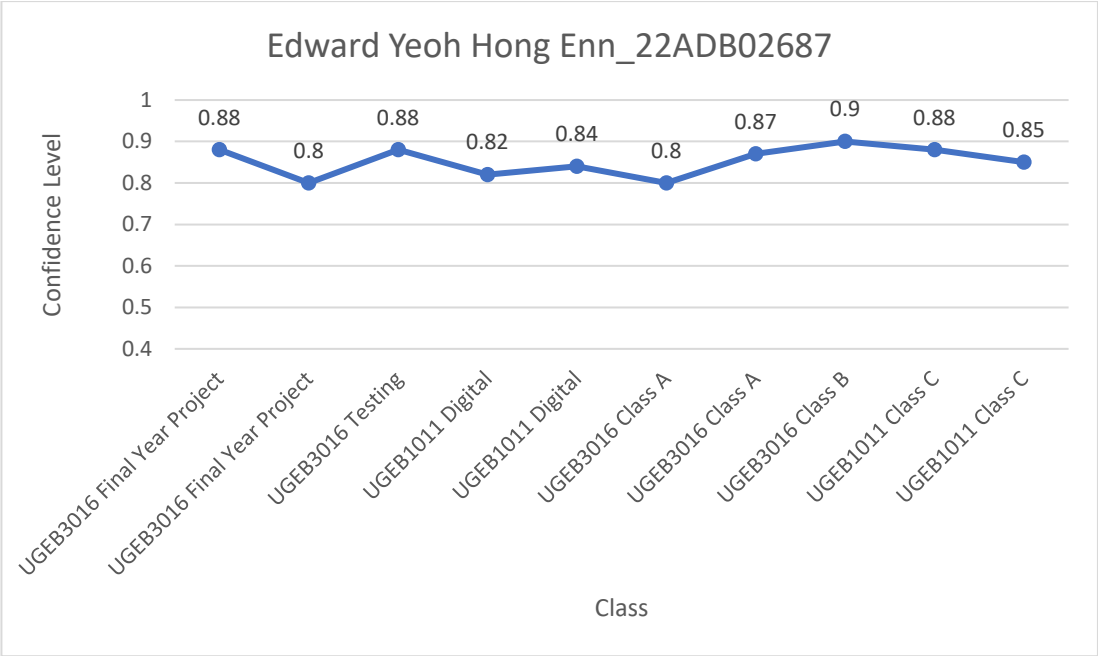


Figure 4.15: Confidence level for Edward Yeoh Hong Enn_22ADB02687

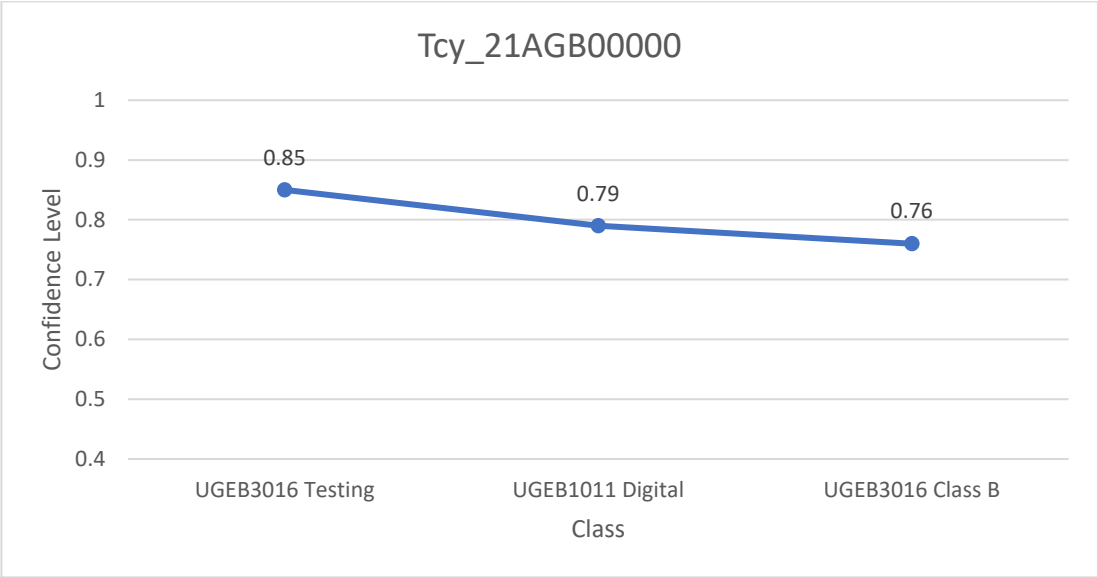


Figure 4.16: Confidence level for Tcy_21AGB00000

4.2.2.2 The JSON and Excel result for each class

The following figures shows the JSON file storing information related to student for each class, along with the attendance details. When a name and ID of a student are specified in the JSON file, but the face of the concerned student is not captured in the session, it will be marked as “Absent”, and the concerned cell in the sheet will be in red colour. Also, the “Status” column will indicate if a student has attendance below the 80% which will easily identify students with attendance problems.

UGEB3016 Final Year Project:


```
[
    {
        "name": "Kuak Xuan Ren",
        "student_id": "21AGB06806"
    },
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Chew You Hui",
        "student_id": "20ADB04796"
    },
    {
        "name": "Boey Zhi Xuan",
        "student_id": "20AGB05745"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    }
]
```


Figure 4.17: JSON file for UGEB3016 Final Year Project

Date:	2024-09-01								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:20:00	Kuak Xuan Ren	21AGB06806	N/A	N/A	N/A	N/A	Absent	N/A
2	0:20:00	Edward Yeoh Hong Enn	22ADB02687	10:02 AM	10:19 AM	0:16:23.249341	81.94%	Present	N/A
3	0:20:00	Chew You Hui	20ADB04796	10:02 AM	10:20 AM	0:17:52.168021	89.35%	Present	N/A
4	0:20:00	Boey Zhi Xuan	20AGB05745	10:02 AM	10:14 AM	0:11:39.574302	58.30%	Present	Attendance below 80%
5	0:20:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A
Date:	2024-09-03								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:20:00	Kuak Xuan Ren	21AGB06806	04:30 AM	04:39 AM	0:09:00.458909	45.04%	Present	Attendance below 80%
2	0:20:00	Edward Yeoh Hong Enn	22ADB02687	04:20 AM	04:38 AM	0:18:34.264394	92.86%	Present	N/A
3	0:20:00	Chew You Hui	20ADB04796	04:20 AM	04:39 AM	0:19:44.120861	98.68%	Present	N/A
4	0:20:00	Boey Zhi Xuan	20AGB05745	04:22 AM	04:39 AM	0:17:39.802925	88.32%	Present	N/A
5	0:20:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A

Figure 4.18: The Attendance of students for UGEB3016 Final Year Project

UGEB3016 Testing:

```
[
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Chew You Hui",
        "student_id": "20ADB04796"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    },
    {
        "name": "Boey Zhi Xuan",
        "student_id": "20AGB05745"
    }
]
```

Figure 4.19: JSON format for UGEB3016 Testing

Date:	2024-08-30								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:15:00	Edward Yeoh Hong Enn	22ADB02687	08:32 AM	08:46 AM	0:14:37.845091	97.54%	Present	N/A
2	0:15:00	Chew You Hui	20ADB04796	N/A	N/A	N/A	N/A	Absent	N/A
3	0:15:00	Tcy	21AGB00000	08:35 AM	08:44 AM	0:08:53.940253	59.33%	Present	Attendance below 80%
4	0:15:00	Boey Zhi Xuan	20AGB05745	08:32 AM	08:46 AM	0:14:35.779055	97.31%	Present	N/A
Date:	2024-09-03								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Edward Yeoh Hong Enn	22ADB02687	02:44 AM	02:53 AM	0:09:47.809489	97.97%	Present	N/A
2	0:10:00	Chew You Hui	20ADB04796	02:44 AM	02:52 AM	0:08:39.735241	86.62%	Present	N/A
3	0:10:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A
4	0:10:00	Boey Zhi Xuan	20AGB05745	02:44 AM	02:53 AM	0:09:47.799480	97.97%	Present	N/A

Figure 4.20: The Attendance of students for UGEB3016 Testing

UGEB1011 Digital:


```
[
    {
        "name": "Kuak Xuan Ren",
        "student_id": "21AGB06806"
    },
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Chew You Hui",
        "student_id": "20ADB04796"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    }
]
```


