DESIGN AND IMPLEMENTATION OF SMART WIRELESS CAMPUS NETWORK BASED ON LORA AND WI-FI

LEONG RONG CHUAN

UNIVERSITI TUNKU ABDUL RAHMAN

DESIGN AND IMPLEMENTATION OF SMART WIRELESS CAMPUS NETWORK BASED ON LORA AND WI-FI

LEONG RONG CHUAN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronic Engineering with Honours

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2023

DECLARATION

Signature	:	chuan
Name	:	LEONG RONG CHUAN
ID No.	:	1803522
Date	:	16/05/2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DESIGN AND IMPLEMENTATION OF SMART WIRELESS CAMPUS NETWORK BASED ON LORA AND WI-FI"** was prepared by **LEONG RONG CHUAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronic Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Ir Ts Dr Tham Mau Luen
Date	:	16 May 2023

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, Ir Ts Dr. Tham Mau Luen for his invaluable advice, guidance and his enormous patience throughout the development of the research. His expertise and wisdom have been invaluable in shaping and refining the ideas presented in this work.

In addition, I would also like to express my gratitude to my friends who provided me with unwavering support, encouragement, and assistance throughout this project. Their willingness to help and their belief in me have been a source of inspiration and motivation.

ABSTRACT

This project presents the development of a smart application that uses object detection and tracking algorithms to count the number of passengers on the UTAR bus. An additional package to recognize the gender of the passengers is provided. The system utilizes YOLO for object detection and DeepSORT for tracking, which runs on a Raspberry Pi 4 with Intel Neural Compute Stick (NCS) 2. The passenger count information is wirelessly transmitted back to the campus using LoRa and Wi-Fi technology. The proposed system offers real-time monitoring and analysis of passenger flow on the bus, providing insights to optimize bus schedules and routes. Furthermore, the system architecture can be extended to support other applications on the campus, creating a smart wireless network for the university. The accuracy of the person counting application using YOLOv4-tiny is 85 %, and the frames per second is 9.97. The accuracy for passenger gender recognition tested using YOLOv5 and YOLOv7 is 100 %. Besides, the overall performance of LoRa still has room for improvement due to the low packet delivery ratio. In addition, the performances of Wi-Fi wireless networking in terms of total packets received and latency are satisfactory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	artificial intelligence
AP	average precision
BoF	bag of freebies
BoS	bag of specials
CBN	cross batch normalization
CmBN	Cross mini-Batch Normalization
CNN	convolutional neural network
CR	coding rate
CSP	stage partial connections
CSS	chirp spread spectrum
DeepSORT	simple online and realtime tracking with a deep association metric
DL	deep learning
E-ELAN	extended efficient layer aggregation networks
EIRP	effective isotropic radiated power
FEC	forward error correction
FPS	frame-per-second
GPS	global positioning system
GPU	graphics processing unit
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of things
IoU	intersection over union
IR	intermediate representation
LoRa	long-range
LPWAN	low-power wide area network
LTS	long-term support
mAP	mean average precision
MCMC	Malaysian Communications and Multimedia Commission
MQTT	MQ Telemetry Transport
NCS2	Neural Compute Stick 2
NICT	National Institute of Information and Communications

	Technology
OpenVINO	Open Visual Inference and Neural Network Optimization
PR	precision-recall
R-CNN	region-based convolutional neural networks
SF	spreading factor
SORT	simple online and realtime tracking
SPP-block	spatial pyramid pooling-block
SSD	single shot detector
SSD	solid-state drive
TDMA	time division multiple access
VPU	vision processing unit
WLAN	wireless local area network
YOLO	You Only Look Once

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

A smart campus combines advanced network infrastructure, internet-connected sensors and devices, and data analytics to create an intelligent environment for students and staff (Martin, 2023).

Integrating various technologies, including Artificial Intelligence (AI) and the Internet of Things (IoT), in a smart campus aims to enhance the quality of life for university students and staff by optimizing the use of resources.

To become a smart wireless campus, the smart campus can utilise wireless technologies such as long-range (LoRa) and Wi-Fi. In a smart wireless campus, various smart applications can be deployed to improve the operational efficiency and safety of the university. NerveNet, a resilient mesh-topological network developed by Japan National Institute of Information and Communications Technology (NICT), is applied as a wireless network in this project.

The smart wireless campus in this project is an application to track and count the number of passengers going into or out of the university's bus. Afterwards, the information will be transmitted back to campus using LoRa and Wi-Fi technologies.

1.2 Importance of the Study

At the UTAR campus, the students face several challenges when using the bus services provided by the university. The challenges include the bus being too crowded and not on time. The proposed smart application has several potential benefits for the university community.

First, the smart application can track and monitor the number of passengers on the university's bus in real-time. The data collected can be used for another deep learning (DL) application. The DL application can predict the number of students onboarding the specific bus trip. With this prediction, the university can adjust the service accordingly.

Besides, the application can improve the safety and security of the students by ensuring the bus is not overcrowded. In addition, the application can enable the university to respond faster in case of emergencies by monitoring the real-time location of the bus.

In short, this smart application will improve the overall student experience and help the university to manage the bus system.

1.3 Problem Statement

The current system of the university does not provide passenger counting and gender detection. Under the current system, the bus is frequently overcrowded, and the student might need to wait for the next trip.

The smart application in this project can count the number of passengers on the bus. The real-time data will be transmitted back to campus using LoRa and Wi-Fi technologies. The student can check the number of passengers on the bus while waiting at the bus stop. If the bus is overcrowded, the student can make necessary adjustments, such as using another way to go to campus.

1.4 Aim and Objectives

The aim of this project is to design and implement a smart wireless campus network based on LoRa and Wi-Fi. The three objectives are:

(i) To develop a smart monitoring application for passenger counting on UTAR buses.

(ii) To deploy a NerveNet-based mesh network testbed with LoRa and Wi-Fi.

(iii) To integrate the smart application into the wireless network test bed.

(iv) To evaluate the experimental performance of the smart application and wireless network system.

1.5 Scope and Limitation of the Study

This study will focus solely on UTAR's bus system, with the wireless network being constructed to cover the UTAR bus route. Hence, some wireless network modifications must be made if the smart application is deployed elsewhere.

There are a few limitations of this study. Firstly, the lighting conditions on the bus might be poor. Poor lighting will lead to inaccurate results because the AI algorithm fails to detect the passenger.

Besides, the limited coverage range of LoRa is another limitation of this study. The ideal coverage range of LoRa for urban areas is only up to 5 km (Anon., 2023).

In addition, the inference speed of the AI algorithm in this project is another significant element. This project will require real-time data, and the smart application will run on an edge device, Raspberry Pi 4. If the inference speed is slow on the edge device, processing the passenger counting application takes a long time and cannot provide real-time data.

Furthermore, the power source of the edge device is another limitation of this study. It is important to look for a long-lasting power source to power up the edge device, which will be located on the bus. The suggested power source is solar power. The edge device can be powered on using a portable solar panel, which will also be located on the bus.

1.6 Contribution of the Study

The study aims to contribute to the university, students, and staff. Firstly, this project will enhance the overall student experience using the university bus services. Using this smart application, students can avoid waiting for an overcrowded bus.

Next, this study will provide the university with an efficient transportation management system. The university management can arrange the bus trip based on the data collected from the smart application.

In addition, this study will provide insights related to the development of LoRa and Wi-Fi technologies in the smart wireless campus.

1.7 Outline of the Report

This report contains five chapters. Chapter 1 will provide this project's overview, including the general introduction, aims and objectives, problem statements, and scope and limitations. Chapter 2 will provide the literature reviews of the previously done works related to this project's scope. After that, Chapter 3 will explain this project's methodology and work plan, including the hardware and software used. Chapter 4 will present the results and the analysis of the obtained results. Lastly, Chapter 5 concludes this project and provides recommendations for future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will provide a literature review of the previously done works related to this project. The areas include object detection, object tracking, person detection method, LoRa, Wi-Fi, MQTT, NerveNet, and edge AI.

2.2 Object Detection

Vision-based object detection can be divided into traditional machine vision and complex deep learning methods (Song et al., n.d.).

2.2.1 Introduction To Traditional Machine Vision Methods

The working principles of traditional machine vision techniques are extracting features from images and using these features to recognize or classify the objects in the image (O' Mahony et al., n.d.). The features extracted include edges, corners, and other distinctive points in the image. After extracting the features, they will be matched to a database of known features to identify the objects in the image.

Manual feature engineering is usually required in traditional machine vision methods, as shown in Figure 2.1. The process can be time-consuming, and the algorithm may not generalize well to new images because the algorithm is not class specific.

However, the traditional method is very general and can be used for simple applications where specific class knowledge is not required (O' Mahony et al., n.d.). In contrast, traditional methods may be more suitable for easier, more domain-specific tasks that do not require as much data.

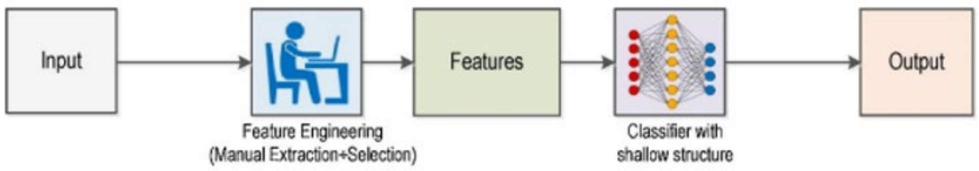


Figure 2.1: Traditional Machine Vision Methods (O' Mahony et al., n.d.).

2.2.2 Introduction to Complex Deep Learning Methods

Rupali (2022) explains that deep learning is a subset of machine learning (ML), and ML is a subset of artificial intelligence. Figure 2.2 visually depicts the relationship between these concepts: artificial intelligence, machine learning, and deep learning.

Machine learning can observe the input data, which consists of past experiences, and produce a model to solve the problems or predict any required value (Baduge et al., 2022). The significant difference between machine learning and deep learning is that deep learning has multiple hidden layers, but machine learning only consists of a single layer.

According to Wang et al. (2018), complex deep learning methods utilize the end-to-end learning structure, as shown in Figure 2.3, rather than the manual feature engineering used in the traditional machine vision method. The end-to-end learning structure in deep learning methods will automatically work out the most descriptive and salient features concerning each class.

Furthermore, deep learning neural networks are trained rather than programmed, which offers greater flexibility and accuracy in applications such as object detection (O' Mahony et al., n.d.). Deep learning demonstrates flexibility by allowing the model to be re-trained using custom datasets. Moreover, complex deep learning methods require less human intervention than traditional methods.

However, one drawback of the deep learning method is that it necessitates using substantial data for training. If only limited data are available for training in deep learning, the algorithm might not perform well in terms of accuracy.

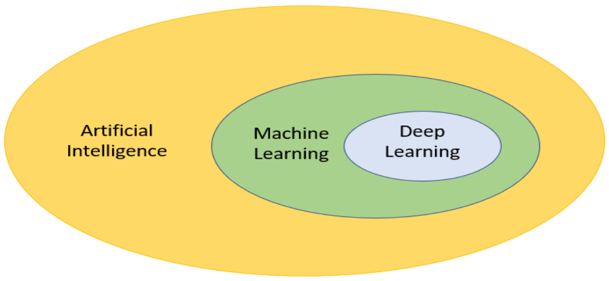


Figure 2.2: Artificial Intelligence, Machine Learning, and Deep Learning.

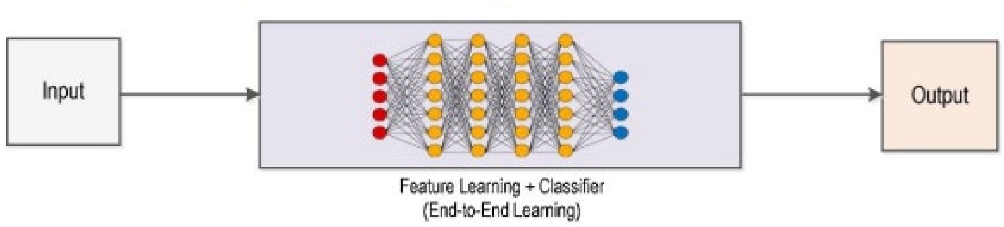


Figure 2.3: Complex Deep Learning Methods (Wang et al., 2018).

