FACIAL RECOGNITION AT GATED COMMUNITY

By Muhammad Hilmi bin Mirza Kelana

17003605

Dissertation report submitted in partial fulfillment of

The requirements of the

Bachelor of Technology (Hons)

(ICT)

JULY 2021

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Facial Recognition at Gated Community

By

Muhammad Hilmi bin Mirza Kelana

17003605

A project dissertation submitted to the

Information Technology Programme

Universiti Teknologi PETRONAS

In partial fulfilment of the requirement for the

BACHELOR OF INFORMATION TECHNOLOGY (Hons)

(IT)

Approved by,

_____

(Mr. Abdullah Sani Bin Abdul Rahman)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

September 2021

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

Muhammad Hilmi bin Mirza Kelana

Abstract

Facial recognition nowadays has been a popular choice in biometric authentication where it makes use of machine learning to perform its task of recognizing faces whether it be in a public or private property. Machine learning in the facial detection context can be trained to recognize faces by using an algorithm for it to perform its task such as training and testing process for accuracy of face recognition. YOLO algorithm has been proven to be popular in object detection with its capability of real-time detecting multiple objects in a fast and accurate manner. Though detecting face can also be a challenge depending on the situation such as wearing a mask and sunglasses that generally block faces from being scanned. Effective training needs to be conducted for it to perform as intended. In the past, the method of face detection lacks accuracy where it yields a poor result. This paper focuses on whether YOLO is capable of processing facial detection and using dlib library as facial recognition as authentication method in the context of the problem statement by testing and doing a comparison with Raspberry Pi 4B and a desktop that acts as the machine. This paper will also share Raspberry Pi 4B's capabilities in conducting facial recognition in terms of speed and accuracy that is if the purpose can be fulfilled.

Acknowledgment

I would like to thank Universiti Teknologi Petronas (UTP) for giving me the opportunity, resources, and environment to learn on the wonderful course of Information Technology. Without obtaining this knowledge, this project should not be possible. I would also like to express my gratitude towards my Supervisor Ts Sani Abdullah bin Abd Rahman for supervising me and giving recommendations upon this project as without his supervision, the project would be difficult upon conducting the research and developing the project

I would also like to extend my deepest gratitude towards my family and friends that supports me, give opinions, and spend time as well as resources for the project as this project would not be possible without them.

Table of Contents

Table of Figures

Chapter 1: Introduction

1.1 Background

In recent years, House break-in and theft cases had been increasing and one of the countries that I would like to take in is as an example of this research is Malaysia as it is the closes resource that can be obtained and easier to do research on. Based on the Crime Statistic rate that is being provided by the *Polis Diraja Malaysia* from 2018 and 2019 showed that House break-in and theft had been one of the few major crimes that and continues to haunt the community security (*Crime Statistics, Malaysia, 2018*, 2018; *Crime Statistics, Malaysia, 2020*, 2020). To decrease the number of crime cases, facial recognition must be utilized to increase the security aspect inside the gated community in Malaysia.

1.1.1 Privacy concern

Privacy has been one of the topics that concern people the most especially with the implementation of Facial recognition technology that may invade people's privacy. The sole idea of being recognized and tracked in a private area by an authority for their benefits is what kept people protesting since it is being done without consent that results in an invasion of privacy. With the Malaysia Personal Data Protection (PDP) Act 2010 being introduced, the policy of processing user's data has been more defined where if their purpose is by processing personal data is not for the benefit of the "data processor" then it is considered legal based on section 4.

To avoid having issues when processing personal data, the intention, and benefits of the "data processor" should be transparent with those whose data is being used. Section 9 of the PDP act 2010 defined more on the security principle when processing Users personal data and from here it can conclude that the usage of personal data is legal except for the purpose of misuse of data, modification, unauthorized or accidental access or disclosure and alteration or destruction of data. With that in mind, initially the consent form should include everything that describes the process that involves in personal data and as well as the benefits for them to comply with the policy.

1.1.2 Implementation

Facial recognition technology is reliable compared to the traditional CCTV monitoring method where it requires human to run the process of recognizing potential threats and identifying human behavior. Making use machine for facial recognition has been far common in this 21st century where it is being use for security purposes such as authentication and identification of people. This technology can be seen and utilize at international airport, for homeland defense, and even schools (Andrejevic & Selwyn, 2020; Anusha et al., 2020). In this scenario, facial will be located specifically at the gate of the community where it monitors the incoming and outgoing traffic.

1.1.3 Overview of the algorithm

The facial recognition will utilize on YOLO algorithm for training and testing, and facial detection. YOLO algorithm is a state-of-the-art deep learning framework that is famous for detecting object fast in real-time (Garg et al., 2018). YOLO algorithm is also based on Convolutional Neural Network algorithm which is a popular method for facial recognition and fast image processing. Face detection should be in a fast manner where YOLO can be put in use for its benefits fast object detection. YOLO framework is becoming drastically fast and accurate for detection with the help of Neural network (Garg et al., 2018). How its work is that from each images have a bounding box where it generates by region-based CNN to then run classifier with on those bounding boxes. The bounding boxes will later be refined with post-processing such as non-maximum suppression to eliminate duplication detection. To put in in comparison, a single CNN can predict multiple bounding boxes and class probabilities of objects (Garg et al., 2018). YOLO algorithm can achieve fast detection with the capability of optimizing its performance where a single neural network is applied onto each image during the training and test time. It encodes the information about the appearance and the classes.

1.2 Problem Statement

House break-in and theft had been one of the few major crime cases that challenges the security of gated community. With the existing security implementation, attackers have always found a way to exploit this security and thus results in house break in.

There can be many factors that can contribute towards this crime case. With the absence of security protocol and poor security implementation that does not cover every aspect of the environment, it will open towards a lot of possibility that can harm the residents inside the community. Usually with the existing security protocol inside a gated community, outsiders are prompt to key in their personal information such as phone numbers or Information Card (IC) and CCTV is being used to record and monitor daily activity. Some of which uses RFID as an authentication as the security protocol. While commonly there is RFID access, CCTV, and security guard, it is rarely that facial recognition is being in to use as authentication or recognition at gated community. In an effort to decrease the crime case rate, applying this technology will help to improve the security of gated community to be better on recognizing residents and deterring attackers.

1.3 Scope of study

This primary focus of the study is to utilize the facial recognition technology at the gate of the community for the purpose of authentication of residents and monitoring the traffic incoming and outgoing people. As such method of facial recognition and facial detection which rely on deep learning algorithm to be able to identify residents between outsiders. In other words, is to research on how and what machine learning algorithm to use that is efficient and accurate for facial recognition for security biometric authentication. The accuracy may differ on which type of algorithm and dataset used. The parts that is necessary to do research in order to understand how facial recognition processes the data by learning the process and architecture of facial detection to find suit for fast facial detection and training algorithm for the machine to perform image processing. Deep learning, which is a subfield of the machine learning that introduces varieties of algorithms and method of training such as You Only Look Once (YOLO). YOLO algorithm is going to be used for facial detection while dlib facial recognition library is used for image classification in this research. As such, I will do research on which is architecture if best suit in problem statement scenario. Languages like python opens a lot of flexibility to interact with the camera and machine itself and with this, logging the traffic should be possible to monitor the traffic.