Figure 4.21: JSON format for UGEB1011 Digital

Date:	2024-08-29								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Kuak Xuan Ren	21AGB06806	01:10 PM	01:19 PM	0:09:49.210596	98.20%	Present	N/A
2	0:10:00	Edward Yeoh Hong Enn	22ADB02687	01:10 PM	01:19 PM	0:09:17.784012	92.96%	Present	N/A
3	0:10:00	Chew You Hui	20ADB04796	01:10 PM	01:19 PM	0:09:24.312407	94.05%	Present	N/A
4	0:10:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A
Date:	2024-09-02								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:15:00	Kuak Xuan Ren	21AGB06806	03:10 AM	03:24 AM	0:14:19.620110	95.51%	Present	N/A
2	0:15:00	Edward Yeoh Hong Enn	22ADB02687	03:09 AM	03:24 AM	0:14:59.086176	99.90%	Present	N/A
3	0:15:00	Chew You Hui	20ADB04796	03:09 AM	03:23 AM	0:14:28.136512	96.46%	Present	N/A
4	0:15:00	Tcy	21AGB00000	03:17 AM	03:21 AM	0:04:15.314039	28.37%	Present	Attendance below 80%

Figure 4.22: The Attendance of students for UGEB1011 Digital

UGEB3016 Class A:


```
[
    {
        "name": "Kuak Xuan Ren",
        "student_id": "21AGB06806"
    },
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Chew You Hui",
        "student_id": "20ADB04796"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    }
]
```


Figure 4.23: JSON format for UGEB3016 Class A

Date:	2024-08-28								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Kuak Xuan Ren	21AGB06806	08:13 AM	08:22 AM	0:09:27.140253	94.52%	Present	N/A
2	0:10:00	Edward Yeoh Hong Enn	22ADB02687	08:13 AM	08:21 AM	0:08:57.102754	89.52%	Present	N/A
3	0:10:00	Chew You Hui	20ADB04796	08:13 AM	08:22AM	0:09:32.542601	95.42%	Present	N/A
4	0:10:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A
Date:	2024-09-03								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Kuak Xuan Ren	21AGB06806	04:56 AM	05:05 AM	0:09:21.579445	93.60%	Present	N/A
2	0:10:00	Edward Yeoh Hong Enn	22ADB02687	04:56 AM	05:03 AM	0:07:16.560218	72.76%	Present	Attendance below 80%
3	0:10:00	Chew You Hui	20ADB04796	04:56 AM	05:05 AM	0:09:40.576450	96.76%	Present	N/A
4	0:10:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A

Figure 4.24: The Attendance of students for UGEB3016 Class A

UGEB3016 Class B:


```
[
    {
        "name": "Kuak Xuan Ren",
        "student_id": "21AGB06806"
    },
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    },
    {
        "name": "Boey Zhi Xuan",
        "student_id": "20AGB05745"
    }
]
```


Figure 4.25: JSON format for UGEB3016 Class B

Date:	2024-09-03								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Kuak Xuan Ren	21AGB06806	N/A	N/A	N/A	N/A	Absent	N/A
2	0:10:00	Edward Yeoh Hong Enn	22ADB02687	05:09 AM	05:19 AM	0:09:58.925533	99.82%	Present	N/A
3	0:10:00	Tcy	21AGB00000	05:11 AM	05:13 AM	0:02:18.829783	23.14%	Present	Attendance below 80%
4	0:10:00	Boey Zhi Xuan	20AGB05745	05:09 AM	05:19 AM	0:09:58.916289	99.82%	Present	N/A

Figure 4.26: The Attendance of students for UGEB3016 Class B

UGEB1011 Class C:


```
[
    {
        "name": "Kuak Xuan Ren",
        "student_id": "21AGB06806"
    },
    {
        "name": "Edward Yeoh Hong Enn",
        "student_id": "22ADB02687"
    },
    {
        "name": "Chew You Hui",
        "student_id": "20ADB04796"
    },
    {
        "name": "Boey Zhi Xuan",
        "student_id": "20AGB05745"
    },
    {
        "name": "Tcy",
        "student_id": "21AGB00000"
    }
]
```


Figure 4.27: JSON format for UGEB1011 Class C

Date:	2024-08-27								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Kuak Xuan Ren	21AGB06806	05:33 AM	05:41 AM	0:08:53.075785	88.85%	Present	N/A
2	0:10:00	Edward Yeoh Hong Enn	22ADB02687	05:32 AM	05:42 AM	0:09:59.209701	99.87%	Present	N/A
3	0:10:00	Chew You Hui	20ADB04796	N/A	N/A	N/A	N/A	Absent	N/A
4	0:10:00	Boey Zhi Xuan	20AGB05745	05:32 AM	05:42 AM	0:09:43.946505	97.32%	Present	N/A
5	0:10:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A
Date:	2024-09-02								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:20:00	Kuak Xuan Ren	21AGB06806	10:15 AM	10:30 AM	0:14:13.701246	71.14%	Present	Attendance below 80%
2	0:20:00	Edward Yeoh Hong Enn	22ADB02687	10:11 AM	10:30 AM	0:19:23.084213	96.92%	Present	N/A
3	0:20:00	Chew You Hui	20ADB04796	10:11 AM	10:30 AM	0:19:48.780246	99.07%	Present	N/A
4	0:20:00	Boey Zhi Xuan	20AGB05745	N/A	N/A	N/A	N/A	Absent	N/A
5	0:20:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A
Date:	2024-09-03								
No.	Class Duration	Student Name	Student ID	Sign-in Time	Sign-out Time	Duration	Total Time	Attendance	Status
1	0:10:00	Kuak Xuan Ren	21AGB06806	09:23 AM	09:32 AM	0:08:53.074512	88.85%	Present	N/A
2	0:10:00	Edward Yeoh Hong Enn	22ADB02687	N/A	N/A	N/A	N/A	Absent	N/A
3	0:10:00	Chew You Hui	20ADB04796	09:22 AM	09:31 AM	0:09:12.854102	92.14%	Present	N/A
4	0:10:00	Boey Zhi Xuan	20AGB05745	N/A	N/A	N/A	N/A	Absent	N/A
5	0:10:00	Tcy	21AGB00000	N/A	N/A	N/A	N/A	Absent	N/A

Figure 4.28: The Attendance of students for UGEB1011 Class C

4.3 Discussion

4.3.1 Hardware Setup

The Rapoo C280 Webcam 2K HD will be setup and use to detect and recognize the human faces.

4.3.2 Model Training

In this section will provide a comprehensive overview of the entire model training process, starting from collecting the dataset through training the model until deploys it for the next phase of face detection.

4.3.2.1 Dataset Collection

Before the beginning of model training, the data needs to be pre-processed and fed to a pre-trained model to enable it to learn and complete detection and classification tasks effectively. In this case, the model used for face detection and classification is trained to detect as well as classify the target object that is faces under a single class. For this, the roboflow tool has been used in order to annotate each face present in the images. In terms of annotation, roboflow provides an ease of use where the users can manually adjust the bounding boxes around the detected faces depending on their interpretation.

The process of resizing the image and all the other preprocessing steps is all done in the tool and then once the annotation is done, the Roboflow tool provides a very neatly formatted dataset. The described dataset is then used to train the face detection YOLOv8 model and the desired distance estimates. It must also be noted that annotation especially labelling is critical in this process because if the labels are wrong, the model will not be able to locate the faces within the images. Another useful feature of Roboflow is auto-labelling, which automatically generates bounding box labels, and which can be easily adjusted by the user. Thereafter, Roboflow splits up the images with all the labels into the training, validation, and testing datasets whilst ensuring the preservation of the labelled data. This automated partitioning helps keep things ordered while making it easy to move to the training phase.