Jiang et al. (2020) categorized deep learning object detection methods in region proposal-based two-stage and regression-based one-stage methods. Popular two-stage methods include Region-based Convolutional Neural Networks (R-CNN) and Fast R-CNN. In contrast, one-stage methods include You Only Look Once (YOLO) and Single Shot Detector (SSD).

Zhang et al. (2020) mentioned that the two-stage methods perform better. However, these approaches require much higher computing power than the one-stage method. Hence, the two-stage methods are unsuitable for real-time applications due to the required high computing complexity.

Moreover, Saponara et al. (2022) mentioned that R-CNN is unsuitable for real-time application because large amounts of computational time are needed to classify the regions for each image. They also mentioned that YOLO only uses a single network evaluation while R-CNN and Fast R-CNN use thousands of regions for every image.

Next, in the research done by Sambolek & Ivasic-Kos (2021), the YOLO algorithm obtained the best results. In the research, Cascade R-CNN and Faster R-CNN detectors were also used. The results show that Cascade R-CNN and Faster R-CNN significantly perform worse than the YOLO algorithm.

Furthermore, the research done by Chen et al. (2020) shows that YOLO algorithms have obtained higher accuracy and faster detection speed compared to Faster-RCNN, which belongs to the two-stage methods.

Besides, in the research done by Arcos-García et al. (Arcos-García, Álvarez-García and Soria-Morillo, 2018), although the results of YOLO and SSD are not as good as Faster R-CNN, both YOLO and SSD still can achieve competitive accuracy results. They also discovered that YOLO and SSD are much faster than Faster R-CNN. Besides, the YOLO and SSD consumed less memory than Faster R-CNN. Hence, the YOLO and SSD, which are the one-stage method, are an optimal choice for any deployment in embedded devices or edge devices.

In addition to comparing YOLO and SSD, Masmoudi et al. (2019) found out that YOLO is very fast compared to SSD, but SSD has a better result than YOLO in their research. They mentioned that YOLO is suitable for applications requiring real-time processing, while SSD is suitable for applications that detect small objects due to its high accuracy.

Additionally, the research done by Kim et al. (2019) shows that YOLO and SSD are faster than Faster R-CNN, but the accuracy of R-CNN is higher. However, upon further analysis in the research, it was found that YOLO produces reasonably accurate results within a shorter time, making it an appropriate model for detecting people in embedded or edge devices.

2.3 Complex Deep Learning Methods

Based on the literature review of the previous part, the regression-based one-stage method is more suitable than the region proposal-based two-stage method for real-time applications due to faster inference speed. Besides, the YOLO algorithm is better than SSD in the real-time application and person detection, as mentioned by Masmoudi et al. (2019) and Kim et al. (2019).

The deep learning algorithms applied in this project are YOLOv4, YOLOv4-tiny, YOLOv5, and YOLOv7. YOLOv4 and YOLOv4-tiny will be used to detect the passenger on the bus. Besides, YOLOv5 and YOLOv7 will be used to recognize the gender of the passengers as an additional package in this project.

2.3.1 You Only Look Once (YOLO)

You Only Look Once (YOLO) was introduced by Joseph Redmon in 2016 (Chamidu, 2020). Back in 2016, more recent approaches used the region proposal-based two-stage method. In these two-stage methods, such as R-CNN, the potential bounding boxes in an image will be generated first, and a classifier will be run on these proposed boxes. After that, a post-processing process refines the bounding boxes of the detected objects using bounding box regression and eliminates duplicate detections (Girshick et al., 2014). These methods are slow, and optimising the algorithm is difficult because each component must be trained separately (Redmon et al., 2015). In YOLO, object detection was reframed as a regression-based one-stage method.

According to Redmon et al. (2015), YOLO has several benefits in object detection compared to traditional methods. Firstly, the speed of YOLO is breakneck. The pipeline of YOLO is simple since YOLO frame the detection as a regression problem. With the simple pipeline, YOLO can process real-time streaming video with less than 25 ms of latency. Secondly, YOLO has been found to have fewer than half the number of background errors compared to Fast R-CNN because it reasons globally about the input image when making predictions. Thirdly, YOLO has a high generalizability level, making it less prone to breaking down when applied to new domains or unforeseen inputs compared to other detection methods such as R-CNN.

When an input image is fed into YOLO, it partitions it into an S x S grid. Each grid cell is then tasked to detect any objects for which the centre falls within the cell. Subsequently, the bounding boxes and corresponding confidence scores for each box are predicted by each grid cell (Parico and Ahamed, 2021). A higher confidence score for a box indicates that the model is more confident that the box contains an object and that the object contained in the box corresponds to the model's prediction. Conversely, the confidence scores for a grid cell should be zero if it contains nothing. In cases where the grid cell detects an object, the confidence score is calculated as the intersection over union (IoU) between the predicted box and the ground truth.

For each bounding box, YOLO predicts five values: *x, y, w, h*, and confidence. The (*x,y*) is a coordination that indicates the bounding box's centre relative to the grid cell's bounds. The *w* and *h* values represent the width and height of the bounding box, predicted relative to the input image. Additionally, regardless of the number of bounding boxes, only one set of conditional class probabilities is predicted for each grid cell(Redmon et al., 2015). The process of YOLO is shown in Figure 2.4.

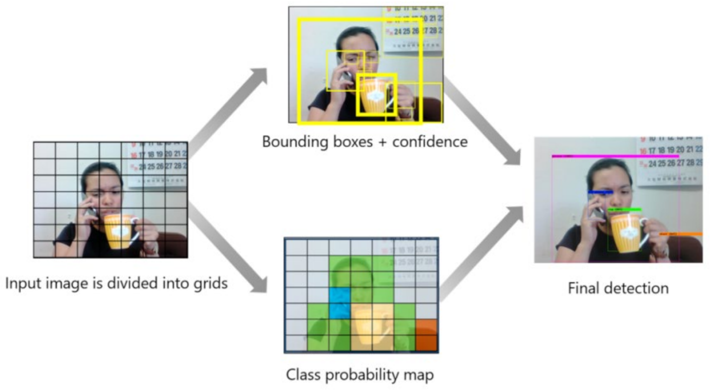

Bounding boxes + confidence
Input image is divided into grids
Final detection
Class probability map

Figure 2.4: Process of YOLO (Parico and Ahamed, 2021).

According to Redmon et al. (2015), the network of YOLO was influenced by GoogLeNet and consisted of 24 convolutional layers and two fully connected layers. However, instead of using the inception modules present in GoogLeNet, YOLO employed 1 x 1 convolutional layers followed by 3 x 3 convolutional layers (Diwan, Anirudh and Tembhurne, 2023). The 1 x 1 reduction layers reduce the feature space from the preceding layers. As a result, the network's final output is a tensor of predictions with a size of 7 x 7 x 30. The complete network of YOLO is shown in Figure 2.5.

There are some limitations of YOLO, as mentioned by Redmon et al. (2015). One limitation is that YOLO struggles to detect small objects that appear in a group, like birds. YOLO’s solid spatial constraints restrict its ability to predict many nearby objects. Next, YOLO may have difficulty detecting objects in novel aspect ratios since it relies on learned data to predict bounding boxes.

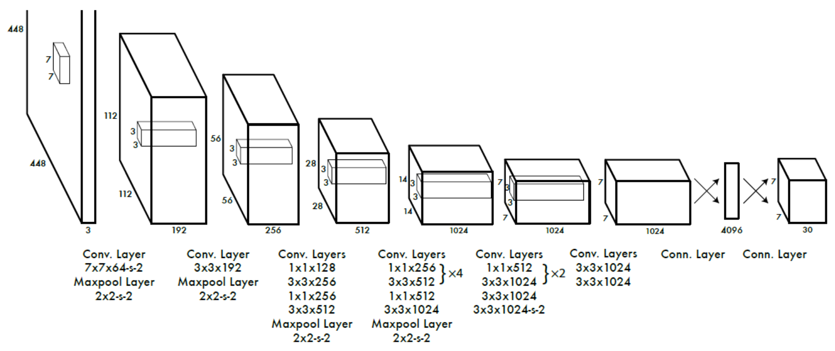


Figure 2.5: Full Network of YOLO (Redmon et al., 2015).

Redmon et al. (2015) compared YOLO with other detection systems in their research. Compared to R-CNN, YOLO only proposes 98 bounding boxes per image, while R-CNN proposes about 2000 bounding boxes. Besides, although R-CNN's speed is up in other fast detection systems such as Fast R-CNN and Faster R-CNN by sharing the computation power, both models still cannot apply in real-time applications.

In addition to the YOLO, a fast version of YOLO was designed. The convolutional layers of Fast YOLO reduced from 24 to 9 compared to YOLO. Furthermore, the number of filters used in the convolutional layers of Fast YOLO is fewer than YOLO. According to Redmon et al. (2015), Fast YOLO is the fastest object detection model on the PASCAL dataset. The mean average precision (mAP) of Fast YOLO is two times higher than previous work on real-time detection.

In short, YOLO is a unified model for object detection. YOLO can be trained directly on entire images. Besides, YOLO can generalize to new domains making it a robust object detection model. In addition, the fastest object detection model at that time is Fast YOLO, making real-time applications workable.

2.3.2 YOLOv4

YOLOv4 was developed by Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao in April 2020. YOLOv4 is the fourth version of the YOLO series. New methods were introduced, and some complex and powerful techniques were showcased in YOLOv4. According to Diwan et al. (2023), YOLOv4 performs better in speed and accuracy than all the previous versions of YOLO at that time. In addition, Jiang et al. (2022) mentioned that the changes in YOLOv4 compared to the previous versions are YOLOv4 more focused on comparing the data and have a substantial improvement.

An object detector can be divided into several building blocks: backbone, neck, and head. Figure 2.6 shows the building blocks of an object detector and the difference between one-stage and two-stage detectors. Backbone acts as the feature extractor of the deep learning architecture. Some examples of backbone include VGG16, ResNet-50, and CSPDarknet53. Neck

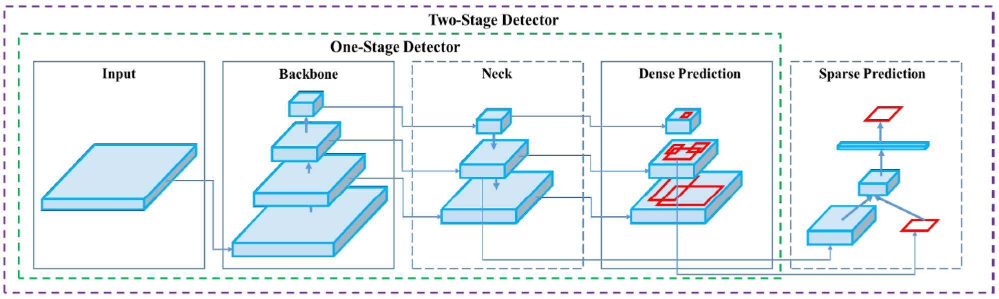


Figure 2.6: Building Blocks of One-Stage Detector and Two-Stage Detector (Bochkovskiy, Wang and Liao, 2020).

acts as a feature aggregator and collects feature maps from different stages of the backbone. Head refers to the object detector, which discovers the region where the object might exist (Vishal, 2021). According to Bochkovskiy et al. (2020), YOLOv4's architecture includes a CSPDarknet53 backbone, a neck composed of an SPP additional module and PANet, and a head based on YOLOv3.

Besides, many choices of the bag of freebies (BoF) and bag of specials (BoS) are available in YOLOv4. According to Bochkovskiy et al. (2020), BoF refers to the methods where the training strategy is changed or the training cost is increased to achieve better accuracy. Furthermore, utilizing BoF to improve a model's accuracy does not result in an increase in inference cost. BoS refers to the post-processing methods and plugin modules that will slightly increase the inference cost and significantly improve the accuracy (Vishal, 2021).

According to Bochkovskiy et al. (2020), one of the BoF for the backbone used in YOLOv4 is Mosaic, and for the detector is Cross mini-Batch Normalization (CmBN). Furthermore, the BoS used in the backbone includes Cross stage partial connections (CSP), and the detector includes Spatial Pyramid Pooling-block (SPP-block) and PAN path-aggregation block.