1.3.1 Objective

The objective of this research is as follows.

- To be able to capture images from live camera and save inside a folder
- To be able to differentiate between residents and outsider/visitor
- To be able to log the face traffic going inside and outside of the community

The first objective test on whether the facial detection method is able to capture and save for image classification or not. Furthermore, the images being saved by OpenCV should also be used for logging purposes. The second objective is defined as to the machine being able to differentiate between the residents and outsiders with the dataset of residents being trained by the machine. If proven to be successful, the third objective will easier for the machine to classify by logging the traffic of who is going inside and outside of the community. It is often that facial recognition being applied to Raspberry

Pi 3 where it prove to be reliable (Sutabri et al., 2019; Syafeeza et al., 2020) but to apply Deep learning method such as YOLO could be a challenge.

Chapter 2: Literature review

This section will discuss on relevant literature that will help upon conducting this research. With further examination, each authors uses various method of facial recognition.

2.1 Facial recognition algorithms

Through the topic of facial recognition, the popular methods are Convolutional Neural Network (CNN) which is a state-of-the-art Deep Learning algorithm. CNN are a type of neural network that is quite similar to regular neural networks. To be able to understand on how YOLO algorithm generally works, we should also understand how CNN processes to get its output since YOLO architecture is based on CNN. CNN built up of neurons with weights and biases that can be learned. Each neuron gets some inputs, does a dot product, and then executes a non-linearity if desired. From the raw picture pixels on one end to class scores on the other, the entire network still represents a single differentiable scoring function (Kamencay et al., 2017).

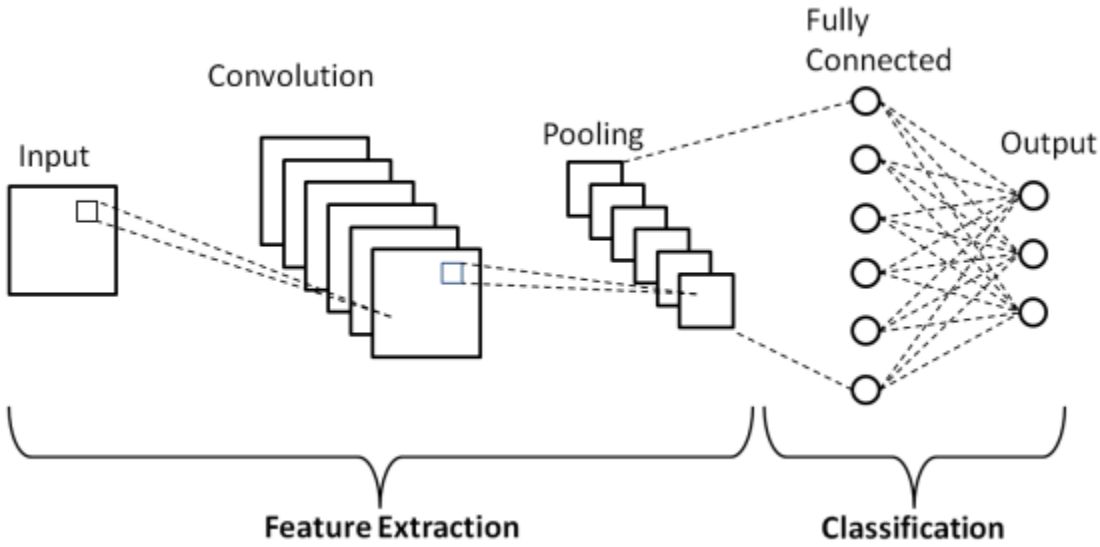

Figure 1 Convolutional Neural Network

The CNN is made up of several levels. Each layer receives a multi-dimensional array of numbers as input and outputs a second multi-dimensional array of numbers (which then

becomes the input of the next layer). When categorizing pictures, the first layer's input is the input image (3232), while the last layer's output is a collection of likelihoods for each category (i.e., 1 1 10 numbers if there are ten categories). A basic CNN is made up by layers, each of which uses a differentiable function to convert one volume of activations to another (Kamencay et al., 2017).With the CNN architecture being explained, Winarno et al. (2019) uses a completely different approach in facial recognition. Instead, CNN is being used to construct from 2D images that is being captured to create 3D facial construction where with it, Principal Component Analysis (PCA) which is a different algorithm uses the 3D facial construction for feature extraction. With the feature of faces being extract, only then it will do a comparison with database using Manhalanobis which is a distance method of classification. In here, Winarno et al. (2019) also define his stages of the process of recognition where I will take reference to construct an optimal process of facial recognition. The process can be seen in figure 5 below.

YOLO algorithm on the other hand is also a deep learning algorithm based on CNN. Like CNN it generally detects object and does image processing as mentioned above, but YOLO can detect multiple of object and does image processing in real-time comparatively faster (Garg et al., 2018). The YOLO frameworks are getting increasingly quick and precise for detection with the aid of neural networks. For a limited collection of items, there is still a restriction. Object detection datasets are currently restricted in comparison to classification and tagging datasets (Garg et al., 2018). Thousands of pictures with tags that are object coordinates in the image make up the object detection databases. Millions of pictures with classifications make up the categorization databases. Garg et al. (2018) proposed architecture takes input as color image with the specific size of 448 x 448. The architecture consists of 7 convolutional layers with max pooling layer of size 2 x 2. After that, three fully connected layers are attached, and output layer followed by last fully connected. In my architecture I will impose a similar solution based on the proposed architecture by Garg et al. (2018) where it will have suitable number of convolutional layers along with the max pooling size.

2.2 Gated community

A several questions has been conducted with google form with 24 respondents being questioned regarding the implementation of facial recognition in a gated community. While 29.2% only do lived in a gated community, majority of the respondent agrees on having a surveillance system located at the community gate will help to reduce the crime rate by having a sense of monitoring on who is going inside or outside of the community and for authentication purposes.