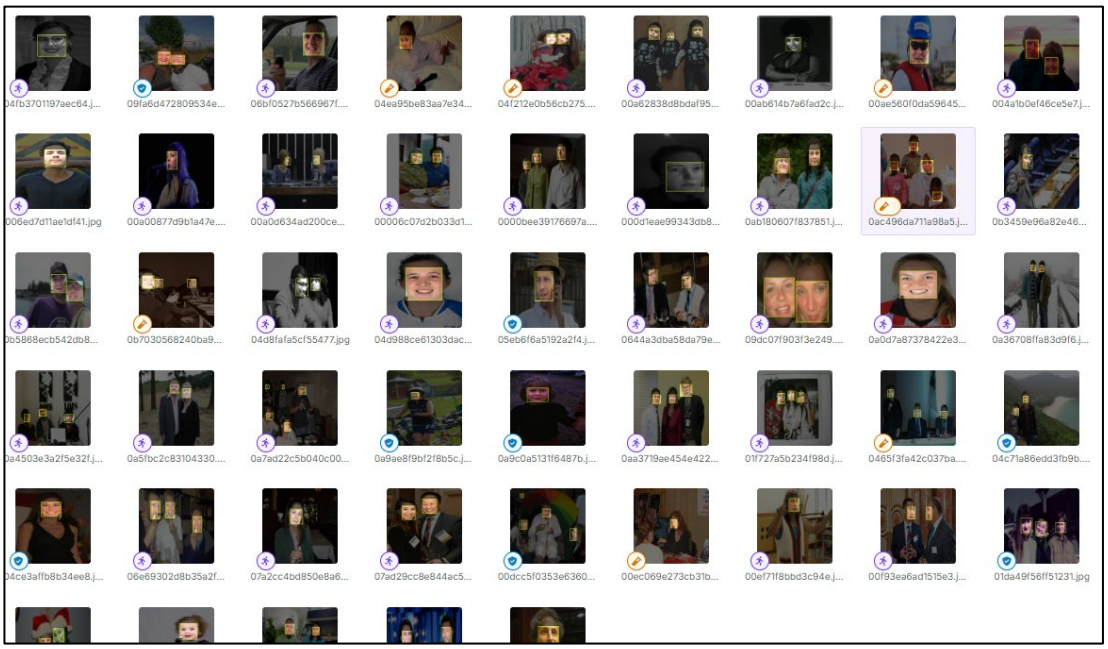


Figure 4.29: The data that use to train model

4.3.2.2 Train YOLOv8 for face detection

After the dataset has been correctly labelled and created the training session of the YOLOv8 model commences. In this phase, the model fine tunes the trained parameters in the process and classification of the faces' annotated images which aid the model in the real-time face recognition systems. This kind of preparation and training is crucial in making sure that at the end the model is fully capable of recognizing faces even in different settings and thus makes a good model for use in tracking attendance or any other tasks that may involve recognition of faces.

```
if __name__ == "__main__":
    import multiprocessing
    multiprocessing.set_start_method('spawn')
    from multiprocessing import freeze_support

    from ultralytics import YOLO
    import torch

    freeze_support()

    data_yaml_path =
    r"C:\Users\KXUANREN\OneDrive\UTAR_AllPrograms\Uni Courses
    Y3S2\FYP_1\FYP
    code\dataset_source\face_detection.yolov8\data.yaml"

    device = 'cuda' if torch.cuda.is_available() else 'cpu'
    print(f"Using device: {device}")

    model = YOLO("yolov8n.pt")
    model.to(device)

    model.train(data=data_yaml_path, epochs=20, device=device,
    batch=2, workers=0)

    model.save("face_detector.pt")

    print("Training completed!")
```

Figure 4.30: Training the face detection model

Figure 4.30 show the code used in training face detection model. However, before the start of the learning process, the system checks whether CUDA is present, having been detected in the process of training with the help of GPU. The model fine-tuning is performed to the model called ‘yolov8n. pt’, which in turn is based on YOLOv8 model.

It is carried out through 20 epochs, which seem to be quite reasonable because of the relatively small amount of data. Less epochs are helpful in preventing overfitting which could be a major issue in case of many epochs are used. Since memory is limited, batch size of 2 is chosen to eliminate out-of-memory (OOM) during the runtime of the process. But once the training is done the model is saved as “face_detector. pt” without needing any more instructions. From the above train and test processes, the newly trained model of face detection is effective in real-time application’s recognition since it utilises the learned patterns from the dataset.

4.3.3 Model Evaluation

After the model was trained successfully, model performance was assessed based on the measurements that are precision, recall and F1 score. Precision checks the number of correct face detection by the model, recall reviews the ability of the model to pick up all faces, the F1 score checks the model’s performance all over and then averages then both the precision and the recall scores. These metrics provide a versatile picture of the model’s performance as a tool for real-time face detection and recognition.

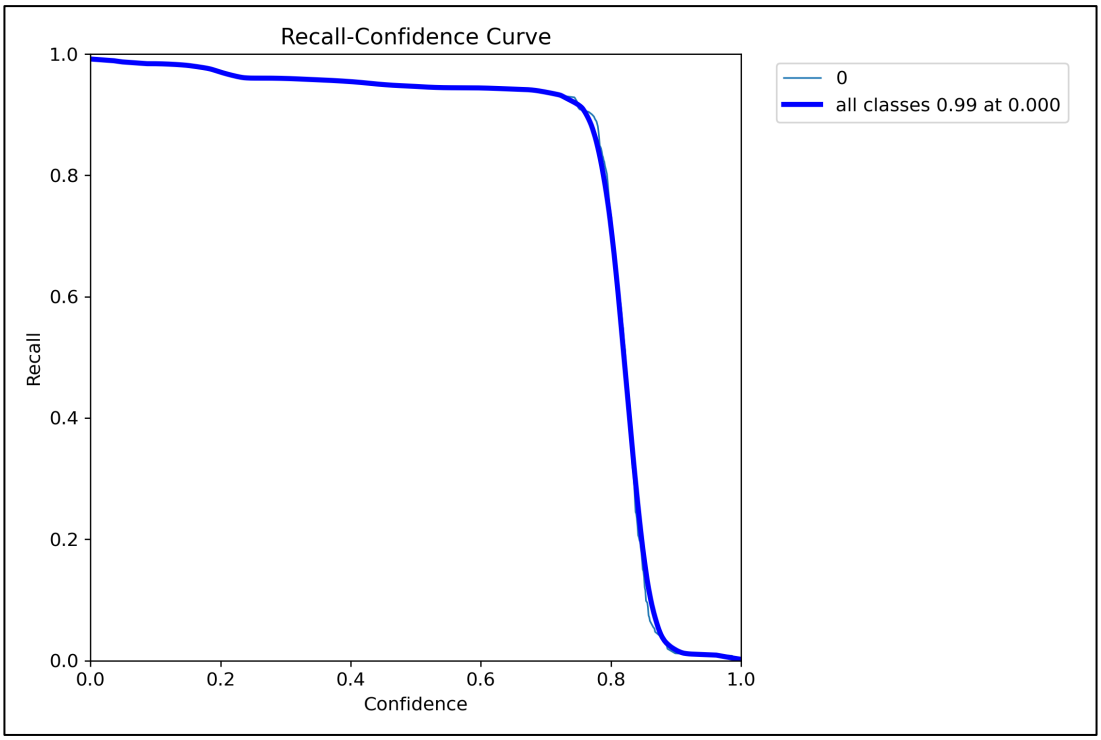


Figure 4.31: The result of recall confidence curve

Figure 4.31 show the recall of the face detection model. It can be seen that the confidence score is inversely proportional with the recall rate of the system. As indicated in figure above, for the overall recall, it gets the value of 0.99 meaning that model is capable of detecting actual 99% face out of the entire set. This indicates that the model seems to have a very high recall value, which suggests that it provides a relatively perfect performance in terms of detecting faces without too many missed detections.

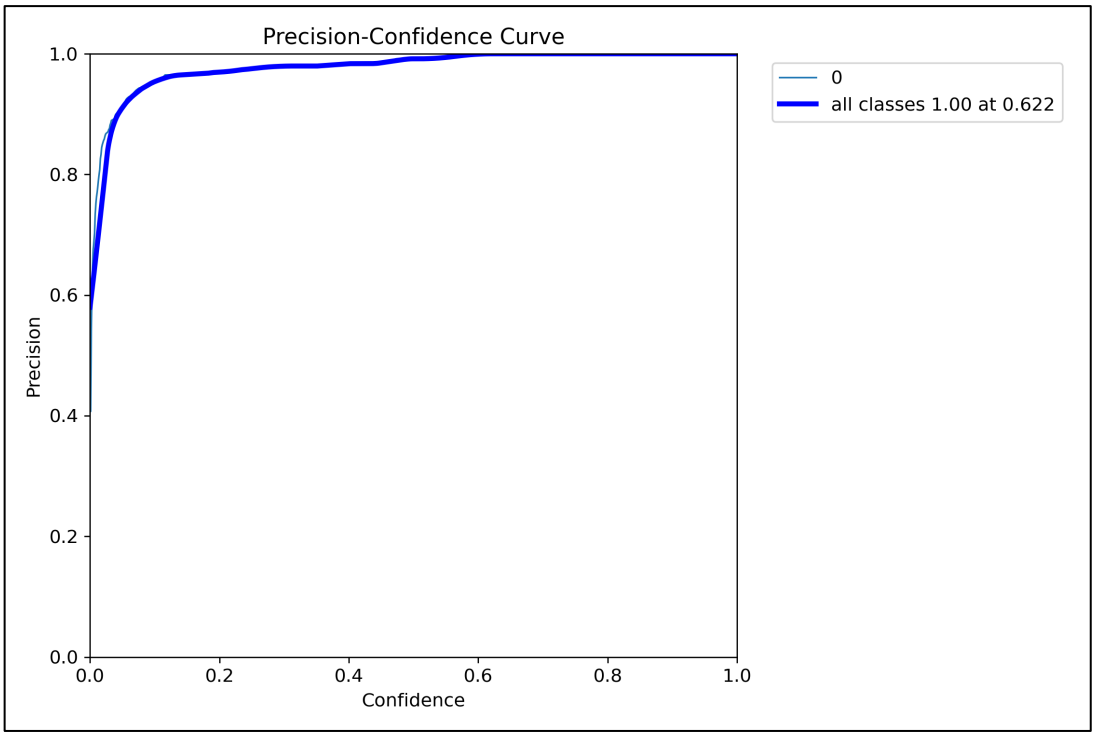


Figure 4.32: The result of precision confidence curve

Figure 4.32 shows the value of the parameter, which defines precision of the face detection model. This is evident from a precision score of a 1.00 for all classes meaning that the model correctly predicts 100% of the identities it labels as faces without any false positive identification. The threshold of 0. 622 stands for the IoU criterion hence it means that an intersection of the predicted bounding box and the ground truth bounding box should be no less than 62. 2%. This shows that despite the use of this much overlap in testing for accuracy of the model's prediction, the model has not lost its accuracy.