Mosaic is a new data augmentation method Bochkovskiy et al. (2020) introduced. Four training images were mixed in Mosaic, as shown in Figure 2.7. Mixing four images allows the detection of objects outside the normal context. Moreover, the activation statistics are computed based on four distinct images at each layer, leading to a reduced dependency on a large mini-batch size.

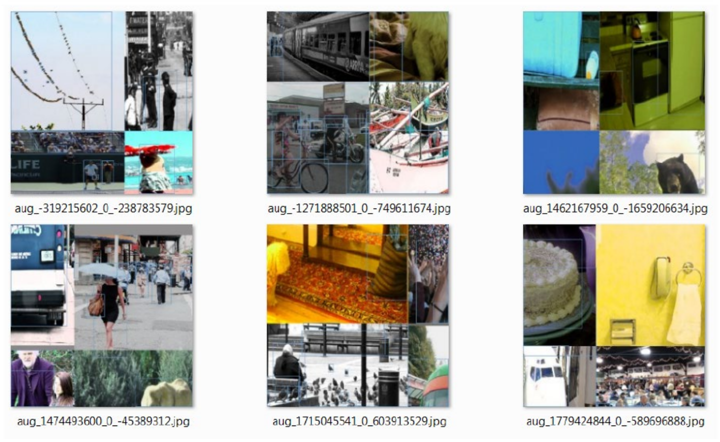

aug_-319215602_0_-238783579.jpg
aug_-1271888501_0_-749611674.jpg
aug_1462167959_0_-1659206634.jpg
aug_1474493600_0_-45389312.jpg
aug_1715045541_0_603913529.jpg
aug_1779424844_0_-589696888.jpg

Figure 2.7: Mosaic in YOLOv4 (Bochkovskiy, Wang and Liao, 2020).

Next, CmBN is a modified version of Cross Batch Normalization (CBN). The differences between CBN and CmBN are shown in Figure 2.8. In CBN, it normalizes the data with four batches in training. CBN utilize the mean and standard deviation of the past four batches to normalize the current batch (Vishal, 2021). CmBNollows the idea of CBN by collecting the statistics only between the mini-batches within a single batch (Bochkovskiy, Wang and Liao, 2020).

Besides, CSP used in YOLOv4 can reduce the computation cost and achieve a more prosperous gradient combination (Diwan, Anirudh and Tembhurne, 2023). According to C. Y. Wang et al. (2020), CSP can achieve this by dividing the feature map of the base layer into two separate parts, which can then be combined via a proposed cross-stage hierarchy. The differences between DenseNet and CSPNet are shown in Figure 2.9.

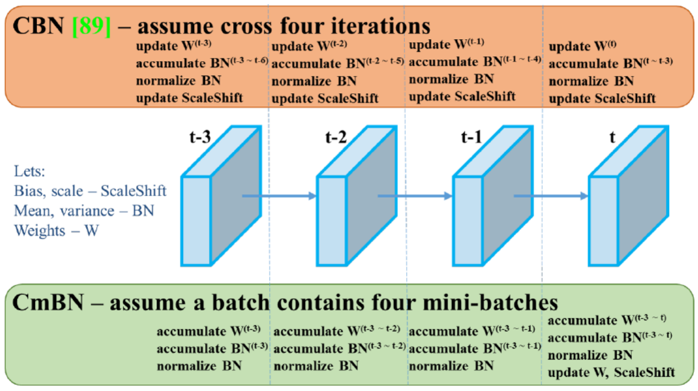

CBN [89] – assume cross four iterations
update $W^{(t-3)}$ accumulate $BN^{(t-3 \sim t-6)}$ normalize BN update ScaleShift
update $W^{(t-2)}$ accumulate $BN^{(t-2 \sim t-5)}$ normalize BN update ScaleShift
update $W^{(t-1)}$ accumulate $BN^{(t-1 \sim t-4)}$ normalize BN update ScaleShift
update $W^{(t)}$ accumulate $BN^{(t \sim t-3)}$ normalize BN update ScaleShift
t-3
t-2
t-1
t
Lets:
Bias, scale – ScaleShift
Mean, variance – BN
Weights – W
CmBN – assume a batch contains four mini-batches
accumulate $W^{(t-3)}$ accumulate $BN^{(t-3)}$ normalize BN
accumulate $W^{(t-3 \sim t-2)}$ accumulate $BN^{(t-3 \sim t-2)}$ normalize BN
accumulate $W^{(t-3 \sim t-1)}$ accumulate $BN^{(t-3 \sim t-1)}$ normalize BN
accumulate $W^{(t-3 \sim t)}$ accumulate $BN^{(t-3 \sim t)}$ normalize BN update W, ScaleShift

Figure 2.8:Differences between CBN and CmBN (Bochkovskiy, Wang and Liao, 2020).

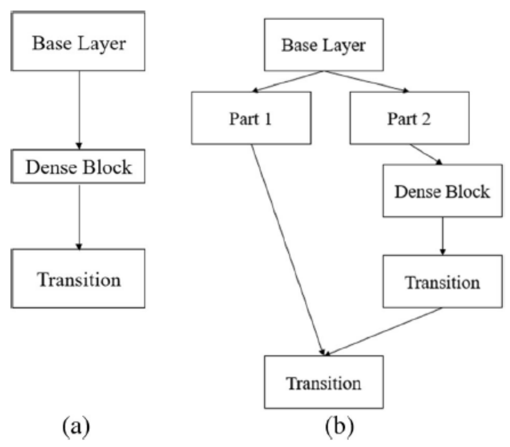


Figure 2.9: Differences between (a) DenseNet and (b) CSPNet (Diwan, Anirudh and Tembhurne, 2023).

Furthermore, an SPP block was used after the CSPDarknet53 in YOLOv4 to separate some important features from the backbone. In SPP, the feature map of the input image will be extracted using convolutional layers. After that, a feature set will be generated. Repeating the feature set generation process by *n* times will produce the different feature maps in height and width dimensions, creating a pyramid. In YOLOv4, SPP was applied in each part and combined to generate an output feature map (Vishal, 2021). Figure 2.10 shows the process of SPP.

In addition, a modified PAN was created in YOLOv4. The addition connection of PAN was replaced by concatenation (Bochkovskiy, Wang and Liao, 2020). Figure 2.11 shows the differences between the original PAN and the modified PAN.

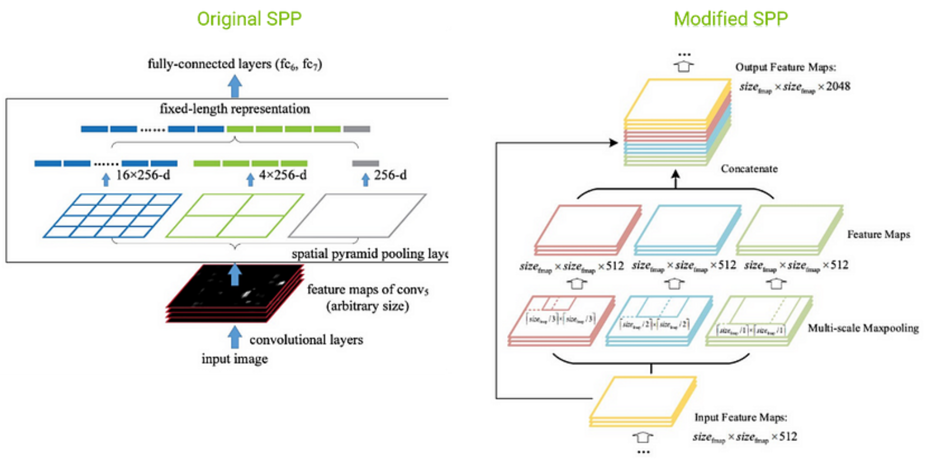

Original SPP
fully-connected layers (fc6, fc7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling laye
feature maps of conv5 (arbitrary size)
convolutional layers
input image
Modified SPP
Output Feature Maps: size_fmap × size_fmap × 2048
Concatenate
Feature Maps
size_fmap × size_fmap × 512
Multi-scale Maxpooling
Input Feature Maps: size_fmap × size_fmap × 512

Figure 2.10: Flows of SPP (Vishal, 2021).

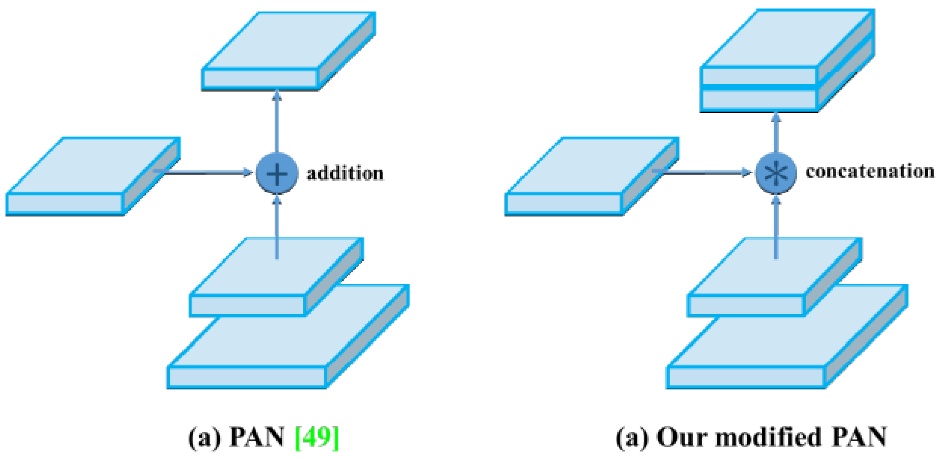


Figure 2.11: Differences between Original PAN and Modified PAN.

In the research done by Chethan Kumar et al. (2020) and Degadwala et al. (2021), the results show that YOLOV4 can achieve high accuracy in object detection. According to Chethan Kumar et al. (2020), YOLOv4 achieved 99 % accuracy for the video dataset. Besides, Degadwala et al. (2021) mentioned that the accuracy of YOLOv4 is 98.9 % for medical face mask detection in their project.

2.3.3 YOLOv4-tiny

YOLOv4-tiny is proposed based on the architecture of YOLOv4 and is a compact version of YOLOv4 (Kulshreshtha et al., 2021). According to Jiang et al. (2020), the network structure of YOLOv4-tiny is more straightforward and more suitable for developing on embedded devices when compared to YOLOv4. In addition, Parico and Ahamed (2021) mentioned that the inference speed (FPS) for YOLOv4-tiny is higher than YOLOv4 when having a lower inference memory usage, as shown in Figure 2.12.

According to Wang et al. (2021), the backbone of YOLOv4-tiny is greatly simplified. Figure 2.13 shows the network structure of YOLOv4-tiny. In addition, Jiang et al. (2020) mentioned that YOLOv4-tiny uses CSPDarknet53-tiny as its backbone network instead of the CSPDarknet53 used in YOLOv4. Moreover, YOLOv4-tiny abandons the Spatial Pyramid Pooling (SPP) and uses the feature pyramid network to increase the object detection speed (Anand, Das and Sarkar, 2021).

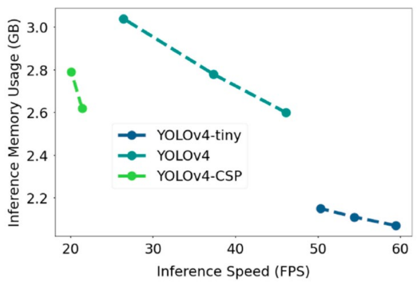


Figure 2.12: Comparison of YOLO v4 models in Inference Speed and Inference Memory Usage (Parico and Ahamed, 2021).

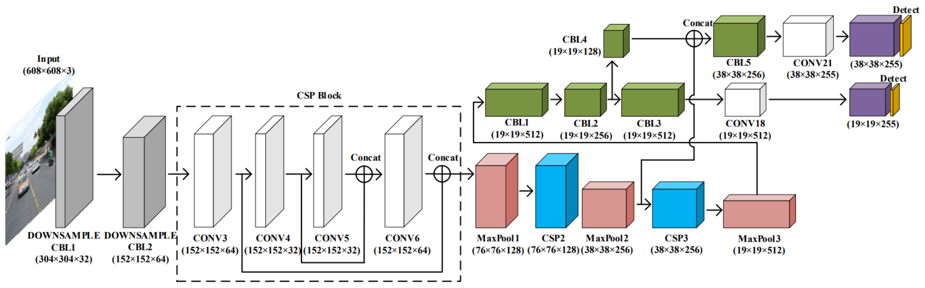


Figure 2.13: The Network Structure of YOLOv4-tiny (Wang et al., 2021).