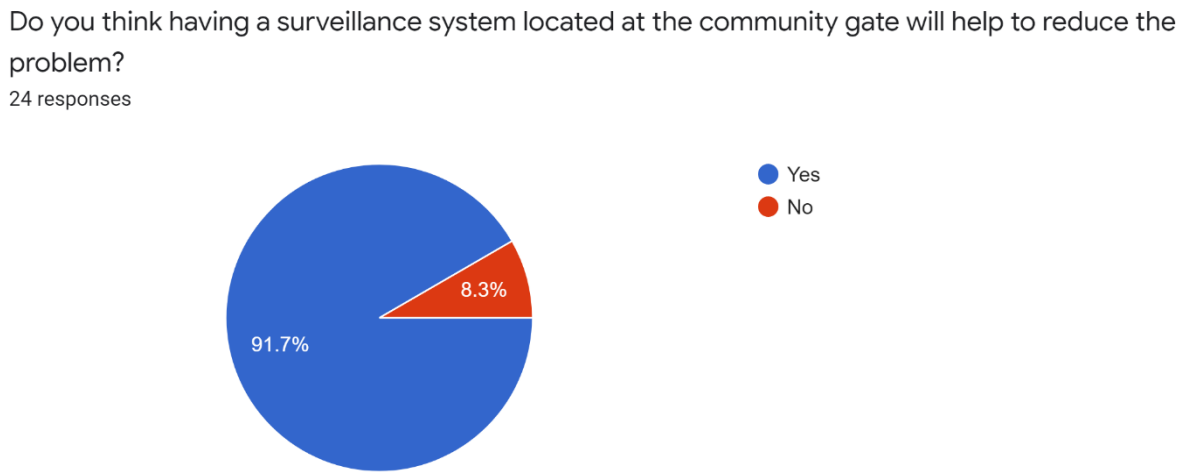

Figure 2 Surveillance system location question

Majority of the respondent also answered having to implement this facial recognition at the gate of the community is not a breach in privacy. This shows that the respondent is willing to sacrifice its privacy for the benefit of living in secured community. Not only that, consent of the respondent also being considered which an important part before implementing facial recognition while abiding the Malaysia Personal Data Protection Act 2010.

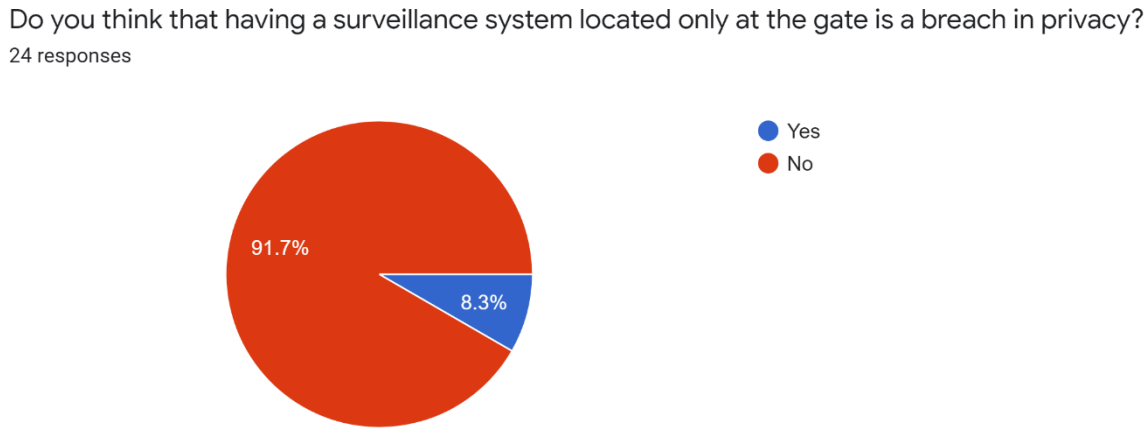

Figure 3 Surveillance system breach in privacy question

2.3 Raspberry Pi hardware implementation

Raspberry Pi product has been in popular use for its small sized computer yet capable of outputting decent power with hardware component. Many Authors such as Syafeeza et al. (2020) and, Sutabri et al. (2019) uses Raspberry Pi 3 to implement its facial recognition system with their respective context. With this, in theory facial recognition implementation would perform much better with Raspberry Pi 4B which is the successor for the Raspberry Pi 3 as it offers better Central Processing Unit (CPU) clock rate speed and Graphical Processing Unit (GPU) power.

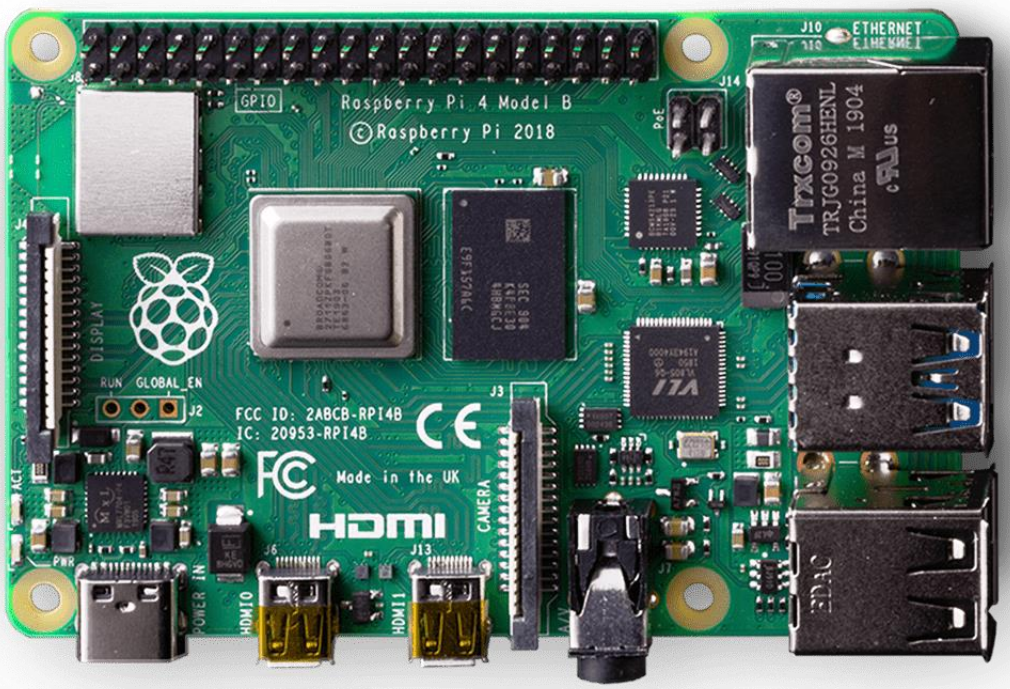

Figure 4 Raspberry Pi 4B (4GB)

Chapter 3: Methodology/Project Work

This chapter will define all the specific procedure and method on the research methodology, the stages of the process of facial recognition, and the training and testing dataset method to clarify on how the system should operate to solve the problem. To make sure the project run smoothly, every component and procedure that is being mentioned should be followed to ensure the objective is obtainable.