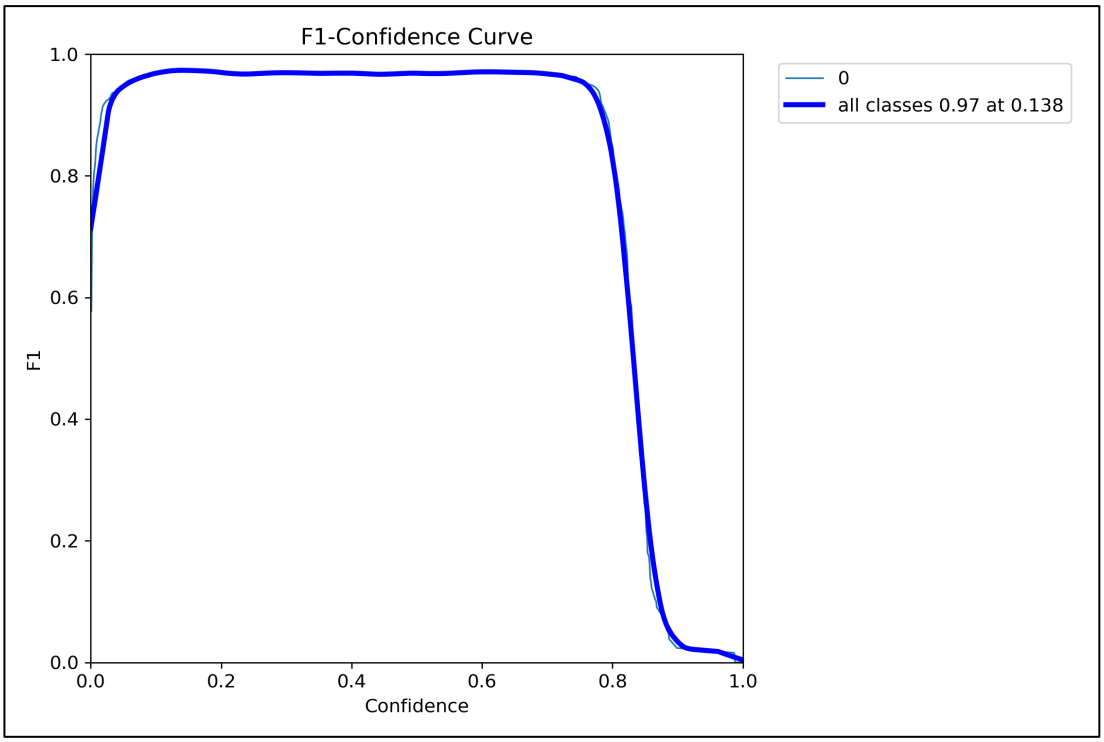


Figure 4.33: The result of F1 score confidence curve

Figure 4.33 shows the F1 score metric for face detection model. It means F1 score is 0.97 at the said threshold of 0.138 is an indication that recall and precision are high for all classes of the model and is therefore an indication of the high ability by the model in achieving high results. The F1 score, which is an average of precision and recollections, is an accurate measurement of the model's efficiency. The threshold of 0. 138 which is the IoU or a similar criterion helps to define the measurable minimum overlap between the predictive bounding boxes and the actual instances' bounding boxes.

4.3.4 Encoding and compare

In this section will show the dlib's face encoding process. It uses a deep learning-based approach to extract a 128-dimensional feature vector (face embedding) for each face. This encoding represents the unique features of a face, which is used for comparing and matching faces.

```
model_path1 = r'saved_model\face_detector.pt'
model1 = YOLO(model_path1)
model1.to(device)
```

Figure 4.34: Load the face detection model

Figure 4.34 shows how to detect and recognize faces in real-time using YOLOv8 and do encoding using dlib. Before starting encoding, the system first imports the pre-trained YOLOv8 face detector model name “face_detector. pt” which is going to be used during face detecting during the real-time session. The use of model for face detection plays a crucial role since it pinpoints the area within the video stream feed or real-time that contains the faces hence only that region is captured for encoding.

```
def face_encodings(face_image, known_face_locations=None,
num_jitters=1, verbose=True):
    face_encoding = []
    for raw_landmark_set in raw_landmarks:
        face_detail =
face_encoder.compute_face_descriptor(face_image, raw_landmark_set,
num_jitters)
        face_detail_encoding = np.array(face_detail)
        face_encoding.append(face_detail_encoding)
```

Figure 4.35: Face encoding

```
def save_known_faces(known_face_encodings, known_face_names,
file_path='saved_model/dlib_C_faces.pkl'):
```

Figure 4.36: Saving the known faces

After the pre-processing, Figure 4.35 shows the face encoding for each dataset is done. It is a process of converting the detected face into a 128-dimensional vector representation of the face. The system computes face encodings by first converting the face locations into "dlib.rectangle" objects which represent the detected face boundaries. Afterward, it computes the facial landmarks i.e. location of eyes, nose, and mouth in order to accurately encode the facial features. Figure 4.36 indicate these encodings are stored in a .pkl file, making sure all the known faces from the dataset are prepared to be compared when detecting face in real-time.

```
def compare_faces(known_face_encodings, face_encoding_to_check,
tolerance=0.6):
    if len(known_face_encodings) == 0:
        return np.empty((0))
    distance = np.linalg.norm(known_face_encodings -
face_encoding_to_check, axis=1)
    best_match_index = np.argmin(distance)
    best_distance = distance[best_match_index]
    match_list = list(distance <= tolerance)
    match = match_list[best_match_index]
    return match, best_match_index, best_distance
```

Figure 4.37: Comparing the known face with the new detected face

In the case of real-time image for face recognition, it uses face_detector.pt model again to find faces in an input live feed. The encoding process is then repeated for each detected face, producing a 128-dimensional vector of the on-the-fly face. This encoding is then compared to the known face encodings loaded from saving it previously in a .pkl file. This comparison is done by measuring the Euclidean distance between the real-time encoding and each known face encoding. In this case, a face is matched if its distance to the reference image falls beneath some standard threshold (usually is 0.6).

```
def load_known_faces(dataset_path):
    known_face_names = []
    known_face_encodings = []
    total = len(datasets)
    count = 0
    for person_name in datasets:
```

```
        count += 1
        person_folder = os.path.join(dataset_path , person_name)
        if os.path.isdir(person_folder):
            persons  = os.listdir(person_folder)
            total_image = len(persons)
            count_image = 0
            for image_name in persons:
                image_path = os.path.join(person_folder, image_name)
                image = PIL.Image.open(image_path)
                mode = 'RGB'
                if mode:
                    image = image.convert(mode)
                image = np.array(image)
                face_locations = detect_face_location_YOLO(image,
verbose=False)
                if face_locations:
                    face_encoding = face_encodings(image,
face_locations, verbose=False)[0]
                    known_face_encodings.append(face_encoding)
                    known_face_names.append(person_name)
                count_image+=1
                print_progress_bar(count, total,count_image,
total_image)
```

Figure 4.38: Load the known encoding face

When a match is found, the system fetches all of the data such as Name and ID related to that known face in the dataset. This enables the system to show, in real-time, who the detected person is perfect for keeping a record of student attendance or other uses where face recognition would be crucial. From face detection to coding comparison, YOLO and dlib ensure real-time and trouble-free face recognition.

4.3.5 Logging to Excel file

Once the system successfully detects a known person, it then logs the student's attendance in an Excel file. This guarantees that a student log will be maintained orderly on a daily basis for each pupil in the class.

```
    def create_or_load_excel(self, course_name):
        file_name = "attendance_record.xlsx"

        headers = ["No.", "Class Duration", "Student Name", "Student
ID", "Sign-in Time", "Sign-out Time", "Duration", "Total Time",
"Attendance", "Status"]
        workbook.save(file_name)
        return file_name
```

Figure 4.39: Create an Excel file

Figure 4.39 reflects the original structure of an Excel file. It lays out the columns for things like student name, student ID, sign-in time, sign-out time, etc. It could also incorporate course-specific information and multi-session attendance tracking with session dates included in the file.

```
    def save_attendance_to_excel(self):
        workbook = load_workbook(self.excel_file_path)
        sheet = workbook[self.course_name]

            if sign_in_time and sign_out_time:
                duration = sign_out_time - sign_in_time
                percentage = calculate_percentage(duration,
self.course_duration)
                attendance = "Present" if percentage >= 10 else
"Absent"
```

Figure 4.40: Saved to the Excel file

Figure 4.40 shows when a student is detected, their attendance (sign-in time and sign-out time) is automatically saved to the relevant row in the Excel file. This helps calculate whether the student has spent a sufficient amount of time in class to be marked as "Present," or if they haven't met the required minimum, they are marked as "Absent." Based on this, the system retrieves recognized student information and fills out their relevant fields, leading to comprehensive attendance records that can be easily accessed.