Furthermore, YOLOv4 and YOLOv4-tiny differ in that YOLOv4 was trained using the weights of 137 pre-trained convolutional layers, whereas YOLOv4-tiny was only trained using the weights of 29 pre-trained convolutional layers (Kulshreshtha et al., 2021).

From the research done by Kulshreshtha et al. (2021), YOLOv4 obtained the final mAP of 97.10 %, and YOLOv4-tiny obtained the last mAP of 95.20 %. However, the detection time of YOLOv4-tiny is six times faster than YOLOv4. Although the mAP of YOLOv4-tiny is slightly lower than YOLOv4 for 1.9 %, YOLOv4-tiny was chosen in this research because the application requires on-the-spot decisions.

Next, the research done by Ying et al. (2021) shows that the YOLOv4-tiny is four times faster than YOLOv4, and the accuracy of YOLOv4-tiny was not compromised much compared to YOLOv4. In addition, the research done by Anand, Das and Sarkar (2021) shows that YOLOv4-tiny achieved 4.6 times higher in frame-per-second (FPS) and the mAP of YOLOv4-tiny only 0.75 % less than YOLOv4. They mentioned that YOLOv4-tiny is preferred for real-world surveillance cameras because it performs better in FPS.

In short, the architecture of YOLOv4-tiny is different from YOLOv4. However, the prediction process of YOLOv4 and YOLOv4-tiny are the same. Overall, the YOLOv4-tiny has a much faster speed than YOLOv4 but a slightly lower mAP than YOLOv4. Hence, YOLOv4-tiny is more suitable for this project because this project requires a real-time application.

2.3.4 YOLOv5

YOLOv5 was released in June 2020, just two months after the release of YOLOv4. Zhu et al. (2021) stated that in YOLOv5, the backbone network is CSPDarknet53 with an SPP layer, the neck network is PANet, and the head network is YOLO detection. Do (2021) mentioned that the architecture of YOLOv4 and YOLOv5 are very similar, and the starting date of the research of boto YOLOv4 and YOLOv5 are close.

According to Jiang et al. (2022), people are dissatisfied with YOLOv5 because it contains fewer innovations when compared to YOLOv4. However, YOLOv5 still has some performance improvements and some significant advantages. The advantages of YOLOv5 include the user-friendly PyTorch framework, which enables a more straightforward training process and makes it easier to develop into production than the Darknet framework used in YOLOv4 (Jiang et al., 2022). Besides, YOLOv5 integrates a large amount of computer vision technologies.

In addition, according to Do (2021), the YOLOv5 possessed advantages in engineering. The installation and integration of YOLOv5 are more straightforward than in previous versions because YOLOv5 is written in Python instead of C language in the previous versions. The growth potential of YOLOv5 is higher than YOLOv4 because YOLOv5 uses the PyTorch framework, and the PyTorch community is larger than the Darknet community.

According to Jacob (2020), the notable difference in YOLOv5 is the auto-learning bounding box anchors. The previous version of YOLO utilized the k-mean clustering algorithm with different k values to select the five most appropriate anchor boxes, which helped to enhance model performance while reducing training time. However, a limitation of this approach is that the anchor boxes cannot easily adapt to unique custom datasets (Do, 2021). To address this, the k-means clustering algorithm is run on the unique custom dataset to obtain the best-fit anchor boxes. After that, these parameters will be manually configurated into YOLO. In YOLOv5, the anchor box selection process was integrated. Hence, the network can automatically learn the most suitable anchor boxes for unique custom datasets during training (Jacob, 2020).

Comparing the performance between YOLOv4 and YOLOv5 is hard to achieve accurately because the two models use different programming languages and frameworks. However, YOLOv5 was proven to perform better than YOLOv4 under certain circumstances (Do, 2021).

2.3.5 YOLOv7

YOLOv7 was introduced by Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao in July 2022 (Gaudenz, 2023). According to Wang, Bochkovskiy and Liao (2022), YOLOv7 has better performance in speed and accuracy than other all-known object detectors. YOLOv7 has the highest accuracy of 56.8 % average precision (AP) and can achieve up to 160 FPS on GPU V100.

There are two significant changes in the YOLOv7's architecture compared to the previous version: extended efficient layer aggregation networks (E-ELAN) and model scaling for concatenation-based models. Besides, another significant change in YOLOv7 is the trainable bag of freebies (BoF). The trainable BoF include planned reparameterized convolution, coarse for auxiliary, and fine for lead loss (Gaudenz, 2023).

According to Wang, Bochkovskiy, and Liao (2022), the primary considerations in designing efficient architectures are the number of parameters, amount of computation, and computational density. To overcome this, the authors propose Extended-ELAN based on ELAN. Compared to ELAN, the gradient transmission path of the original architecture in E-ELAN will not change. The cardinality of the added features was increased using the group convolution in E-ELAN. After that, the features of different groups will be combined in the shuffle and merged in a cardinality manner. Hence, the use of parameters and computations can be improved in E-ELAN by enhancing the features learned by different feature maps. Figure 2.14 shows the difference between ELAN and E-ELAN.

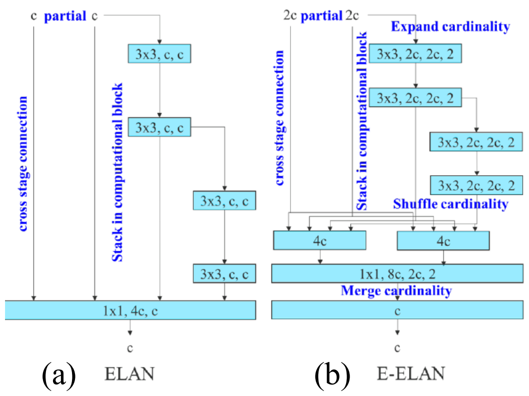


Figure 2.14: (a) ELAN and (b) E-ELAN (Wang, Bochkovskiy and Liao, 2022).

Besides, Wang, Bochkovskiy, and Liao (2022) proposed the corresponding compound model scaling for the concatenation-based model in YOLOv7. Performing a depth scaling on both the concatenation-based model and the scaled-up concatenation-based model results in an increase in the output width of a computational block. Hence, the input width of the subsequent transmission layer will increase in these two models. To address this issue, Wang, Bochkovskiy, and Liao (2022) introduced a compound scaling approach for increasing the depth and width of a concatenation-based model. In this approach, only the depth of the computational block needs to be scaled, and then the remaining transmission layers are scaled correspondingly in width. Figure 2.15 shows the differences between the three models mentioned above.

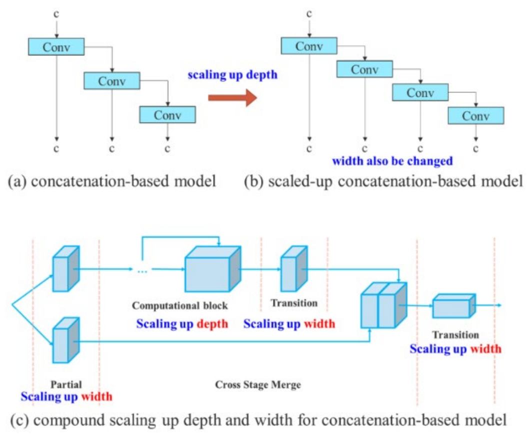


Figure 2.15: The differences between (a) the concatenation-based model, (b) the scaled-up concatenation-based model, and (c) the proposed model (Gaudenz, 2023).

According to Wang, Bochkovskiy, and Liao (2022), although RepConv obtained better performance in VGG architectures, there is a significant accuracy loss due to the direct application in ResNet or DenseNet. Hence, to address this issue, the planned re-parameterized convolution architecture in YOLOv7 used RepConv without the identity connection (Gaudenz, 2023). With this implementation, the identity connection will not exist because the re-parameterized convolution replaces the convolutional layer.

In addition, the architecture of YOLOv7 contains more than one head. The lead head is responsible for the final output, while the auxiliary head assists in training the middle layers (Gaudenz, 2023). In YOLOv7, Wang, Bochkovskiy, and Liao (2022) introduced the "label assign", a mechanism that takes into account both the network prediction results and the ground truth, and then assigns a soft label accordingly. Lead-guided assigner and coarse-to-fine lead-guided assigner were proposed by Wang, Bochkovskiy, and Liao (Wang, Bochkovskiy and Liao, 2022).

In short, the authors of YOLOv7 claim that they have developed the YOLOv7, which receives state-of-the-art results. The significant changes of YOLOv7 include E-ELAN, model scaling for concatenation-based models, planned reparameterized convolution, coarse for auxiliary, and fine for lead loss.

2.4 Object Tracking Algorithm

In the field of computer vision, object tracking is a significant research direction. Object tracking technology has made incredible progress in the past two to three decades (Zhang, Chen and Wei, 2020). In general, an object tracking algorithm will automatically identify the objects in the videos and assume the detected objects as a set of high-accuracy trajectories (Nico, 2023).

The object tracking algorithm can be divided into classical and deep learning algorithms. Examples of classical object tracking algorithms are Meanshift, Kalman Filter, and Particle Filter. Examples of deep learning object tracking algorithms are Simple Online and Realtime Tracking (SORT) and Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT).

According to Perera et al. (2021), deep learning object-tracking algorithms are gaining more and more attention from researchers due to their promising performances. In addition, the performances of the classical tracking algorithm are poor, especially under ambient lighting changes and occlusions.

2.4.1 Deep Learning Tracking Algorithm

Recent advancements in deep learning tracking algorithms have led to breakthroughs in object-tracking technology. The deep learning tracking algorithm used in this project is DeepSORT.

2.4.1.1 DeepSORT

According to Wojke, Bewley, and Paulus (2017), DeepSORT is an object-tracking algorithm that extends to SORT. SORT was introduced by Bewley et al. (2016), and it is designed for online tracking, where only the detections from the previous and current frames are provided to the tracker. DeepSORT was integrated with a deep appearance-based metric derived from Convolutional Neural Network (CNN) (Parico and Ahamed, 2021). Figure 2.16 shows the architecture of DeepSORT.

The single conventional hypothesis tracking methods using frame-by-frame data association and recursive Kalman filtering were adopted in DeepSORT (Wojke, Bewley and Paulus, 2017). A space with an eight-dimensional state consisting of a bounding box centre position with the height *h*, aspect ratio γ, and the respective velocities in the coordination of the images were defined in Deep SORT. Besides, a Kalman filter was used in DeepSORT.

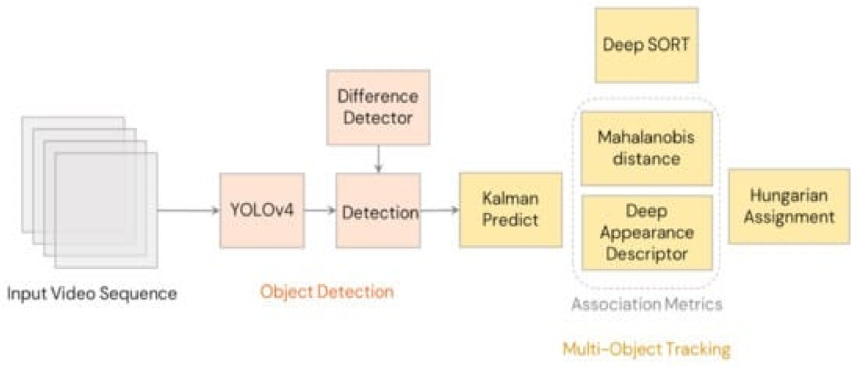


Figure 2.16: Architecture of DeepSORT (Parico and Ahamed, 2021).

In DeepSORT, the algorithm was integrated with appearance information and improved DeepSORT’s performance. In addition, the objects can be tracked with more extended periods of occlusion, reducing the number of identities switched due to the integration (Nico, 2023). According to Wojke, Bewley, and Paulus (2017), DeepSORT has demonstrated remarkable robustness and speed and is considered among the top-performing tracking algorithms.