3.1 Stages of the process of facial recognition

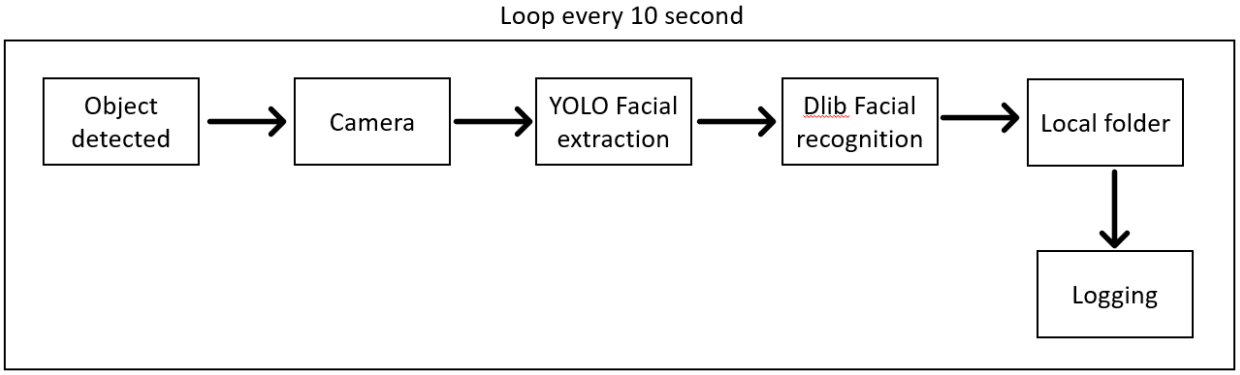

Figure 5 Stage of the process of facial recognition

Firstly, the camera will detect an object, with the help of YOLO algorithm it would be able to differentiate between a literal object and a face. YOLO algorithm is being put to use as facial detection as the framework itself are proven to be fast and accurate for detection (Garg et al., 2018). The camera will be connected via USB 3.0 port towards the main computer while the raspberry pi 4b will use CSI camera being connected directly towards the 2-lane MIPI CSI camera port. Webcam uses CPU power to be able to function where it might bottleneck its performance for it to perform faster decision-making processes such as comparison and classification of images while running its module. Therefore, authors would very much prefer to use GPU to power the Webcam by using Nvidia CUDA cores if the CPU being used has lack of processing power to handle the process. Since the provided GPU for the desktop and Raspberry Pi 4B is not CUDA compatible where it cannot possibly use GPU cores to handle the process, this thesis will be using CPU instead. Through the camera, YOLO algorithm will capture an image every 10 second from the live camera feed and pass it to dlib library to do facial feature extraction from the initial images being captured from the

facial detection process with the YOLO model. After the dlib library determine whether the person is resident or an outsider, the data of residents will be stored within the local storage as a form of log to record the traffic.

3.2 Hardware review

In this section, I will be discussing on the hardware comparison between Raspberry Pi 3 and Raspberry Pi 4. From the literature review section, there are several authors that uses Raspberry Pi 3 product where none that uses Raspberry Pi 4. Since Raspberry Pi 4 is a new product that was release in 2019, there is some concern on whether it is compatible with the technology or not.

Technically by comparing the hardware, Raspberry Pi 4B features Broadcom BCM2711 with 1.5 GHz of clock rate which packs the latest and faster clock speed compared to the Raspberry Pi 3 B+ which features Broadcom BCM2837B0 with 1.4 GHz. Faster in terms of clock rate is always better for processing and finishing task. Other than that, Raspberry Pi 4B had also feature VideoCore VI which is the successor for previous GPU version from Raspberry Pi3 VideoCore IV. VideoCore VI has better processing power which in theory would handle object detection process by learning better than the previous GPU hardware version. With Raspberry Pi 4B in theory it would handle much better than the predecessor where it offers the latest mobile hardware that can theoretically results in faster and accurate results. The specification of the Raspberry Pi 4B can be shown below.

- Broadcom BCM2711, 4 cores
- 8GB LPDDR4-3200 SDRAM
- Raspberry Pi 8MP NoIR Camera board v.2.1

On the other hand, a desktop (PC) will also be used to test the facial recognition system that is packed with more powerful CPU and larger RAM size in comparison with Raspberry Pi 4. The specification of the pc is as shown below. This desktop will be the main computer to test and run the facial recognition system.

- Intel i5 8400 4Ghz, 6 cores, 6 threads
- 16 GB DDR4

- Logitech Webcam C920

3.3 Model weights and Dataset

This section will be explaining more on the weights of the model that is being used to create the model. This pre-trained model used a dataset from WIDER FACE that was posted in their website published by Yang, Shuo and Luo, Ping and Loy, et.al (2016). The dataset featured 32,203 images and 393,703 faces being labeled with a high degree of variability in scale, pose and occlusion. The WIDER FACE dataset is organized based on 61 event classes as shown in figure 6 where each of the event class, the author randomly selects 40%/10%/50% data as training, validation and testing sets. The dataset has various scenarios and angles of faces to assist the training of the model to determine the facial structure and features. The images being used have faces from different ethnic and cultural background.

The model (weight file) itself is heavy with a size of 234 MB that is trained based on YOLO version 3 architecture based on the figure 6 below. Figure below shows the number of layers that is being deployed to further enhance the feature extraction. The more feature being extracted from the image, the better results it gets in term of the accuracy. After Training and testing the data, the model will come out in a form of ". weight" format of file where it be used by YOLO to detect faces.