4.4 Limitations

This face encoding (which relies on dlib) cannot be accelerated with the GPU, due to system limitations. This makes the encoding process very slow, especially when multiple faces are detected at the same time. As the number of faces the system must detect increases, the time required to calculate the encoding increases, which slows down the entire face recognition process. This becomes a big problem in very time-critical applications where high throughput processing is required to run smoothly.

Another difficulty is related to lighting conditions. Illumination changes can greatly affect the accuracy of face recognition. For example, if the faces in the dataset were acquired under optimal lighting conditions, but the live image is poorly illuminated or has uneven illumination, the detection rate may be affected. So, with that in mind, images of each person in the dataset may need to be captured under a wider variety of lighting conditions. Alternatively, the system could implement a method to adjust the colour and brightness of each detected face in real time before matching it to the dataset, thus improving overall accuracy.

In addition, the face detection camera (Rapoo C280 Webcam 2K HD) lacks a night mode function, which results in poorer results in low-light or dark environments. If there is insufficient lighting, the system may not be able to correctly identify or identify faces, which will reduce its effectiveness at night or in dimly lit scenes. This limitation suggests that the system would be greatly improved by using camera hardware with better low-light specifications or by employing external lighting to improve detection accuracy in poor visibility conditions.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the project has reached all the objectives and built an effective automatic system which ensures attendance is recorded easily through facial recognition technology. To begin with, YOLO uses CNN to calculate the class probabilities of objects in input photos to implement an automatic facial recognition system that can accurately detect and recognize human faces. This system not only detects but also identifies an It's reliable even when people are in various places to ensure that this objective is met. Hence, I have created an AI-based face recognition system for class attendance.

The second objective, which focused on analyse the data collected from the designed class attendance system, was also realized through use of YOLO the facial detection and facial encoding techniques. This encoding process is achieved through representing each face as a unique 128-dimensional vector and making accurate comparisons between them such that individuals are unmistakably matched according to their facial features by the system. Such approach employs Euclidean distance metric which guarantees the high level of precision in recognizing and differentiating faces even under complicated circumstances. Therefore, face detection, encoding and comparison procedure has been crucial in providing reliability and strength to the class attendance system.

Lastly, the project has streamlined management processes by integrating Excel-based attendance recording and management. Besides, it has the capability to calculate total attendance percentage per student hence providing a complete automated for tracking attendance over a period of time. Administrative efficiency is enhanced through this feature as a result accurate and consistent attendance records are kept reducing human mistakes thus giving teachers and administrators more time for other important tasks.

5.2 Recommendations

In this place, several recommendations that could improve the future improvements to the project in order to improve perform of the system were developed. Some of the recommendations are as follows, first, there is the issue of having a camera that can work well under both day and night. This would enhance the facial recognition since it's possible to have instances where the face is illuminated resulting in a clear identification of the face. By eliminating distortions introduced by changing lighting conditions, the strategy will lead to increased accuracy of the still method in real-world settings.

The other recommendation is to consider another approach to face encoding and can also try to use other approaches or replace facial recognition algorithms or libraries for training this model. At the moment, it uses Dlib encoding, and this is a disadvantage because it does not support the use of GPU. This results to slower processing time than the traditional face encoding and comparison. Thus, if the GPU methods are tested and used in the encoding and recognition procedures, the system's speed will be increased, and efficiency improved. However, other possible workaround to avoid several days of encoding time is also to use the available access to high performance CPUs and GPUs offered in tools such as Google Colab. The use of such platforms would also help to perform computational tasks much faster and without the need for special hardware which in turn would improve the flow of the system.

Moreover, the optimization of student's registration process and giving an opportunity to students to register their faces online also would save a lot of time. Rather than registering each student by going through the process one by one the use of online registration system could be adopted this could help reduce the inconveniences caused by the process.

Enhancing the quality of the camera used in classroom is also highly encouraged for example using a good quality camera. Larger F. O. V cameras of better quality would help the system to scan more people at once – up to 10-30 students. This would help in increasing the effectiveness of the system's capability to detect the face in a complex environment such as a classroom. In addition, with so much enhancement in algorithms, adding some algorithms would help in easily detecting fake faces like photo on a phone hence reducing chances of cheating in addition to making the attendance system more reliable.

REFERENCES

Abramov, M. (2023). *Comparing YOLOv8 and YOLOv7: What's New?* [online] Keylabs: latest news and updates. Available at: https://keylabs.ai/blog/comparing-yolov8-and-yolov7-whats-new/#:~:text=YOLOv8%20and%20YOLOv7%20are%20versions.

Augmented A.I. (2023). *Unlock the Full Potential of Object Detection with YOLOv8*. [online] www.augmentedstartups.com. Available at: https://www.augmentedstartups.com/blog/unlock-the-full-potential-of-object-detection-with-yolov8-faster-and-more-accurate-than-yolov7-2.

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K. and Ghayvat, H. (2021). CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics*, 10(20), p.2470.

Bouvrie, J. (2006). *Notes on Convolutional Neural Networks*.

Cui, Y., Jin, J.S., Luo, S., Park, M. and Sherlock S.L. Au (2009). Automated Pattern Recognition and Defect Inspection System. *proc. 5th International Conference on Computer Vision and Graphical Image*, 59, pp.768–773. doi:https://doi.org/10.1109/icig.2009.144.

E King, D. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10, pp.1755–1758.

elifezgisen (2023). *GitHub - elifezgisen/face-recognition: It is a Face Recognition project prepared using libraries such as OpenCV and Dlib.* [online] GitHub. Available at: https://github.com/elifezgisen/face-recognition [Accessed 8 Sep. 2024].

Fischler, M.A. and Elschlager, R.A. (1973). The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, C-22(1), pp.67–92. doi:https://doi.org/10.1109/t-c.1973.223602.

Goldstein, A.J., Harmon, L.D. and Lesk, A.B. (1971). Identification of human faces. *Proceedings of the IEEE*, 59(5), pp.748–760. doi:https://doi.org/10.1109/proc.1971.8254.

He, K., Zhang, X., Ren, S. and Sun, J. (2016a). Deep Residual Learning for Image Recognition. *Thecvf.com*, [online] pp.770–778. Available at: http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

He, K., Zhang, X., Ren, S. and Sun, J. (2016b). Identity Mappings in Deep Residual Networks. *Computer Vision – ECCV 2016*, pp.630–645. doi:https://doi.org/10.1007/978-3-319-46493-0_38.

Huang, L., Qin, J., Zhou, Y., Fang, Y., Liu, L. and Shao, L. (2023). Normalization Techniques in Training DNNs: Methodology, Analysis and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp.1–20.

IBM (2023). *What Is Machine Learning?* [online] IBM. Available at: https://www.ibm.com/topics/machine-learning.

Kirk, D. (2007). NVIDIA cuda software and gpu parallel computing architecture. *Proceedings of the 6th international symposium on Memory management - ISMM '07*. doi:https://doi.org/10.1145/1296907.1296909.

Mirkhan, A. (2023). *YOLO Algorithm: Real-Time Object Detection from A to Z*. [online] kili-website. Available at: https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z [Accessed 20 Apr. 2024].

Paramjeet Kaur, E. and Nishi, E. (2014). *A Survey on CUDA*. [online] Available at: https://www.ijcsit.com/docs/Volume%205/vol5issue02/ijcsit20140502282.pdf [Accessed 8 Sep. 2024].

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [online] pp.779–788. Available at: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html.

Rosebrock, A. (2018). *Face recognition with OpenCV, Python, and deep learning*. [online] PyImageSearch. Available at: https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/.

S.S.R. Abibi (2002). Proceedings TENCON 2000. *Simulating evolution: connectionist metaphors for studying human cognitive behaviour*, 1, pp.167–173. doi:https://doi.org/10.1109/tencon.2000.893563.

Stepanov, S., Spiridonov, D. and Mai, T. (2023). Prediction of numerical homogenization using deep learning for the Richards equation. *Journal of Computational and Applied Mathematics*, 424, p.114980.

Uddin, Md.P., Mamun, Md.A. and Hossain, Md.A. (2021). PCA-based Feature Reduction for Hyperspectral Remote Sensing Image Classification. *IETE Technical Review*, 38(4): 377–396.