2.5 Person Detection Methods

There are two primary categories of person detection methods: those that operate on an overhead view and those that operate on a normal view. Person counting has many applications, such as crowd analysis and person counting. However, it is challenging for the researchers to detect the person accurately due to a wide range of variations such as posture, size, and orientation of a person (Ahmad, Ahmed and Adnan, 2019).

According to Ahmed and Adnan (2018), researchers have done a great job on the person detection method, but the algorithm might fail in some situations, as shown in Figure 2.17. For example, in Figure 2.17, the machinery might block the person, and the camera might not capture the person being covered. In this situation, the person detection algorithm might fail.

However, in the overhead view, as shown in Figure 2.18, the person can be seen easily from the camera, and the failure of the person detection algorithm can be avoided (Ahmed and Adnan, 2018). In addition, according to Ahmad, Ahmed, and Adnan (2019), the wide angle of the overhead view can provide more coverage and save cameras’ installation costs.

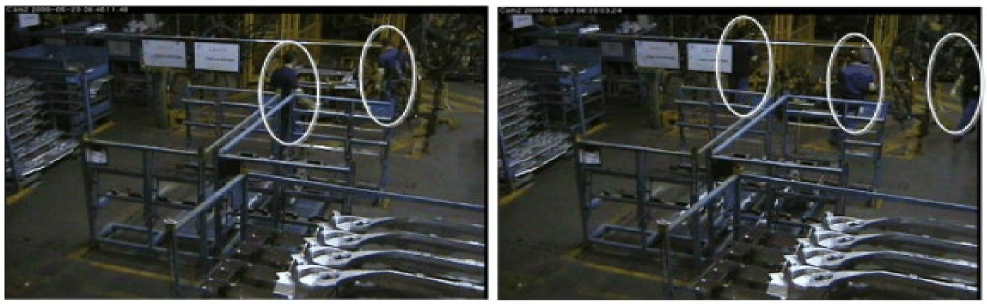


Figure 2.17: Normal View (Ahmed and Adnan, 2018).

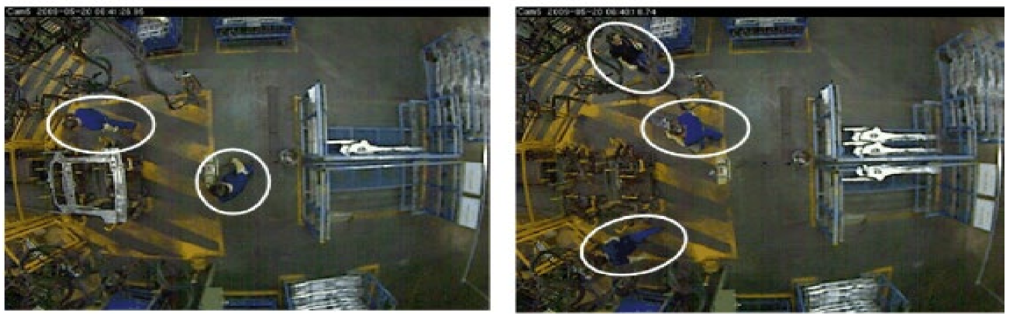


Figure 2.18: Overhead View (Ahmed and Adnan, 2018).

2.6 Wi-Fi

One of the wireless networking technologies that fall under the Institute of Electrical and Electronics Engineers (IEEE) standard is Wi-Fi, which has gained popularity over time. The IEEE 802.11 standard lists the protocol that enables communications between Wi-Fi-enabled wireless devices, such as access points and routers (Cisco, 2023).

Wi-Fi was first introduced by CNR Corporation/AT&T in the Netherlands in 1991, and since then, it has rapidly developed to provide wireless connectivity for computing devices such as laptops. Wi-Fi is a wireless local area network (WLAN) that enables local area networks to operate without cables and wiring, making it a popular choice for home and business networks.

In current worlds, Wi-Fi can mainly be divided into 2.4 GHz and 5 GHz. The option of choosing either 2.4 GHz or 5 GHz depends on the users' application. 2.4 GHz offers a broader coverage range than 5 GHz, and 5 GHz offers a much faster speed than 2.4 GHz. The maximum Effective Isotropic Radiated Power (EIRP) for WLAN devices under different frequency bands are listed in the guidelines provided by Malaysian Communications and Multimedia Commission (MCMC) (Malaysian Communications And Multimedia Commission, n.d.). The details of the maximum EIRP for each frequency band are listed in Appendix A. In addition, the formula of EIRP is shown below:

$$EIRP = P_{out} - C_t + G_t \tag{2.1}$$

where

P_{out} = transmitter power output (dBm)

C_t = signal loss in cable (dB)

G_t = gain of the antenna (dBi)

2.7 LoRa

LoRa is a wireless networking technique derived from Chirp Spread Spectrum (CSS) technology (The Things Network, 2023). LoRa used Chirp pulses to encode the information on radio waves. One of the advantages of LoRa is that it can transmit over a long distance compared to other wireless technologies such as Wi-Fi and Bluetooth.

The ideal application for LoRa is one that only transmits a small packet of data with relatively low bit rates. Besides, LoRa is also suitable for operating sensors and actuators that operate in low-power mode.

The Internet of Things (IoT) has snowballed in recent years, and many sensors and actuators are required in IoT applications. One of the challenges in the IoT field is the connectivity between IoT devices (BEHRTECH, 2023). Bluetooth can only be applied to short-range applications, and Wi-Fi consumes too much power for the sensors and actuators. Hence, LoRa is suitable for most IoT applications that only require low bit rates due to the long-distance coverage and low power consumption. Figure 2.19 compares LoRa and other wireless networking technologies such as Wi-Fi and cellular.

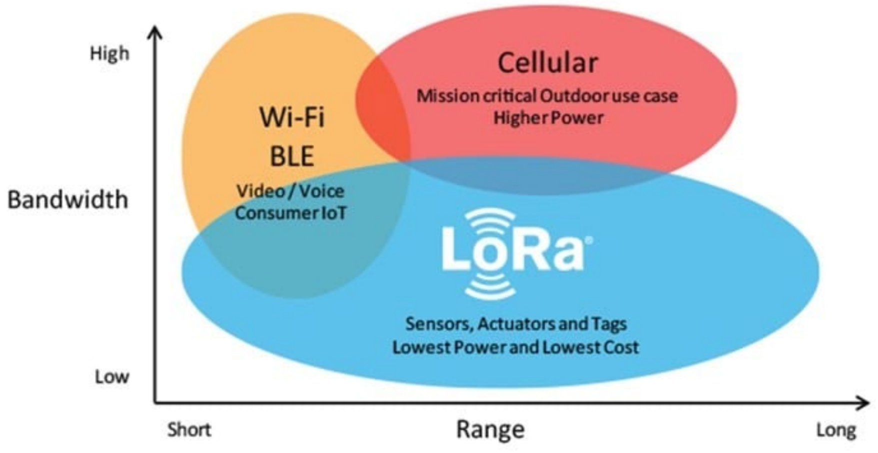

High
Bandwidth
Low
Short
Range
Long
Wi-Fi
BLE
Video / Voice
Consumer IoT
Cellular
Mission critical Outdoor use case
Higher Power
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost

Figure 2.19: LoRa vs Other Wireless Networking Technologies.

Microchip Technology (2023) states that LoRa is the standard for wireless communication for Low-Power Wide Area Networks (LPWAN). LPWAN provides low-cost, low-power, ad wide-area coverage for wireless networks. Ideally, the range of LoRa is up to 15 km in rural areas and more than 2 km in urban areas.

There are several configurations of LoRa, including Coding Rate (CR), Spreading Factor (SF), and bandwidth (Ali, Adilah and Salimi, 2019). CR is expressed in the Forward Error Correction (FEC), a process that adds some redundant bits to the original data being transmitted to increase the protection against data corruption. For example, the CR of 4/8 indicates that for every four bits of useful information, another eight bits of redundant data will be generated. Besides, SF refers to the ratio of symbol rate and chip rate. When the value of SF is higher, more chips will be used to represent a symbol. The higher the SF, the longer the transmission range of the LoRa, but the lower the data rate. Next, bandwidth is the frequency range of the chirp signal used to send the data.

2.8 MQ Telemetry Transport (MQTT)

MQTT is a messaging protocol designed for IoT with an extremely lightweight and efficient structure. Due to its small code footprint, it is particularly well-suited for connecting remote devices with limited network bandwidth and memory constraints (MQTT, 2023). MQTT employs a publish or subscribe communication pattern for machine-to-machine communication. Figure 2.20 shows the MQTT publish/subscribe architecture.

There are several advantages of MQTT, including its being lightweight and efficient. Besides, MQTT is a bi-directional communication that makes broadcasting messages to groups of things easier. In addition, MQTT can provide a reliable message delivery which is essential for IoT use cases.

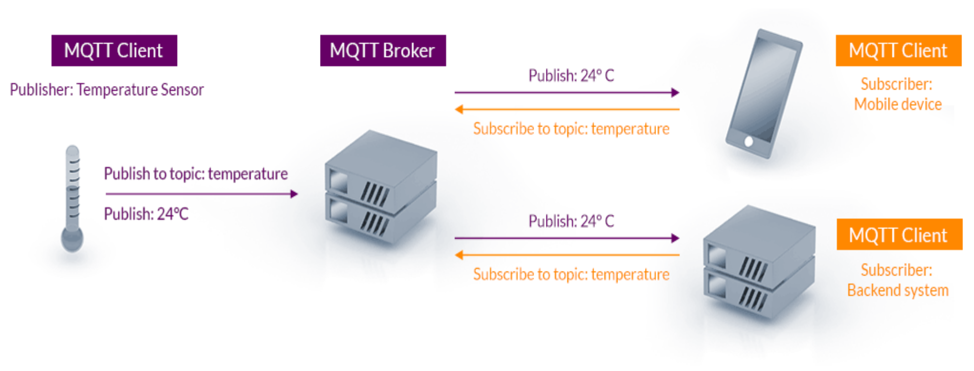

MQTT Client
Publisher: Temperature Sensor
MQTT Broker
Publish to topic: temperature
Publish: 24°C
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber: Mobile device
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber: Backend system

Figure 2.20: MQTT Publish/Subscribe Architecture.

2.9 NerveNet

According to Inoue and Owada (Inoue and Owada, 2017), NerveNet was initially designed and developed in 2008 by Japan's NICT. At that time, the purpose of establishing NerveNet was to provide several services to solve social problems in the regions and improve the quality of residents' lives using sensors and actuators (Owada, Inoue and Ohnishi, 2011). Figure 2.21 shows the configuration and service image of NerveNet.

The East Japan Great Earthquake and subsequent tsunami on March 11, 2011, motivated Inoue and Owada (2017) to upgrade the NerveNet to a disaster-resilient information sharing and communication system. According to Inoue and Owada (2017), the upgraded NerveNet can provide a robust network without relying on the current network or Internet system. In addition, according to Tham et al. (2023), the advantage of NerveNet is that an end device does not depend on another end device's availability. The service of NerveNet will not be interrupted when any node is down since the other nodes will automatically find a new pathway to transfer the message.

2.10 Edge AI

Edge AI combines both edge computing and AI. According to Liang, Shenoy, and Irwin (2020), edge AI refers to the process of running machine learning or deep learning algorithm on the edge nodes.

Edge AI also refers to deploying AI applications in the device throughout the physical world. The computation process of the AI algorithm will happen at the edge and not the computing facility as in the traditional ways.

Edge AI is a powerful solution for processing real-time data (Ramasubramanian et al., 2022). Many edge hardware accelerators, including Intel NCS 2, have recently merged to support edge AI such as computer vision (Liang, Shenoy and Irwin, 2020).

One of the main advantages of edge AI is that it is suitable for applications that require real-time processing. Edge AI will process the application locally rather than in a faraway cloud or computing facility, which will cause a delay in the application. In addition, the Internet bandwidth is much lesser because the processing power is nearer to the edge. This can reduce networking costs.

2.11 Summary

This chapter discussed the literature review done by the previous works and the innovation of the models used in this project. The literature review studied object detection methods, object tracking algorithms, person detection methods, Wi-Fi, LoRa, MQTT, NerveNet, and edge AI.