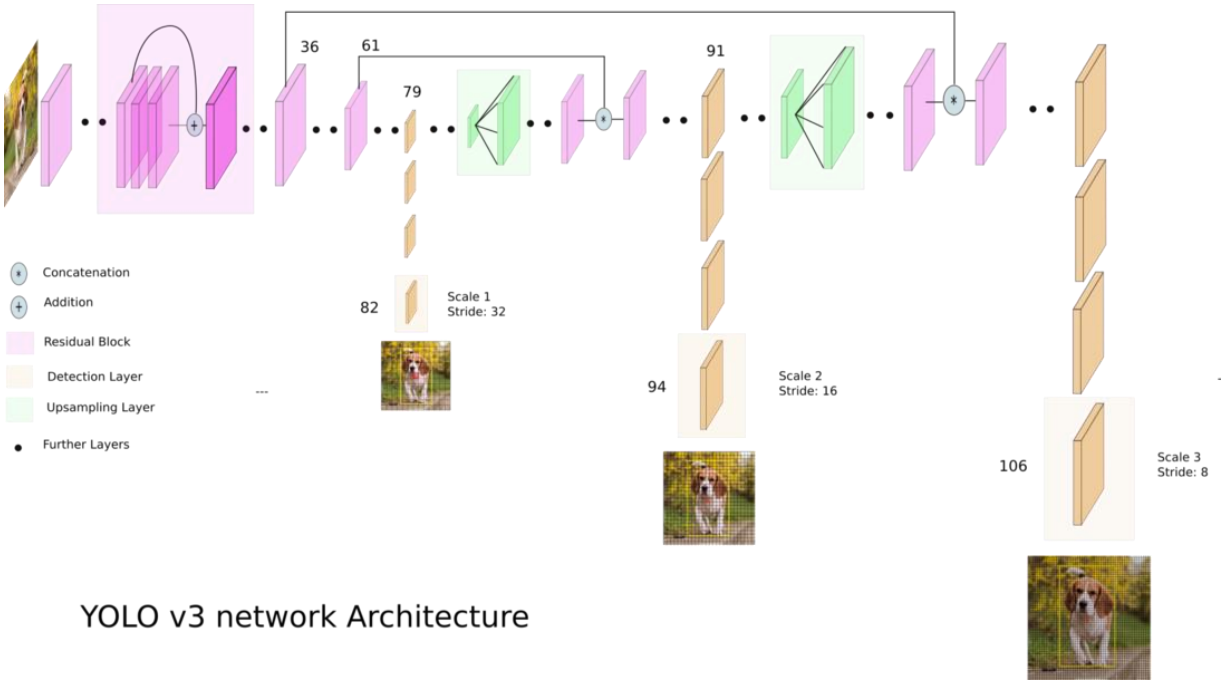

Figure 6 YOLO v3 network architecture

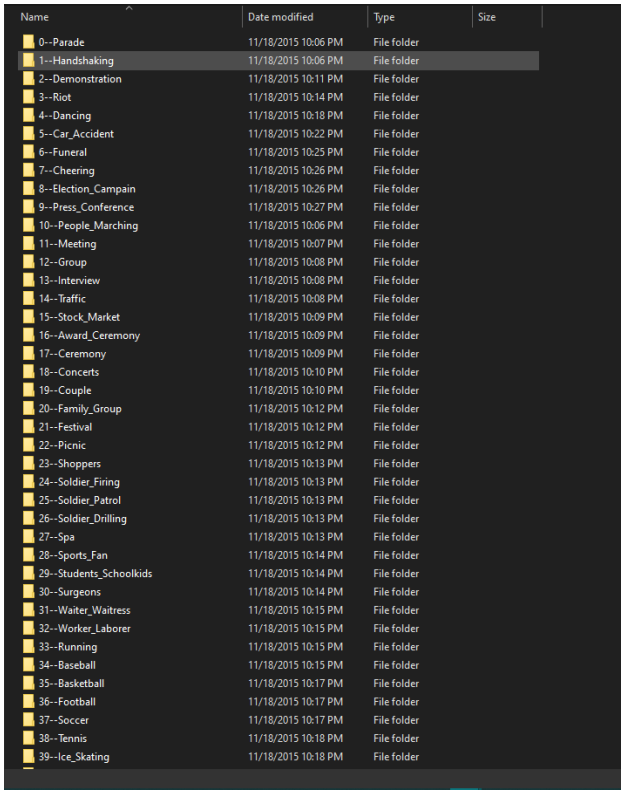

Figure 7 Images being organized with 61 event classes

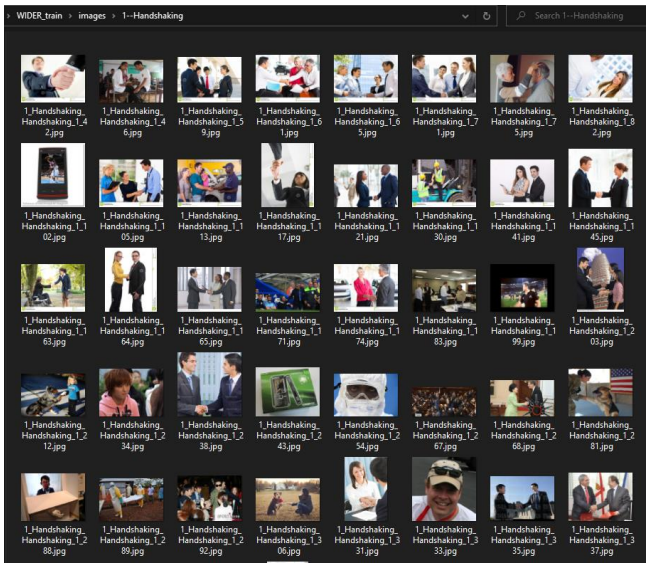

Figure 8 Images inside handshaking folder

3.4 Project Methodology

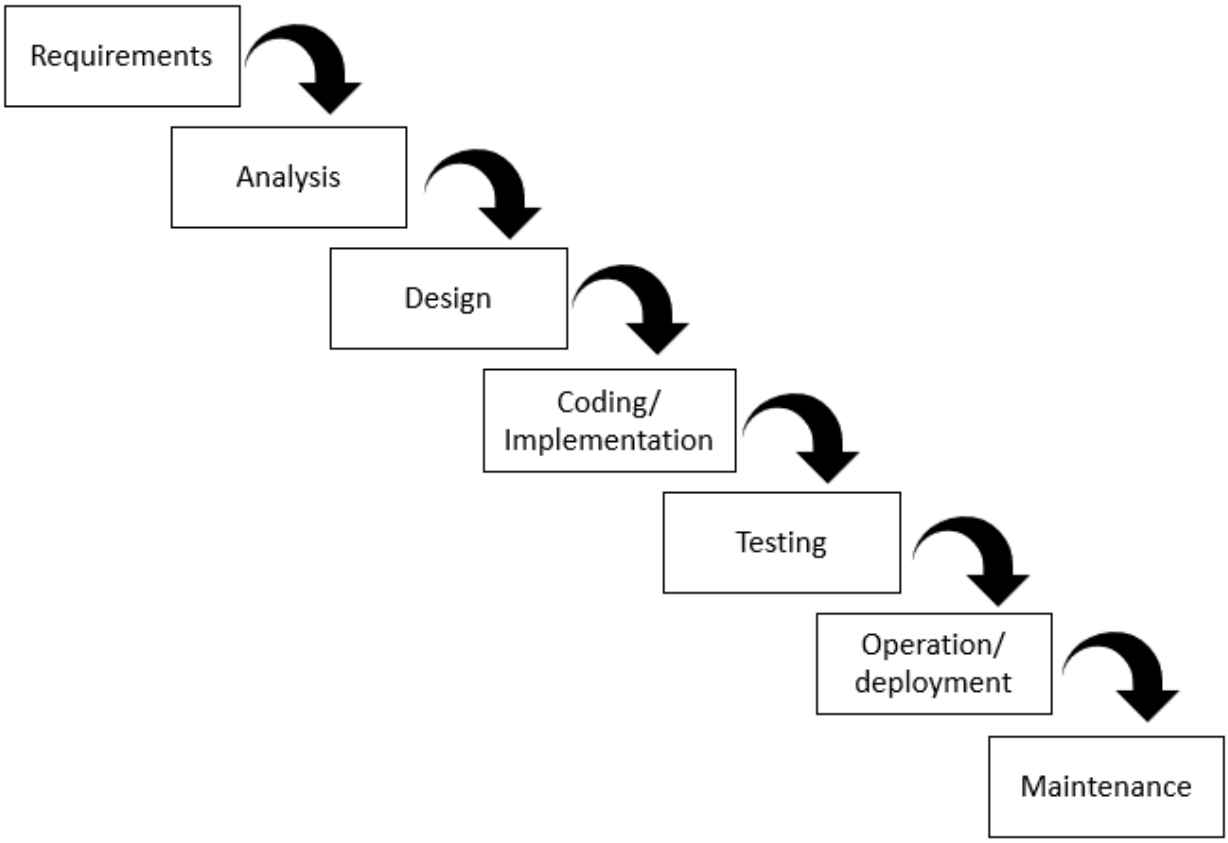

Figure 9 Waterfall methodology

The overall project will follow on Waterfall methodology where it is perfect in this specific context that will use machine learning to benefit the security aspect in gated community. Initially the requirements will gather the consent of the residents, the physical environment of the gate, and the requirements to build the facial recognition system and product. The tools can be defined on section 3.5 where I explain on what tools and library to use to achieve the objective. The analysis section will use all the data and information from the requirement phase where it will determine the best decision to choose. The designing phase should output the physical and the system design to choose the best and optimal structure for the facial recognition implementation. After the coding or implementation phase is done, the follow up is testing phase where it tests the theory of the design. If it be proven successful then the product should be put on deployment or operation where the product should also be

maintained to get the latest data of the residents, keeping the product functional, and optimal.

3.5 Tools and its usage

- Keras Library
- Visual Studio
- Python
- Face recognition (dlib)
- OpenCV

Mentioned above are the tools that is going to be used to achieve objective of this project. Firstly, YOLO library is essential as the project will be utilizing YOLO algorithm to detect object such as faces. YOLO will use Keras library to mainly use the YOLO model and generate output tensor targets for filtered bounding boxes. And of course, the project will solely code with python language as it is a common language among programmers that relates with machine learning. OpenCV is a well-known library developed and acquired by Intel corporation where developers choose to use real-time operation function to do image processing, video processing and real-time object detection.

```
self.input_image_shape = K.placeholder(shape=(2,))
boxes, scores, classes = eval(self.yolo_model.output, self.anchors,
                              len(self.class_names),
                              self.input_image_shape,
                              score_threshold=self.args.score,
                              iou_threshold=self.args.iou)
return boxes, scores, classes
```

Figure 10 Keras used for YOLO model

OpenCV is essential towards this project to run camera and image processing. The snippet below shows on how it uses the OpenCV library to run image processing based on image, video, and camera (webcam). Based on the defined argument it will call upon each directory when being called to do the facial recognition and pass it through dlib library. For webcam it will call the source argument and run the camera along with YOLO to capture image based on the bounding boxes.

```
def _main():
    wind_name = 'face detection using YOLO'
    cv2.namedWindow(wind_name, cv2.WINDOW_NORMAL)

    output_file = ''

    if args.image:
        if not os.path.isfile(args.image):
            print("[!] ==> Input image file {} doesn't exist".format(args.image))
            sys.exit(1)
        cap = cv2.VideoCapture(args.image)
        output_file = args.image[:-4].rsplit('/')[-1] + '_yoloface.jpg'
    elif args.video:
        if not os.path.isfile(args.video):
            print("[!] ==> Input video file {} doesn't exist".format(args.video))
            sys.exit(1)
        cap = cv2.VideoCapture(args.video)
        output_file = args.video[:-4].rsplit('/')[-1] + '_yoloface.avi'
    else:
        # Get data from the (source) camera