Viswanatha, V., Chandana R, K. and Ramachandra, A.C. (2022). *Real Time Object Detection System with YOLO and CNN Models: A Review*. [online] Available at: https://arxiv.org/ftp/arxiv/papers/2208/2208.00773.pdf.

Zhao, X., Wang, L., Zhang, Y., Han, X., Muhammet Deveci and Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4). doi:https://doi.org/10.1007/s10462-024-10721-6.

APPENDICES

APPENDIX A: Main Register Code

```
from registration import register_course
from registration import register_user

if __name__ == "__main__":
    course_name = register_course()
    register_user(course_name)
```

APPENDIX B: Registration

```
import re
import cv2
import os
import json
from ultralytics import YOLO
import torch

device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f"Using device: {device}")
```

```
def validate_student_id(student_id):
    pattern = r'^2\d[A-Z]{3}\d{5}$'
    return re.match(pattern, student_id)

def validate_course_name(course_name):
    pattern = r'^[A-Z]{3,4}\d{4,5} .+$'
    return re.match(pattern, course_name)

def format_name(name):
    return ' '.join(part.capitalize() for part in name.split())

def save_registration_data(name, student_id, course_name):
    registration_data = {
        "name": name,
        "student_id": student_id
    }
    json_file_path = f"dataset_json/{course_name}.json"

    if os.path.exists(json_file_path):
        with open(json_file_path, 'r') as f:
            data = json.load(f)
    else:
        data = []

    for entry in data:
        if entry['name'] == name and entry['student_id'] ==
student_id:
            print("You already registered.")
            return False

    data.append(registration_data)

    with open(json_file_path, 'w') as f:
        json.dump(data, f, indent=4)

    return True
```

```
def register_course():
    while True:
        course_name = input("Please Enter Your Course Name (e.g., 
UGEB3016 Final Year Project 1): ")
        if validate_course_name(course_name):
            json_file_path = f"dataset_json/{course_name}.json"
            dataset_path = os.path.join("dataset", course_name)
            if os.path.exists(json_file_path) and 
os.path.exists(dataset_path):
                print(f"Course {course_name} already exists. Adding 
new student to this course.")
            else:
                print(f"Creating new course entry for 
{course_name}.")
                if not os.path.exists("dataset_json"):
                    os.makedirs("dataset_json")
                if not os.path.exists(dataset_path):
                    os.makedirs(dataset_path)
            return course_name
        else:
            print("Invalid course name format, e.g., UGEB0316 Final 
Year Project")

def register_user(course_name):
    name = input("Please Enter Your Name: ")
    formatted_name = format_name(name)
    if formatted_name != name:
        print(f"Name corrected to: {formatted_name}")

    while True:
        student_id = input("Please Enter Your Student ID: ")
        if validate_student_id(student_id):
            print("Wait to start capturing your face...")
            if video_loop(formatted_name, student_id, course_name):
                save_registration_data(formatted_name, student_id, 
course_name)
            break
```

```
        else:
            print(f"Invalid student ID format, e.g., 21AGB06806:
{student_id}")

def video_loop(name, student_id, course_name, max_images=50):
    dataset_path = os.path.join("dataset", course_name)

    student_folder = os.path.join(dataset_path,
f"{name}_{student_id}")
    if not os.path.exists(student_folder):
        os.makedirs(student_folder)

    cam = cv2.VideoCapture(0)
    cv2.namedWindow("Capture Face")

    face_model_path =
r'C:\Users\KXUANREN\OneDrive\UTAR_AllPrograms\Uni Courses
Y3S2\FYP_1\FYP code\face_detector.pt'
    face_model = YOLO(face_model_path)

    img_counter = 0
    success = False

    while img_counter < max_images:
        ret, frame = cam.read()
        if not ret:
            print("Failed to grab frame")
            break

        results = face_model(frame)
        faces = results[0].boxes

        face_detected = False
        for box in faces:
            cls = int(box.cls[0])
            if cls == 0:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
```

```
                face_roi = frame[y1:y2, x1:x2]
                face_detected = True
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0),
2)

        if not face_detected:
            cv2.putText(frame, "Please put your face in the right
position",
                        (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2, cv2.LINE_AA)

        cv2.imshow("Capture Face", frame)

        k = cv2.waitKey(1)
        if k % 256 == 27:
            print("Escape hit, closing...")
            break
        elif k % 256 == 32 and face_detected:
            face_img_name = os.path.join(student_folder,
f"{name}_{student_id}_{img_counter}.png")
            cv2.imwrite(face_img_name, face_roi)
            print(f"{face_img_name} written!")
            img_counter += 1

    if img_counter == max_images:
        success = True

    cam.release()
    cv2.destroyAllWindows()

    if not success:
        if os.path.exists(student_folder):
            for file in os.listdir(student_folder):
                os.remove(os.path.join(student_folder, file))
            os.rmdir(student_folder)
        print(f"Failed to capture images for {name} (ID:
{student_id})")
```

```
    else:
        print(f"Captured {img_counter} images for {name} (ID:
{student_id})")

    return success

if __name__ == "__main__":
    course_name = register_course()
    register_user(course_name)
```

APPENDIX C: Face Encoding and Compare

```
import os
import sys
import pickle
import PIL
import numpy as np
import dlib
from PIL import ImageFile
import time
import torch
import multiprocessing
multiprocessing.set_start_method('spawn')
from multiprocessing import freeze_support
from ultralytics import YOLO

freeze_support()

device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f"Using device: {device}")
```

```
model_path1 = r'saved_model\face_detector.pt'
model1 = YOLO(model_path1)
model1.to(device)

ImageFile.LOAD_TRUNCATED_IMAGES = True

model_path = 'saved_model'

predictor_point_model =
os.path.join(model_path,"shape_predictor_5_face_landmarks.dat") #
facenet_keras.h5
face_recognition_model =
os.path.join(model_path,"dlib_face_recognition_resnet_model_v1.dat")

def print_progress_bar(main_iteration, main_total, iteration, total,
length=20):
    percent = ("{0:.1f}").format(100 * (iteration / float(total)))
    filled_length = int(length * iteration // total)
    bar = '' * filled_length + '-' * (length - filled_length)
    sys.stdout.write(f'\rTraining --> {main_iteration}/{main_total}
[{bar}] {percent}% Complete')
    sys.stdout.flush()

def save_known_faces(known_face_encodings, known_face_names,
file_path='saved_model/dlib_C_faces.pkl'):
        if not os.path.exists(os.path.dirname(file_path)):
            os.makedirs(os.path.dirname(file_path))

        with open(file_path, 'wb') as f:
            pickle.dump({
                'encodings': known_face_encodings,
                'names': known_face_names
            }, f)
        print(f"Saved known faces to {file_path}")

def load_known_faces(dataset_path):
    known_face_names = []
```

```
    known_face_encodings = []
    datasets = os.listdir(dataset_path)
    total = len(datasets)
    count = 0
    for person_name in datasets:
        count += 1
        person_folder = os.path.join(dataset_path , person_name)
        if os.path.isdir(person_folder):
            persons  = os.listdir(person_folder)
            total_image = len(persons)
            count_image = 0
            for image_name in persons:
                image_path = os.path.join(person_folder, image_name)
                image = PIL.Image.open(image_path)
                mode = 'RGB'
                if mode:
                    image = image.convert(mode)
                image = np.array(image)
                face_locations = detect_face_location_YOLO(image,
verbose=False)
                if face_locations:
                    face_encoding = face_encodings(image,
face_locations, verbose=False)[0]
                    known_face_encodings.append(face_encoding)
                    known_face_names.append(person_name)
                count_image+=1
                print_progress_bar(count, total,count_image,
total_image)
        print()
    print("Training complete!")
    save_known_faces(known_face_encodings, known_face_names)

    return known_face_encodings, known_face_names

def detect_face_location_YOLO(image, verbose=True):
    results = model1.predict(source=image, conf=0.5, save=False,
stream=True, device=device, verbose=verbose)
```

```
    boxes = []
    for result in results:
        orig_img = np.copy(result.orig_img)
        for detection_index, detection in enumerate(result):
            detection_boxes = detection.boxes
            if int(detection_boxes.cls) == 0:  # Assuming class 0 is
face
                x1, y1, x2, y2 = map(int, detection_boxes.xyxy[0])
                # boxes.append((x1, y1, x2, y2))
                boxes.append((y1, x2, y2, x1))
    return boxes

def detect_face_location_dlib(img, verbose=True):
    start_time = time.time()
    face_detector = dlib.get_frontal_face_detector()
    faces = face_detector(img, 1)
    end_time = time.time()
    if verbose == True:
        print(f"DLIB_Detection taken 1: {((end_time - start_time) *
1000):.4f} ms")

    return [(rect.top(), rect.right(), rect.bottom(), rect.left())
for rect in faces]

def compare_faces(known_face_encodings, face_encoding_to_check,
tolerance=0.6):
    if len(known_face_encodings) == 0:
        return np.empty((0))
    distance = np.linalg.norm(known_face_encodings -
face_encoding_to_check, axis=1)
    best_match_index = np.argmin(distance)
    best_distance = distance[best_match_index]
    match_list = list(distance <= tolerance)
    match = match_list[best_match_index]
    return match, best_match_index, best_distance
```

```
def face_encodings(face_image, known_face_locations=None,
num_jitters=1, verbose=True):
    start_time = time.time()
    if known_face_locations is None:
        known_face_locations = detect_face_location_dlib(face_image)

    known_face_locations = [dlib.rectangle(face_location[3],
face_location[0], face_location[1], face_location[2]) for
face_location in known_face_locations]

    pose_predictor = dlib.shape_predictor(predictor_point_model)
    face_encoder =
dlib.face_recognition_model_v1(face_recognition_model)

    raw_landmarks = []
    for known_face_location in known_face_locations:
        pose_predict = pose_predictor(face_image,
known_face_location)
        raw_landmarks.append(pose_predict)

    face_encoding = []
    for raw_landmark_set in raw_landmarks:
        face_detail =
face_encoder.compute_face_descriptor(face_image, raw_landmark_set,
num_jitters)
        face_detail_encoding = np.array(face_detail)
        face_encoding.append(face_detail_encoding)

    end_time = time.time()
    if verbose == True:
        print(f"Time taken 1: {((end_time - start_time) * 1000):.4f}
ms")
    return face_encoding
```