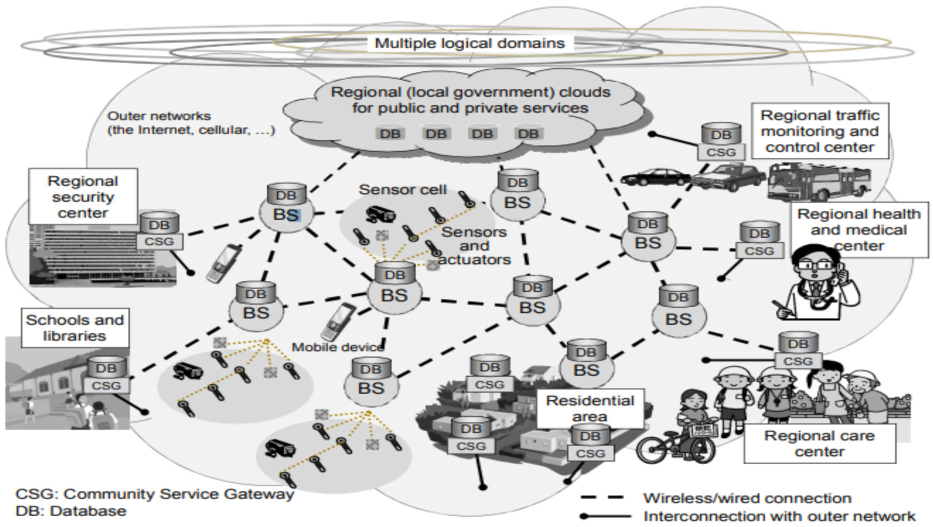

Multiple logical domains
Regional (local government) clouds for public and private services
DB
Outer networks (the Internet, cellular, ...)
Regional traffic monitoring and control center
Regional security center
Sensor cell
BS
CSG
Sensors and actuators
Regional health and medical center
Schools and libraries
Mobile device
Residential area
Regional care center
CSG: Community Service Gateway
DB: Database
Wireless/wired connection
Interconnection with outer network

Figure 2.21: Configuration and Service Image of NerveNet (Owada, Inoue and Ohnishi, 2011).

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This section will discuss the overview of the hardware and software required, the methodology and the work plan of this project. The methodology will be discussed including AI training, AI model optimization, implementation of DeepSORT, edge AI testbed, and evaluation of the results.

The hardware equipment required includes Intel NUC, Raspberry Pi 4, Sony IMX219 camera, DFRobot GPS module, and GlobalSat GPS module. The operating system used in this project is Ubuntu 18.04 LTS and Raspberry Pi OS. Next, the toolkit used in this project is OpenVINO. In addition, the setup on the UTAR bus with the Sony IMX219 camera module will be shown. Lastly, this project's network configurations using LoRa and Wi-Fi will be shown.

3.2 Overview of Hardware Required

This project's hardware implementation includes AI training, wireless networking, and smart application.

3.2.1 AI Training

The training of AI for YOLOv4, YOLOv4-tiny, YOLOv5, and YOLOv7 was done on Intel NUC with an external Graphics Processing Unit (GPU). The Intel NUC's operating system (OS) is Ubuntu 18.04 long-term support (LTS). The specifications of the NUC and external GPU used for AI training are listed in Table 3.1.

Intel NUC is a small personal computer engineered, built, and backed by Intel, as shown in Figure 3.1. Intel NUC allows users to customise their mini-PC experience and is used in many use cases. Intel NUC in this project consists of 64 GB RAM and a 2 TB solid-state drive (SSD). In addition, the Intel NUC was connected to an external NVIDIA GeForce GTX 1080 Ti GPU. The external GPU mainly supports the training, which involves thousands of images.

Table 3.1: Specifications of NUC and External GPU for AI Training.

CPU	Intel i7-10710U
CPU Cores	6
Memory	64 GB
Storage	2 TB SSD
External GPU	NVIDIA GeForce GTX 1080 Ti
Operating System	Ubuntu 18.04 LTS

Figure 3.1: Intel NUC.

The operating system used for the AI training is Ubuntu 18.04 LTS. The Ubuntu 18.04 used in this project is the long-term support (LTS) version. Ubuntu is a Linux distribution based on Debian that is available with community and professional support at no cost. Ubuntu would be one of the most popular OS used by most developers due to the freedom to customise the OS. Although the latest version of Ubuntu is Ubuntu 23.04, the latest version is not recommended to avoid software incompatibility issues in this project.

3.2.2 Smart Application

The smart application on this project is counting the number of passengers on the UTAR bus. The smart application will be run on a Raspberry Pi, assigned as R3 in this project. The specifications of the R3 are listed in Table 3.2. The smart application will run on the Raspberry Pi 4 consisting of 264 GB of Secure Digital (SD) card storage and 8 GB of memory. The OS of this Raspberry Pi is the official supported OS, which is Raspberry Pi OS.

R3 will support both LoRa and Wi-Fi. The Wi-Fi used in R3 is 2.4 GHz, and the power consumption is 21 dBm, equal to 126 mW. According to

Table 3.2: Specifications of R3 used for Smart Application.

Model	Raspberry Pi 4
Storage	264 GB SD card
Memory	8 GB

the guideline shown in Appendix A, for frequency bands between 2.4 GHz to 2.5 GHz, the maximum EIRP is 27 dBm. In this case, the configuration of Wi-Fi follows the guidelines set by MCMC. For LoRa, the bandwidth used by R3 is 125 kHZ, the CR is 4/8, and the SF is 12.

Since the computational powers of a Raspberry Pi are limited, Intel Neural Compute Stick 2 (NCS2) is plugged into the Raspberry Pi used for the smart application. Figure 3.2 shows the Intel NCS2. The NCS2 is a development kit for AI inferencing that has been designed by Intel to be plug-and-play. It is often used in conjunction with low-cost edge devices, such as the Raspberry Pi, in order to provide additional computational power. The NCS2 includes the Movidius Myriad X vision processing unit (VPU) within its hardware.

In addition, Alfa AWUS036ACH Wi-Fi USB 3.0 AC Wi-Fi Adapter Dual Band and Alfa Wi-Fi Antenna ARS-N19 2.4GHz 9dBi Dipole Antenna are plugged into the Raspberry Pi for smart application. These two pieces of hardware shown in Figure 3.3 are used for Wi-Fi wireless networking.

Furthermore, LoRa RFLink RM-92A, RFLink RM-92X USB, 920 MHz Whip Antenna ANT-92XA, and an antenna cable are added to this Raspberry Pi for LoRa wireless networking. The combination of devices for LoRa is shown in Figure 3.4.

Figure 3.2: Intel NCS2.

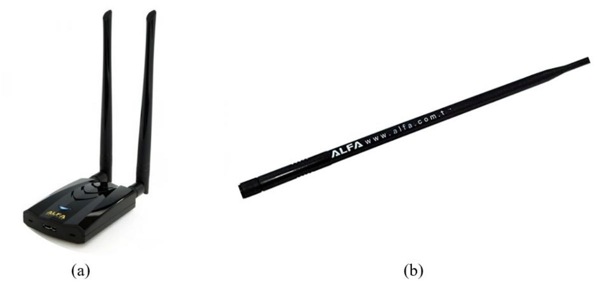


Figure 3.3: (a) Alfa AWUS AWUS036ACH Wi-Fi Adapter and (b) Alfa Wi-Fi Antenna.

Figure 3.4: Combination of the Devices for LoRa.

Moreover, the 8MP Sony IMX219 camera module for Raspberry Pi, as shown in Figure 3.5, is connected to the Raspberry Pi 4 for capturing the passengers from the overhead view.

Lastly, GlobalSat BU-353S4 G-STAR IV global positioning system (GPS) shown in Figure 3.6 is connected to R3. The (GPS) is used to read the current locations of the R3. Besides, GPS is also used for LoRa transmission. According to Tham et al. (2023), time division multiple access (TDMA) is used in NerveNet to overcome any potential signal interference of LoRa. The GPS information will be received by the LoRa node for time synchronization with other nodes so that the data will be transmitted within the pre-configured period.

Figure 3.5: 8MP Sony IMX219 Camera Module for Raspberry Pi.

Figure 3.6: GlobalSat BU-353S4 G-STAR IV GPS.

3.2.3 Wireless Networking (Wi-Fi)

In this project, two devices will be purely used for Wi-Fi wireless networking. The two devices will be Intel NUC, as shown in Figure 3.1. Both of the devices will run on Ubuntu 18.04 LTS OS. Besides, both Intel NUC will use 2.4 GHz Wi-Fi.

The first device will be assigned as N8. N8 will be connected to Alfa APA-M25 Dual Band 2.4GHz/5GHz 10dBi high gain Directional Indoor Panel Antenna as shown in Figure 3.7. The EIRP of N8 is 27 dBm which is the maximum EIRP allowed according to the guideline shown in Appendix A.

Next, another device purely used for Wi-Fi will be assigned as N10. N10 is also connected to Alfa APA-M25 Dual Band 2.4GHz/5GHz 10dBi high gain Directional Indoor Panel Antenna. Besides, the combinations of Alfa AWUS036ACH Wi-Fi USB 3.0 AC Wi-Fi Adapter Dual Band and Alfa Wi-Fi Antenna ARS-N19 2.4GHz 9dBi Dipole Antenna shown in Figure 3.3 also plugged into this Intel NUC. The EIRP of N10 is 27 dBm, equal to the maximum EIRP allowed for 2.4 GHz, as shown in Appendix A.

Figure 3.7: Alfa APA-M25 Antenna.

3.2.4 Wireless Networking (LoRa)

There will be another four Raspberry Pi used in this project, which is known as R1, R2, R4, and R5. All four Raspberry Pi are Raspberry Pi 4 models and run on the official Raspberry Pi OS. Besides, the four Raspberry Pi are connected to a combination of devices for LoRa, the same as R3. The combination of the devices for LoRa connected to these four Raspberry Pi is shown in Figure 3.4. The configurations of LoRa for the four Raspberry Pi are the same as R3, which is 125 kHz bandwidth, CR of 4/8, and SF of 12.

In addition, the GPS module is connected to these four nodes for TDMA purposes. For R1, the GPS module connected is GlobalSat BU-353S4 G-STAR IV GPS as shown in Figure 3.6. For R2, R4, and R5, the GPS module connected is DFRobot, as shown in Figure 3.8.

3.2.5 Wireless Networking (Wi-Fi + LoRa)

One Intel NUC in this project will use both Wi-Fi and LoRa wireless networking technologies. This NUC is assigned as N4 and runs on Ubuntu 18.04 LTS OS. N4 will use 2.4 GHz Wi-Fi, and the configuration of LoRa for N4 is the bandwidth of 125 kHz, CR of 4/8, and SF of 12. The maximum EIRP of N4 is 27 dBm which follows the guideline in Appendix A.

N4 is connected to Alfa APA-M25 Dual Band 2.4GHz/5GHz 10dBi high gain Directional Indoor Panel Antenna as shown in Figure 3.7 and Alfa APA-M04 Accurate 7dBi Wi-Fi Directional Antenna as shown in Figure 3.9. In addition, the N4 is connected to the DFrobot GPS module, as shown in Figure 3.8, for LoRa transmission.

Figure 3.8: DFRobot GPS Module.

Figure 3.9: Alfa APA-M04 Accurate 7dBi Wi-Fi Directional Antenna.

3.3 AI Training

This section describes the various stages involved in AI training, which include preparing the dataset, training the model, optimizing the model, and evaluating the model.

3.3.1 Dataset Preparation

In this project, two different datasets are being used for training. The first set focuses on overhead passenger counting, while the second focuses on the additional package provided to the smart application, passenger gender recognition.

3.3.1.1 Overhead Passenger Counting

The overall flowchart for this dataset preparation is shown in Figure 3.10. The dataset for overhead passenger counting is obtained from Sun et al. (2019). The dataset consists of videos of passengers entering and leaving a bus. There will be only one class of object in this dataset: passenger.