        cap = cv2.VideoCapture(args.src)
```

Figure 11 Main function

The initial process is to use OpenCV to run webcam with YOLO to detect any faces being presented in front of the camera. How it works is every 10 second interval, it will capture images when it detects any faces. Code from the figure 12 below shows that every 10 second it will call upon a function that sends the capture image and bounding box value to dlib library.

```
if (index%30==0):
    # Multithread function so it does nopt interfere with the live video
    Thread(target=resident_recognition, args=(faces, frame)).start()
```

Figure 12 Calling function every 10 second

The bounding boxes value is defined by YOLO algorithm where it has 4 values which are x, y, width, height and it will pass to a function called *resident_recognition* that uses dlib to recognize the capture faces. This function will utilize facial recognition library to determine whether the capture image is a resident, or an outsider based on the images folder.

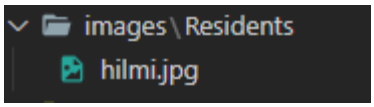

Figure 13 Residents faces storage

The images folder contain only residents faces where the library will compare from the images folder to the current picture captured. After that the log will be in a form of text named after the capture image based on figure 14 and 15. The name format is followed by image counter, categorization, date (dd,mm,yyyy), and time (hh,mm,ss).

```python
def resident_recognition(bboxes,image):
    # Declare initial variables and get list of local images
    path = 'images/Residents'
    images = []
    myList = os.listdir(path)
    print(myList)
    for cl in myList:
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)

    # Get encoding list of the local images
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)

    # Extract faces from bounding boxes from YOLO
    index = 0
    while index<len(bboxes):
        # Get each individual bounding box values
        x = bboxes[index][0]
        y = bboxes[index][1]
        w = bboxes[index][2]
        h = bboxes[index][3]
        # Extracts image inside bounding boxes
        faceImage = image[y:y+h, x:x+w]
        # Converts image color format to RGB
        imageS = cv2.cvtColor(faceImage, cv2.COLOR_BGR2RGB)
        # Gets face locations and encodings for current image
        facesCurFrame = face_recognition.face_locations(imageS)
        encodesCurFrame = face_recognition.face_encodings(imageS, facesCurFrame)
        # Sets residential status for each face
        status="UNKNOWN"
        # Iterates through face encodings and locations and checks for matches to the local images
        for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(encodeList, encodeFace)
            faceDis = face_recognition.face_distance(encodeList, encodeFace)
            matchIndex = np.argmin(faceDis)
            if matches[matchIndex]:
                status="RESIDENT"
                print("FACE {} IS A RESIDENT".format(str(index)))

            else:
                status="OUTSIDER"
                print("FACE {} IS AN OUTSIDER".format(str(index)))
        date_now = datetime.datetime.now().strftime("%d %m %Y %H %M %S")
```

Figure 14 resident_recognition function

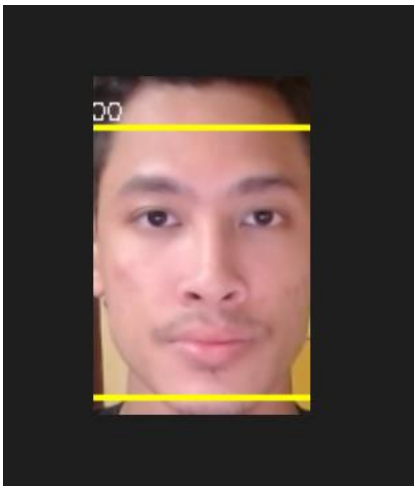

Figure 15 captured image

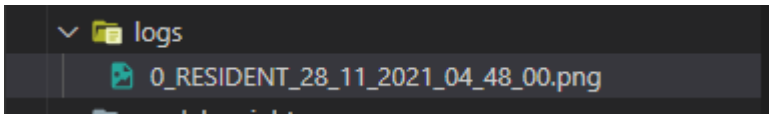

Figure 16 log stored by image

Visual Studio Code

Visual Studio Code is a popular choice by programmer on compiling or code editing to do debugging, task running and etc. This code editor will be my choice as I find the User Interface is easier compared to another compiler like Spyder and Anaconda. Visual Studio Code also supports dependencies installation interface to use certain library. Personally, Visual Studio Code have a lot of quality of life that will eases the process of developing and debugging the project.