APPENDIX D: Real-time Face Recognition

```
import os
import cv2
import pickle
import face_reco_2 as face_reco
import json
import sys
from realtime_ui import start_realtime_ui
from PIL import ImageTk, Image
import time
from openpyxl import Workbook, load_workbook
from openpyxl.styles import PatternFill
from datetime import datetime

tolerance = 0.5

json_directory = r'C:\Users\KXUANREN\Desktop\UTAR_AllPrograms\Uni
Courses Y3S2\FYP_1\FYP code\dataset_json'

def load_student_data(json_file):
    with open(json_file, 'r') as f:
        return json.load(f)

def get_course_file():
    course_name = input("Course name (eg: UGEB3016 Final Year Project
1): ")
    json_file = os.path.join(json_directory, f'{course_name}.json')

    if os.path.exists(json_file):
        return json_file, course_name
    else:
        print(f"No JSON file found for course '{course_name}' in the
specified directory.")
        sys.exit(1)
```

```
def get_course_and_student_data():
    json_file, course_name = get_course_file()
    student_data = load_student_data(json_file)
    return json_file, course_name, student_data

def get_course_duration():
    while True:
        try:
            duration_input = input("Enter the class duration (hh:mm):
")
            duration = datetime.strptime(duration_input, '%H:%M') -
datetime(1900, 1, 1)
            return duration
        except ValueError:
            print("Invalid duration format. Please use 'hh:mm'.")

class FaceRecognize():
    def __init__(self, input_source=None):
        super().__init__()

        self.input_source = input_source

        json_file, course_name, student_data =
get_course_and_student_data()
        self.dataset_path = os.path.join('dataset', course_name)
        self.course_duration = get_course_duration()
        self.known_face_encodings, self.known_face_names =
self.load_or_train_faces()
        self.excel_file_path = self.create_or_load_excel(course_name)
        self.course_name = course_name
        self.student_data = student_data

        self.sign_in_times = {}
        self.sign_out_times = {}

        self.start_gui(course_name, student_data)
```

```
    def load_or_train_faces(self):
        pkl_file = 'saved_model/dlib_C_faces.pkl'

        if os.path.exists(pkl_file):
            with open(pkl_file, 'rb') as f:
                data = pickle.load(f)
            print('Loaded known faces from file.')
            print('Opening the camera...\nPlease wait...')
            return data['encodings'], data['names']
        else:
            print(f'{pkl_file} not found. Training model using
dataset at {self.dataset_path}.')
            return face_reco.load_known_faces(self.dataset_path)

    def process_frame(self, frame, known_face_encodings,
known_face_names):
        face_locations_yolo =
face_reco.detect_face_location_YOLO(frame, True)
        face_encodings = face_reco.face_encodings(frame,
face_locations_yolo, verbose=True)

        detected_faces = {}

        for (y1, x2, y2, x1), face_encoding in
zip(face_locations_yolo, face_encodings):
            match, best_match_index, best_distance =
face_reco.compare_faces(known_face_encodings, face_encoding,
tolerance)

            confidence_level = 1.25 - best_distance

            if match and confidence_level >= 0.6:
                name = known_face_names[best_match_index]
                if name not in detected_faces or confidence_level >
detected_faces[name]['confidence']:
                    detected_faces[name] = {
```

```
                        'location': (y1, x2, y2, x1),
                        'confidence': confidence_level
                    }
            else:
                detected_faces[f"Unknown_{len(detected_faces)}"] = {
                    'location': (y1, x2, y2, x1),
                    'confidence': confidence_level,
                    'name': "Unknown"
                }

        for name, face_data in detected_faces.items():
            y1, x2, y2, x1 = face_data['location']
            colour = (0, 0, 255)

            if name != "Unknown" and face_data['confidence'] >= 0.75:
                colour = (0, 255, 0)
                if name not in self.sign_in_times:
                    self.sign_in_times[name] = datetime.now()
                    print(f"Sign-in time recorded for {name}:
{self.sign_in_times[name].strftime('%I:%M %p')}")
                self.sign_out_times[name] = datetime.now()
                print(f"Updated sign-out time for {name}:
{self.sign_out_times[name].strftime('%I:%M %p')}")

            name_fontsize = 0.35
            display_name = name if name != "Unknown" else "Unknown"
            (name_width, name_height), _ =
cv2.getTextSize(display_name, cv2.FONT_HERSHEY_SIMPLEX,
name_fontsize, 1)
            cv2.rectangle(frame, (x1, y2 - name_height - 10), (x2,
y2), colour, cv2.FILLED)
            cv2.putText(frame, display_name, (x1 + 6, y2 - 5),
cv2.FONT_HERSHEY_SIMPLEX, name_fontsize, (0, 0, 0), 1, cv2.LINE_AA)

            confidence_fontsize = 0.35
            confidence_text = f"{face_data['confidence']:.2f}"
```

```
            (conf_width, conf_height), _ =
cv2.getTextSize(confidence_text, cv2.FONT_HERSHEY_SIMPLEX,
confidence_fontsize, 1)
            cv2.rectangle(frame, (x1, y1 - conf_height - 10), (x1 +
conf_width, y1), colour, cv2.FILLED)
            cv2.putText(frame, confidence_text, (x1 + 6, y1 - 5),
cv2.FONT_HERSHEY_SIMPLEX, confidence_fontsize, (0, 0, 0), 1,
cv2.LINE_AA)
            cv2.rectangle(frame, (x1, y1), (x2, y2), colour, 2)

        return frame

    def start_processing(self, video_label):
        self.video_capture = cv2.VideoCapture(self.input_source)

        video_label.update_idletasks()
        video_label_width = video_label.winfo_width()
        video_label_height = video_label.winfo_height()

        start_time = time.time()
        end_time = start_time +
(self.course_duration.total_seconds())

        def update_video():
            while True:
                current_time = time.time()

                if current_time > end_time:
                    print("Class duration ended. Closing video
capture...")
                    self.save_attendance_to_excel()
                    break

                ret, frame = self.video_capture.read()
                if ret:
                    frame = cv2.flip(frame, 1)
```

```
                    frame = cv2.resize(frame, (video_label_width,
video_label_height))

                    frame = self.process_frame(frame,
self.known_face_encodings, self.known_face_names)
                    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                    img = Image.fromarray(img)
                    img = ImageTk.PhotoImage(img)

                    video_label.config(image=img)
                    video_label.image = img

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
                else:
                    print('Cannot capture frame, resetting capture.')
                    self.video_capture.release()
                    self.video_capture =
cv2.VideoCapture(self.input_source)

            self.video_capture.release()
            cv2.destroyAllWindows()

        update_video()

if __name__ == "__main__":
    input_source = 0
    app = FaceRecognize(input_source)
```