Firstly, the videos are downloaded from the source. After that, the VLC media player extracts the frames from the downloaded videos. The steps for extracting the frames are shown in Figure 3.11. After the extractions process, the output images will be uploaded to *roboflow* at *https://roboflow.com/* for annotation (Dwyer, 2022). The comparison of the image before and after an annotation is shown in Figure 3.12. Next, the images and labelled text files are downloaded from *roboflow* after annotation. Lastly, the images and the labels are divided into training and validation.

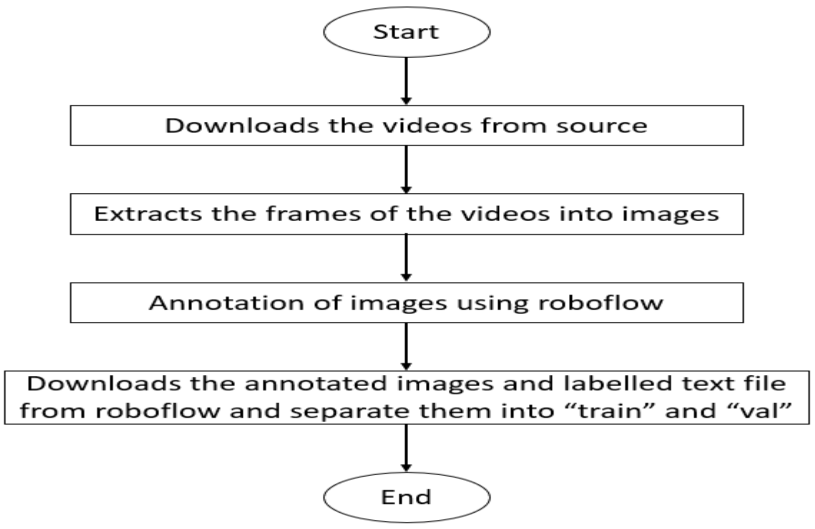

Start
Downloads the videos from source
Extracts the frames of the videos into images
Annotation of images using roboflow
Downloads the annotated images and labelled text file from roboflow and separate them into “train” and “val”
End

Figure 3.10: Flowchart for Overhead Passenger Counting Dataset Preparation.

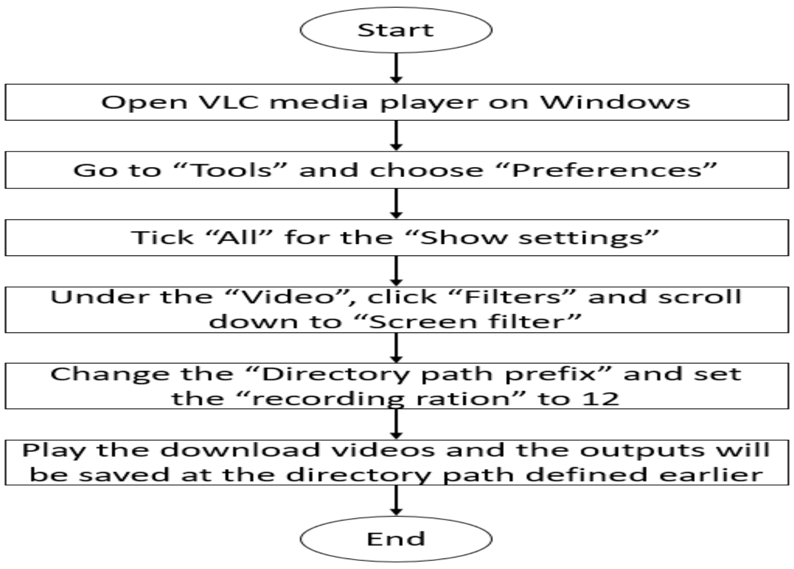

Start
Open VLC media player on Windows
Go to “Tools” and choose “Preferences”
Tick “All” for the “Show settings”
Under the “Video”, click “Filters” and scroll down to “Screen filter”
Change the “Directory path prefix” and set the “recording ration” to 12
Play the download videos and the outputs will be saved at the directory path defined earlier
End

Figure 3.11: Flowchart for Extracting the Frames from Downloaded Videos.

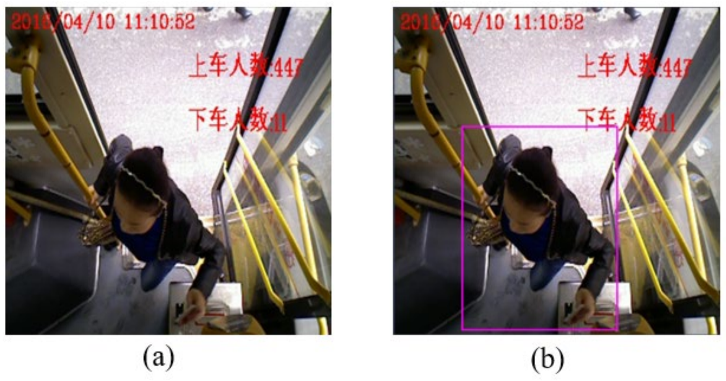


Figure 3.12: Comparison of (a) Before and (b) After Annotation.

3.3.1.2 Passenger Gender Recognition

The overall flowchart for this dataset preparation is shown in Figure 3.13. The dataset for passenger gender recognition is obtained from Kucev (2021). The dataset consists of images with four types of masks worn by females and males.

In this project, there will be eight classes of objects in this dataset, which are female_type1, male_type1, female_type2, male_type2, female_type3, male_type3, female_type4, and male_type4. The difference between the eight classes is shown in Figure 3.14. This project will use this dataset to recognize the gender of the passengers and whether the passengers are wearing the mask properly.

Firstly, the images are downloaded from the source. Then, a Python file, “preprocess.py”, is run to remove the non-image files and categorize the images into eight classes, as mentioned above. After that, the images are uploaded to *roboflow* at *https://roboflow.com/* for annotation (Dwyer, 2022). Next, the annotated images and the labelled text file are downloaded. Lastly, the annotated images and labels are divided into training and validation.

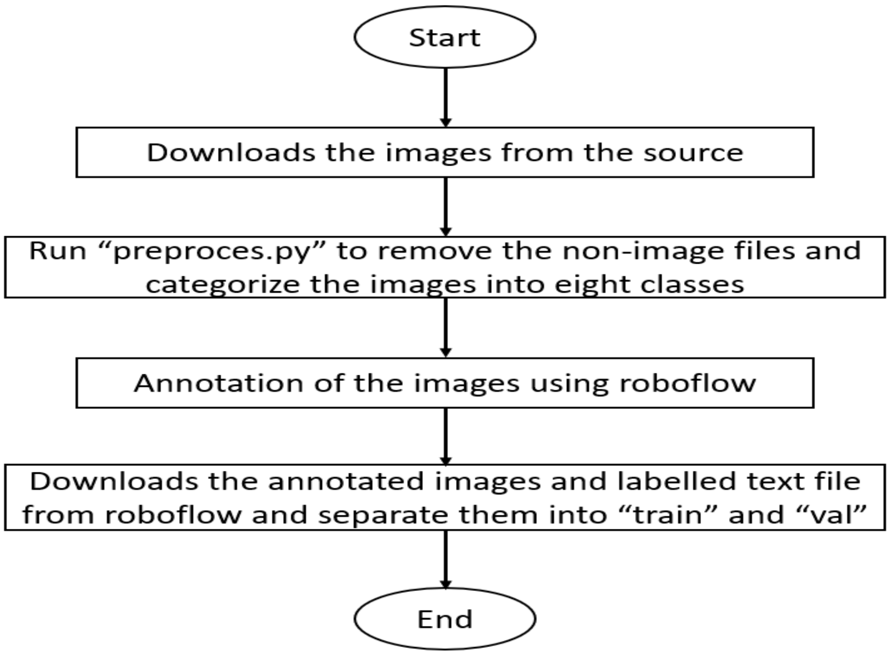

Start
Downloads the images from the source
Run “preproces.py” to remove the non-image files and categorize the images into eight classes
Annotation of the images using roboflow
Downloads the annotated images and labelled text file from roboflow and separate them into “train” and “val”
End

Figure 3.13: Flowchart for Passenger Gender Recognition Dataset Preparation.

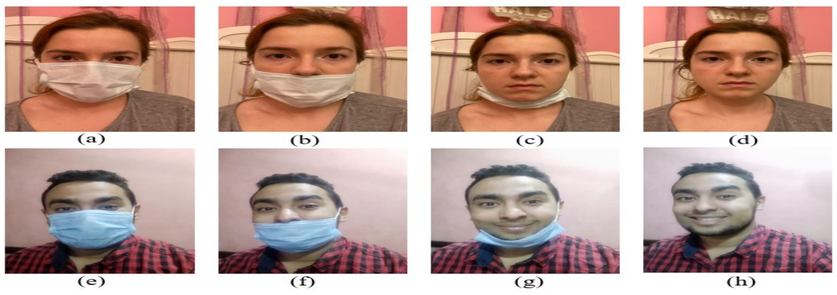


Figure 3.14: (a) female_type1, (b) female_type2, (c) female_type3, (d) female_type4, (e) male_type1, (f) male_type2, (g) male_type3, and (h) male_type4.

3.3.2 Model Training

This section will describe the model training of YOLOv4, YOLOv4-tiny, YOLOv5, and YOLOv7.

3.3.2.1 YOLOv4 and YOLOv4-tiny

This section will discuss the model training for YOLOv4 and YOLOv4-tiny since the processes are similar. YOLOV4 and YOLOv4-tiny are used to detect and count the number of passengers on the UTAR bus. Figure 3.15 presents an overview of the training process for both YOLOv4 and YOLOv4-tiny models.

First, the darknet repository is cloned from GitHub into the Intel NUC used for AI training. Then, the first four lines of "*Makefile*" in the darknet repository are changed, as shown in Figure 3.16. The changes are to speed up the training process and allow camera detection. Next, the "*make*" command is run to create the darknet executable file. After that, the darknet executable file is copied into two newly created folders: "*yolov4*" and "*yolov4-tiny*".

The next step is customising the *cfg* file, which is the same for YOLOv4 and YOLOv4-tiny. The "*classes*" inside the *cfg* file are the number of classes in our project. For YOLOv4 and YOLOv4-tiny, there is only one class. Hence, the "classes" inside the *cfg* file is changed to "1". Besides, the "*filters*" also changed in both *cfg* files to "18" following the formula below:

$$filters = (classes + 5) \times 3 \tag{3.1}$$

where

classes = number of classes

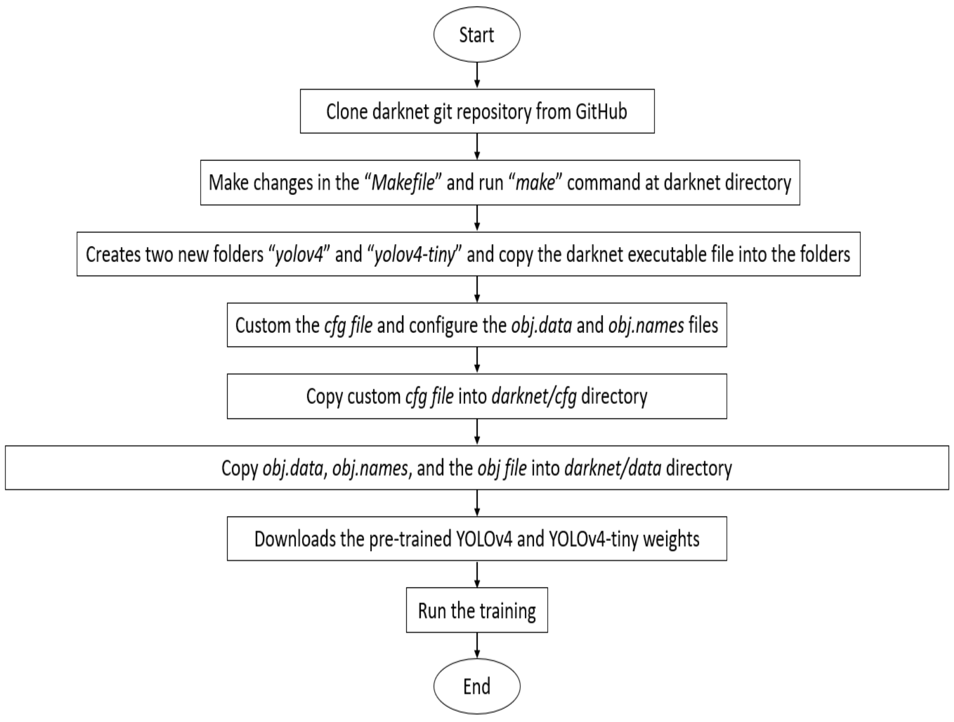
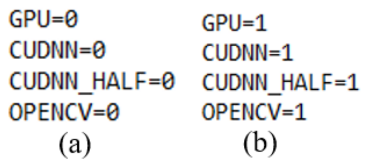

Start
Clone darknet git repository from GitHub
Make changes in the "*Makefile*" and run "*make*" command at darknet directory
Creates two new folders "*yolov4*" and "*yolov4-tiny*" and copy the darknet executable file into the folders
Custom the *cfg file* and configure the *obj.data* and *obj.names* files
Copy custom *cfg file* into *darknet/cfg* directory
Copy *obj.data*, *obj.names*, and the *obj file* into *darknet/data* directory
Downloads the pre-trained YOLOv4 and YOLOv4-tiny weights
Run the training
End

Figure 3.15: Overall Training Process for YOLOv4 and YOLOv4-tiny.