Chapter 4: Results and Discussion

This section will present and discuss upon the results that was given throughout the development of the project.

4.1 Raspberry Pi 4b

The results of testing and running YOLO model with Raspberry Pi 4B is not practical in the real-world scenario due to the reason being.

- Insufficient CPU power to run YOLO model

As mentioned before, the YOLO pre-trained model for object detection is heavy where Raspberry Pi 4B could not handle or afford to run the model efficiently. The camera being used with to run with YOLO model for object detection runs at 0.02 frames per second (FPS) even with 100% CPU usage running at 1.5 GHz. This put the Raspberry Pi 4b in a major disadvantage if it were to run within real world scenario. There is however a slight improvement when I did try to overlock the CPU to 2.5 GHz. The FPS did improve to 0.009 in which I did tried to push its performances compared to the default CPU clock rate. But the effort came to waste whereby it is still running bellow 1 frames per second therefore the Raspberry Pi 4B is not practical to run facial detection as it takes a lengthy number of seconds to identify faces. Not only that, but the Raspberry Pi 4b also output hot temperature when running the model where this shorten the lifespan of the components if not being cooled properly. Raspberry Pi tends to be kept within a confined space in practical use as seen in figure 16 whereby if it were to be applied towards Raspberry Pi 4B, it will crash often and damages other onboard hardware component during its lifespan.

Figure 17 Raspberry Pi 4B case

4.2 Desktop Computer

The desktop computer on the other performed decently with YOLO object detection model along with facial recognition. Initially, the I will call the webcam from the terminal by writing the code below.

```
PS C:\Users\hilmi\Downloads\yoloface-master\yoloface-master> python yoloface.py --src 0
```

Figure 18 Terminal code to run webcam on gaining input

This will call in the live camera using OpenCV and run the code for facial detection using the pre-trained model. The terminal will feed me with information on the bounding box location and how many frames have passed as shown below. The index value represents the frame, the faces represent the bounding box coordination faces detected counts how many faces are within the camera.

```
PS C:\Users\hilmi\Downloads\yoloface-master\yoloface-master> python yoloface.py --src 0
----- info -----
[i] The config file:  ./cfg/yolov3-face.cfg
[i] The weights of model file:  ./model-weights/yolov3-wider_16000.weights
[i] Path to image file:
[i] Path to video file:
######################################################

==> Skipping create the outputs/ directory...
[ERROR:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-1i5nllza\opencv\modules\videoio\src\cap.cpp (589) cv::VideoWriter::open VIDEOIO(CV_IMAGES): raised OpenCV exception:

OpenCV(4.5.3) C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-1i5nllza\opencv\modules\videoio\src\cap_images.cpp:253: error: (-5:Bad argument) CAP_IMAGES: can't find starting number (i
n the name of file): outputs/ in function 'cv::icvExtractPattern'


FACES:  [[291, 259, 91, 121]]
[i] ==> # detected faces: 1
3
4
######################################################
['hilmi.jpg']
index:  0
FACES:  [[291, 265, 94, 107]]
[i] ==> # detected faces: 1
3
4
######################################################
index:  1
FACES:  [[291, 266, 91, 105]]
[i] ==> # detected faces: 1
3
4
######################################################
index:  2
FACES:  [[294, 268, 90, 104]]
[i] ==> # detected faces: 1
3
4
######################################################
index:  3
FACES:  [[295, 268, 89, 105]]
[i] ==> # detected faces: 1
3
4
######################################################
index:  4
FACES:  [[294, 268, 90, 106]]
[i] ==> # detected faces: 1
3
4
######################################################
index:  5
FACES:  [[293, 269, 91, 104]]
```

Figure 19 Terminal feed

After 30 frames had passed which is roughly around 10 second, the terminal will tell if the faces detected is resident or not such as in figure 18 below. This is an improvement

compared to Raspberry Pi 4B since the desktop provide more processing power. The FPS is calculated around 3 frames with the desktop where it uses 100% CPU usage to run the YOLO model. And of course, the better the CPU, the better the results. This can not only be improved by changing into better CPU but can also better if the model utilizes CUDA cores from Nvidia GPU. As for now, I have not been provided with such GPU that supports CUDA enable feature and therefore I am using CPU for this sake of project.

```
############################################################
['hilmi.jpg']
index:  30
FACE 0 IS A RESIDENT
FACES:  [[279, 200, 102, 140]]
[i] ==> # detected faces: 1
```

Figure 20 Terminal feed when it reaches 10 second

4.3 Limitation of Facial recognition using dlib

The facial recognition featured in dlib library haves its limitation. It does however only work in certain scenario whereby it needs to be a in a good condition for the facial recognition to work. Facial recognition process can be affected by lights, face angle and object being far away from the camera. Sure, the YOLO model can detect faces from far away but when being captured in a distant position, the facial recognition would not be able to recognize whether it is a face or not. Other than that, lights also play a major role in this scenario whereby if the capture images are dark, the facial recognition function can not function correctly and label its faces as “unknown” or an “outsider” such as displayed below. Lasty, the angle of faces can also affect the facial recognition results as the way the library work, the library did a sketch and find location of your mouth, nose, eyes, and chin and compare with existing data to output results in figure 23. In this scenario, the face is being tilted sideway where the library cannot recognize and thus labeled this is unknown object.