APPENDIX E: Create an Excel file to logging the student attendance

```
    def create_or_load_excel(self, course_name):
        file_name = "attendance_record.xlsx"
        today_date = datetime.now().strftime('%Y-%m-%d')

        if os.path.exists(file_name):
            workbook = load_workbook(file_name)
            if course_name in workbook.sheetnames:
                sheet = workbook[course_name]
            else:
                sheet = workbook.create_sheet(title=course_name)
        else:
            workbook = Workbook()
            sheet = workbook.active
            sheet.title = course_name

        last_row = sheet.max_row
        start_row = last_row + 4 if last_row > 0 else 5

        sheet.cell(row=start_row - 2, column=1, value="Date:")
        sheet.cell(row=start_row - 2, column=2, value=today_date)

        headers = ["No.", "Class Duration", "Student Name", "Student
ID", "Sign-in Time", "Sign-out Time", "Duration", "Total Time",
"Attendance", "Status"]
        for col_num, header in enumerate(headers, start=1):
            sheet.cell(row=start_row - 1, column=col_num,
value=header)

        workbook.save(file_name)
        return file_name

    def save_attendance_to_excel(self):
        workbook = load_workbook(self.excel_file_path)
```

```
        sheet = workbook[self.course_name]

        last_row = sheet.max_row
        start_row = last_row + 1

        for idx, student in enumerate(self.student_data,
start=start_row):
            student_name = student['name']
            student_id = student['student_id']
            full_name = f"{student_name}_{student_id}"

            sign_in_time = self.sign_in_times.get(full_name, None)
            sign_out_time = self.sign_out_times.get(full_name, None)

            if sign_in_time and sign_out_time:
                duration = sign_out_time - sign_in_time
                percentage = calculate_percentage(duration,
self.course_duration)
                attendance = "Present" if percentage >= 10 else
"Absent"

                if percentage >= 80:
                    status = "N/A"
                elif percentage > 10:
                    status = "Attendance below 80%"
                else:
                    status = "Attendance below 10%"

                duration_str = str(duration)
                percentage_str = f'{percentage:.2f}%'
            else:
                duration_str = "N/A"
                percentage_str = "N/A"
                attendance = "Absent"
                status = "N/A"

            sheet.append([
```

```
                idx - start_row + 1,
                str(self.course_duration),
                student_name,
                student_id,
                sign_in_time.strftime('%I:%M %p') if sign_in_time
else "N/A",
                sign_out_time.strftime('%I:%M %p') if sign_out_time
else "N/A",
                duration_str,
                percentage_str,
                attendance,
                status
            ])

        absent_fill = PatternFill(start_color='FF0000',
end_color='FF0000', fill_type='solid')
        below_80_fill = PatternFill(start_color='FFFF00',
end_color='FFFF00', fill_type='solid')

        end_row = start_row + len(self.student_data) - 1

        for row in sheet.iter_rows(min_row=start_row,
max_row=end_row, min_col=9, max_col=9):
            for cell in row:
                if cell.value == "Absent":
                    cell.fill = absent_fill
                elif cell.value == "Attendance below 80%":
                    cell.fill = below_80_fill

        workbook.save(self.excel_file_path)
        print(f"Attendance recorded in {self.excel_file_path}")

    def start_gui(self, course_name, student_data):
        start_realtime_ui(course_name, student_data,
self.start_processing)

def calculate_duration(sign_in_time, sign_out_time):
```

```
    return sign_out_time - sign_in_time

def calculate_percentage(duration, class_duration):
    if duration:
        duration_minutes = duration.total_seconds() / 60
        class_duration_minutes = class_duration.total_seconds() / 60
        percentage = (duration_minutes / class_duration_minutes) *
100
        return percentage
    return 0

if __name__ == "__main__":
    input_source = 0
    app = FaceRecognize(input_source)
```

APPENDIX F: Real-time UI

```
import tkinter as tk
from tkinter import Label, Text, Scrollbar, RIGHT, Y, END, BOTH
from threading import Thread

def display_student_data(data, student_info_text):
    student_info_text.delete(1.0, END)
    for idx, student in enumerate(data, start=1):
        name = student.get('name', 'Unknown')
        student_info_text.insert(END, f"{idx}. {name}\n")

def start_realtime_ui(course_name, student_data, update_frame):
    root = tk.Tk()
    root.title(course_name)
```

```
    root.grid_rowconfigure(0, weight=1)
    root.grid_columnconfigure(0, weight=4)
    root.grid_columnconfigure(1, weight=1)

    video_frame = tk.Frame(root, bg='black')
    video_frame.grid(row=0, column=0, sticky="nsew", padx=10,
pady=10)

    data_frame = tk.Frame(root)
    data_frame.grid(row=0, column=1, sticky="nsew", padx=10, pady=10)

    video_label = Label(video_frame)
    video_label.pack(fill=BOTH, expand=True)

    student_info_text = Text(data_frame, wrap="word", height=30,
width=40)
    student_info_text.pack(side="left", fill=BOTH, expand=True)

    scrollbar = Scrollbar(data_frame)
    scrollbar.pack(side=RIGHT, fill=Y)
    scrollbar.config(command=student_info_text.yview)
    student_info_text.config(yscrollcommand=scrollbar.set)

    display_student_data(student_data, student_info_text)

    def resize_video(event):
        video_label.update_idletasks()
        video_width = video_label.winfo_width()
        video_height = video_label.winfo_height()
        video_label.config(width=video_width, height=video_height)

    root.bind("<Configure>", resize_video)

    update_thread = Thread(target=lambda: update_frame(video_label))
    update_thread.daemon = True
    update_thread.start()
```

```
    root.mainloop()
```

APPENDIX G: Overall attendance checking

```
import os
from openpyxl import load_workbook, Workbook
from openpyxl.styles import PatternFill

def calculate_overall_attendance(course_name):
    file_name = "Overall Attendance.xlsx"

    if not os.path.exists(file_name):
        overall_workbook = Workbook()
        overall_workbook.save(file_name)
    else:
        overall_workbook = load_workbook(file_name)

    if course_name in overall_workbook.sheetnames:
        course_sheet = overall_workbook[course_name]
        overall_workbook.remove(course_sheet)  # Clear the existing
sheet by removing it
        course_sheet = overall_workbook.create_sheet(course_name)  #
Create a new empty sheet
    else:
        course_sheet = overall_workbook.create_sheet(course_name)  #
Create a new sheet for the course

    attendance_file = "attendance_record.xlsx"
    if not os.path.exists(attendance_file):
        print("Attendance record file does not exist.")
        return
```

```
    attendance_workbook = load_workbook(attendance_file)

    if course_name not in attendance_workbook.sheetnames:
        print(f"Course sheet '{course_name}' does not exist in
{attendance_file}.")
        return

    attendance_sheet = attendance_workbook[course_name]

    student_data = {}
    total_classes = 0
    current_class_index = 0

    for row in attendance_sheet.iter_rows(min_row=1,
values_only=True):
        if row[0] and isinstance(row[0], str) and
row[0].startswith("Date:"):
            total_classes += 1
            current_class_index += 1
            continue

        if row[0] == "No." or row[0] is None:
            continue

        student_name = row[2]
        student_id = row[3]
        total_time = row[7]

        if student_name is None or student_id is None:
            continue

        key = (student_name, student_id)
        if key not in student_data:
            student_data[key] = {
                'total_time': 0,
                'classes_attended': 0,
```

```
                'class_count': 0
            }

        if total_time is not None and total_time != "N/A":
            try:
                total_time_value = float(str(total_time).rstrip('%'))
                student_data[key]['total_time'] += total_time_value
                student_data[key]['classes_attended'] += 1
            except ValueError:
                pass

        student_data[key]['class_count'] = current_class_index

    headers = ["No.", "Student Name", "Student ID", "Group", "Average
Total Time"]
    for col_num, header in enumerate(headers, start=1):
        course_sheet.cell(row=1, column=col_num, value=header)

    absent_fill = PatternFill(start_color='FFFF00',
end_color='FFFF00', fill_type='solid')  # Yellow for below 80%
    students_below_80 = []

    for idx, ((student_name, student_id), data) in
enumerate(student_data.items(), start=1):
        total_time = data['total_time']
        classes_attended = data['classes_attended']
        class_count = total_classes

        average_total_time = total_time / classes_attended if
classes_attended > 0 else 0
        group = f"{classes_attended}/{class_count}"

        average_total_time_cell = course_sheet.cell(row=idx+1,
column=5, value=f"{average_total_time:.2f}%")
        if average_total_time < 80:
            average_total_time_cell.fill = absent_fill
            students_below_80.append(f"{student_name}_{student_id}")
```

```
            course_sheet.cell(row=idx+1, column=1, value=idx)
            course_sheet.cell(row=idx+1, column=2, value=student_name)
            course_sheet.cell(row=idx+1, column=3, value=student_id)
            course_sheet.cell(row=idx+1, column=4, value=group)

    overall_workbook.save(file_name)
    print(f"Overall attendance saved to {file_name}")

    if students_below_80:
        print("\nAttendance below 80% for overall semester:")
        for i, student in enumerate(students_below_80, start=1):
            print(f"{i}. {student}")
    else:
        print("\nAll students have 80% or more attendance.")

course_name = input("Enter the course name (e.g., UGEB3016 Final Year
Project 1): ")
calculate_overall_attendance(course_name)
```