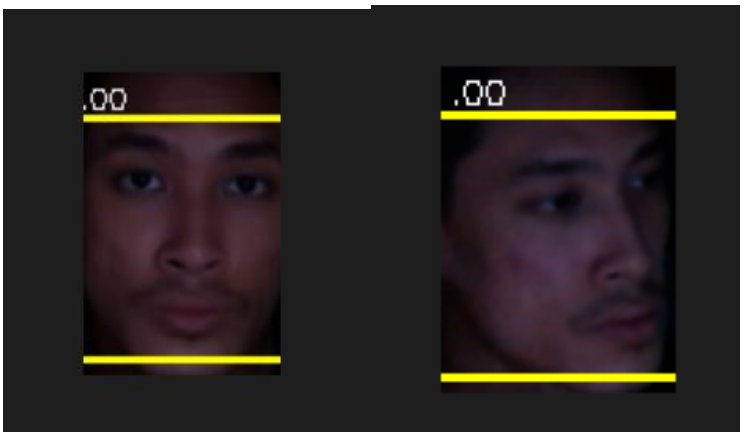

Figure 21 Lighting and face angles label as unknown

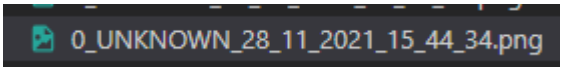

Figure 22 Unknown faces labeled

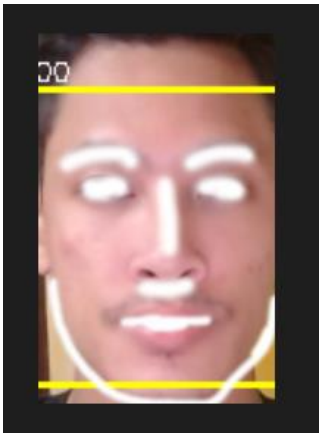

Figure 23 rough sketch of dlib face recognition

To conclude the test, the desktop had run more efficiently compared to Raspberry Pi 4B as the desktop provided more processing power to run facial detection and facial recognition. Raspberry Pi 4B is incapable of running the heavy YOLO model due to its limited hardware capabilities such as its CPU. Therefore, the implementation decided to run this on a desktop where it is more efficient to run the facial detection and facial recognition process.

Chapter 5: Conclusion and Recommendation

5.1 Conclusion

To conclude the project, the initial plan of implementing the facial detection and recognition system in Raspberry Pi 4b proves to be non-efficient and not practical in live scenario due to its limitation of hardware capabilities outputting powerful processes. YOLO model requires a lot of power to run the and therefore it is more suitable to run it on a desktop with decent hardware such as the CPU or GPU and RAM. Nonetheless, the project is able to reach three of the objectives which are to recognize residents and non-residents, log the events, and capture images from live camera and save it within the folder. YOLO is more than capable of just handling facial detection but in return required more processing power to do so. This project proves that using YOLO outputs or results in fast object detection and can accurately pinpoint faces even in such scenarios. The facial recognition from dlib can also distinguish between resident and an outsider. Furthermore, the program also be able to log the traffic of faces with images. Overall, the project can be proven success however it is not in way perfect which I will be talking about in recommendation.

5.2 Recommendation

There is multiple implementation that can further improve this project. It is to add a better model or by using another model to further enhance the images taken for facial recognition. Firstly, YOLO algorithm had multiple of version. There is YOLOv4 -Tiny where it is made to run in mobile suited hardware. YOLOv4-Tiny is the compressed version of YOLOv4 designed to train on machines or computer that have less computer power. In theory it can run on Raspberry Pi 4B, but the result of the project says otherwise. Another small and compact machine that packs power is Nvidia Jetson Nano from Nvidia. The specification is far better in terms of outputting GPU power compared to Raspberry Pi 4B where it is also CUDA enabled. Since Raspberry Pi 4B is not capable of running the YOLO model with its GPU and runs poorly with its CPU, Nvidia Jetson nano is capable and will run better than the latter CPU. Which in theory, Jetson Nano will run faster compared to Raspberry Pi 4B.

Other than that, as of for this project the YOLO model runs specifically to detect faces. While it does detect faces fast and accurately, the model can however be trained

to detect and recognize faces. This will reduce the usage of using dlib library where it is at best to avoid due to its limitation of recognizing faces from afar. But the catch is that it will require a lot of processing power to run. Therefore, if this is possible it may need to run-on high-end hardware to run it efficiently to obtain accurate data.

Moreover, the recent research theme has been around in depixelization of images that improves the image quality by removing the pixels from the source image. This process can be implemented within the project facial recognition process. The reason being is that when YOLO algorithm detects faces from afar, the images being captured will send a heavy pixelated images towards dlib where it cannot recognize whether it is a face or not. By putting this process of depixelization after YOLO detects faces, it will further enhance the pixelated image and act as a compliment towards dlib to run facial recognition.

In terms of User Interface where it is close nonexistence on this project, the facial recognition could also use UI to visualize the process of facial detection and facial recognition towards users. UI can also be made using python UI where it is more feasible and easier to apply towards the project.

References

Andrejevic, M., & Selwyn, N. (2020). Facial recognition technology in schools: critical questions and concerns. *Learning, Media and Technology*, *45*(2), 115–128. https://doi.org/10.1080/17439884.2020.1686014

Anusha, P., Prasad, K. L., Kumar, G. R., Lydia, E. L., & Subbiah, V. (2020). *FACIAL DETECTION IMPLEMENTATION USING PRINCIPAL COMPONENT ANALYSIS (PCA)*. *7*(10), 1863–1872.

Crime Statistics, Malaysia, 2018. (2018, December 28). DEPARTMENT OF STATISTICS MALAYSIA OFFICIAL PORTAL. https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=455&bul_id=SnJlWjNGZ3VWajUraDlBcFpMQ3JWUT09&menu_id=U3VPMldoYUxzVzFaYmNkWXZteGduZz09

Crime Statistics, Malaysia, 2020. (2020, November 20). DEPARTMENT OF STATISTICS MALAYSIA OFFICIAL PORTAL. https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=455&bul_id=UFZxVnpONEJqUU5pckJIbzlXeEJ1UT09&menu_id=U3VPMldoYUxzVzFaYmNkWXZteGduZz09

Garg, D., Goel, P., Pandya, S., Ganatra, A., & Kotecha, K. (2018). A Deep Learning Approach for Face Detection using YOLO. *1st International Conference on Data Science and Analytics, PuneCon 2018 - Proceedings*, 2–5. https://doi.org/10.1109/PUNECON.2018.8745376

Kamencay, P., Benco, M., Mizdos, T., & Radil, R. (2017). A new method for face recognition using convolutional neural network. *Advances in Electrical and Electronic Engineering*, *15*(4 Special Issue), 663–672. https://doi.org/10.15598/aeee.v15i4.2389

Sutabri, T., Pamungkur, Kurniawan, A., & Saragih, R. E. (2019). Automatic attendance

system for university student using face recognition based on deep learning. *International Journal of Machine Learning and Computing*, *9*(5), 668–674. https://doi.org/10.18178/ijmlc.2019.9.5.856

Syafeeza, A. R., Mohd Fitri Alif, M. K., Nursyifaa Athirah, Y., Jaafar, A. S., Norihan, A. H., & Saleha, M. S. (2020). IoT based facial recognition door access control home security system using raspberry pi. *International Journal of Power Electronics and Drive Systems*, *11*(1), 417–424. https://doi.org/10.11591/ijpeds.v11.i1.pp417-424

Y.S.L.P.L.C.C.T.X. (2016). *WIDER FACE: A face detection benchmark*. WIDER FACE: A Face Detection Benchmark. Retrieved 2021, from http://shuoyang1213.me/WIDERFACE/index.html