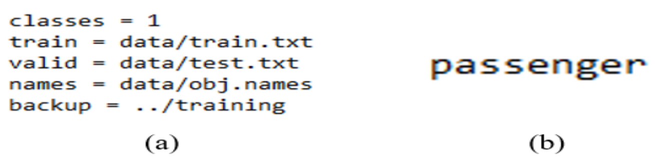


```
GPU=0           GPU=1
CUDNN=0         CUDNN=1
CUDNN_HALF=0    CUDNN_HALF=1
OPENCV=0        OPENCV=1
```

Figure 3.16: The Changes Made in "*Makefile*" (a) before (b) after.

After that, the *obj.data* and *obj.names* files are configured accordingly. The configuration is shown in Figure 3.17. The *obj.data* file consists of the number of classes and the training data directory while the obj.names consist of the classes' names.

After that, the datasets obtained from the previous preparation will be copied to a new file, "*obj*". The images are divided into a ratio of 9:1, where 90 % of the images are used for training, and 10 % are used for validation. Inside the "*obj*" file, two new files, which are "*train*" and "*val*" are created. The images for training and validation are copied into these two files. The number of training images is 450, and for validation is 50.



(a)

```
classes = 1
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = ../training
```

(b)

```
passenger
```

Figure 3.17: Configuration of (a) *obj.data* and (b) *obj.names*.

In this project, transfer learning is used instead of training from scratch. Hence, the pre-trained weights of YOLOv4 and YOLOv4-tiny are downloaded from https://github.com/AlexeyAB/darknet/releases/. The pre-trained weights of YOLOv4 are yolov4conv.137, and for YOLOv4-tiny are yolov4-tiny.weights. The difference between these two weights is that the weights of YOLOv4 used have been trained up to 137 convolutional layers, and the weights of YOLOv4-tiny have only trained up to 29 convolutional layers. After obtaining the pre-trained weights, the training for YOLOv4 and YOLOv4-tiny can be run. In this project, the training will run for 6,000 iterations, and the batch size is 64 for both YOLOv4 and YOLOv4-tiny.

3.3.2.2 YOLOv5

In this project, YOLOv5 will be trained to recognize the gender of the passengers and detect whether they are wearing the mask properly as an additional package for the smart application. The overall training process of YOLOv5 is shown in Figure 3.18.

Firstly, the *yolov5* repository is cloned from GitHub at https://github.com/ultralytics/yolov5. After that, the dataset prepared earlier for gender recognition is separated into the ratio of 9:1, where 90 % of the images are used for training, and the remaining 10 % are used for validation. The number of training images is 3600, and for validation is 400.

Then, a new folder, "*dataset*", is created on the Desktop of the Intel NUC. After that, a folder "*y5*" is created in the "*dataset*" folder. Next, two new folders are created into the "*y5*", which are "*images*" and "*labels*". Then, two new folders, which are "*train*" and "*val*", are created in both the "*images*" and "*labels*" folders, respectively. The structure of the new folders created here is shown in Figure 3.19. After that, the images and labels are copied into the *"train"* and *"val"* folders, respectively.

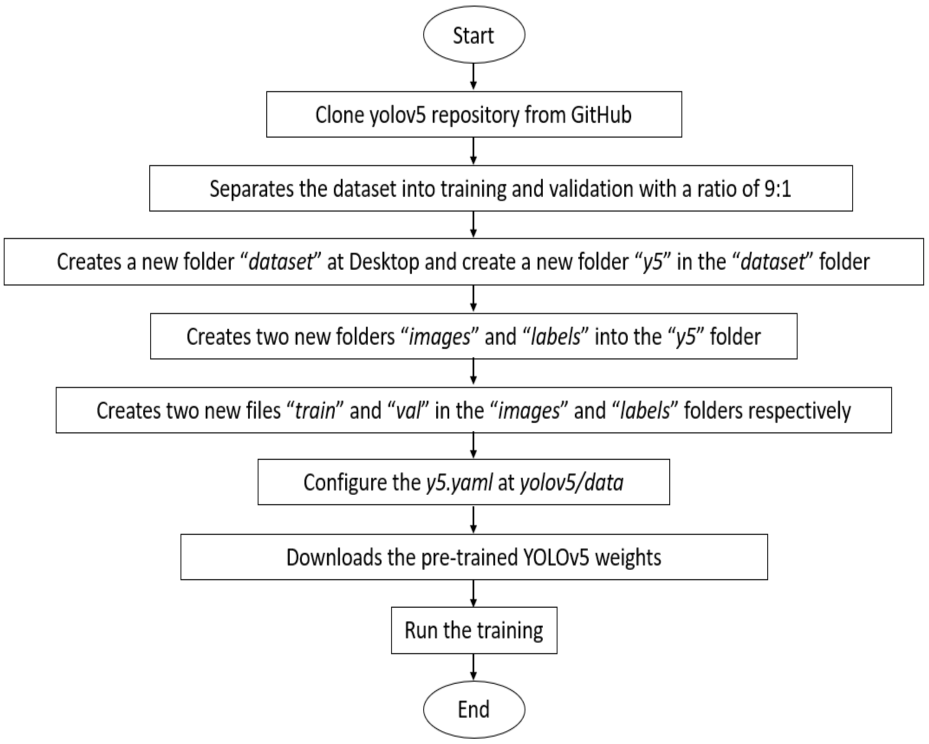
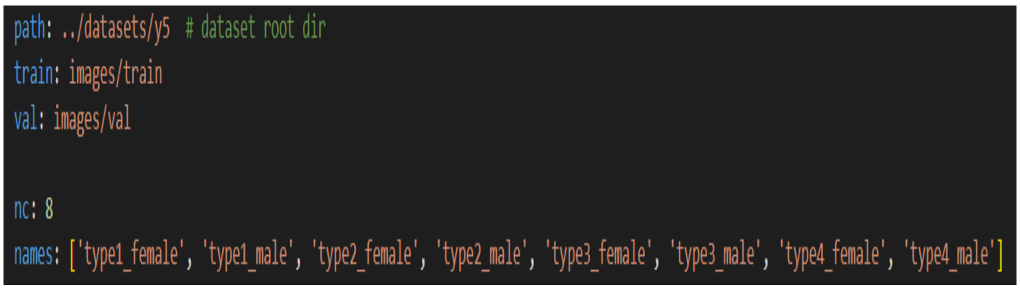

Start
Clone yolov5 repository from GitHub
Separates the dataset into training and validation with a ratio of 9:1
Creates a new folder "*dataset*" at Desktop and create a new folder "*y5*" in the "*dataset*" folder
Creates two new folders "*images*" and "*labels*" into the "*y5*" folder
Creates two new files "*train*" and "*val*" in the "*images*" and "*labels*" folders respectively
Configure the *y5.yaml* at *yolov5/data*
Downloads the pre-trained YOLOv5 weights
Run the training
End

Figure 3.18: Overall Training Process of YOLOv5.

```
Desktop
   |_____
        dataset
          |_____y5
                 |_____images
                 |       |_____ train (folder includes all training images)
                 |       |_____ val (folder includes all validation images)
                 |_____labels
                         |_____ train (folder includes all training labels)
                         |_____ val (folder includes all validation labels)
```

Figure 3.19: The Structure of the New Folders Created for YOLOv5.

Then, the *y5.yaml* is configured. The *y5.yaml* contains the directory of the images and labels. It also contains the number of classes and the names of classes. Figure 3.20 shows the *y5.yaml* after configuration.



```
path: ../datasets/y5  # dataset root dir
train: images/train
val: images/val

nc: 8
names: ['type1_female', 'type1_male', 'type2_female', 'type2_male', 'type3_female', 'type3_male', 'type4_female', 'type4_male']
```

Figure 3.20: *y5.yaml* File After Configuration.

Finally, the pre-trained YOLOv5 weights are downloaded from https://github.com/ultralytics/yolov5/releases/. Similar to YOLOv4 and YOLOv4-tiny, transfer learning is also applied to YOLOv5. The chosen weight is *yolov5m.pt*. The training can be started once the pre-trained weights are downloaded. For the training of YOLOv5, the epochs are 300, and the batch size is 16.

3.3.2.3 YOLOv7

In this project, YOLOv7 will also be trained to recognize the gender of the passengers and detect whether they are wearing the mask properly as an additional package for the smart application. The overall training process of YOLOv7 is shown in Figure 3.21.

Firstly, the *yolov7* repository is cloned from GitHub at https://github.com/WongKinYiu/yolov7. After that, the dataset prepared earlier for gender recognition is separated into the ratio of 9:1, where 90 % of the images are used for training, and the remaining 10 % are used for validation. The number of training images is 3600, and for validation is 400.

Then, two new folders, which are "*train*" and "*val*", are created inside the "*data*" folder. Furthermore, two new folders, which are "*images*" and "*labels*", are created in the *train*" and "*val*" folders, respectively. The structure of the new folders created is shown in Figure 3.22.

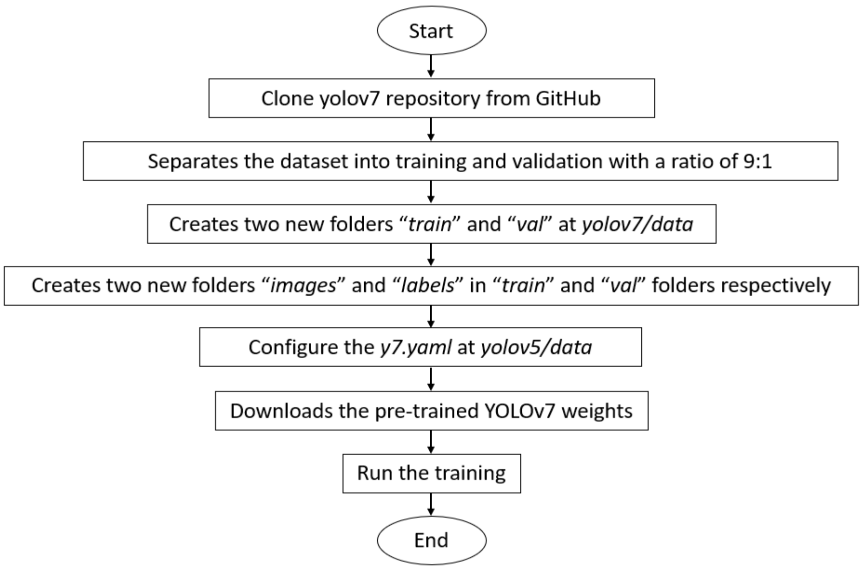

Start
Clone yolov7 repository from GitHub
Separates the dataset into training and validation with a ratio of 9:1
Creates two new folders "*train*" and "*val*" at *yolov7/data*
Creates two new folders "*images*" and "*labels*" in "*train*" and "*val*" folders respectively
Configure the *y7.yaml* at *yolov5/data*
Downloads the pre-trained YOLOv7 weights
Run the training
End

Figure 3.21: Overall Training Process for YOLOv7.

```
├── data
    └── train
      └── images (folder includes all training images)
      └── labels (folder includes all training labels)
    └── val
      └── images (folder includes all validation images)
      └── labels (folder includes all validation labels)
```

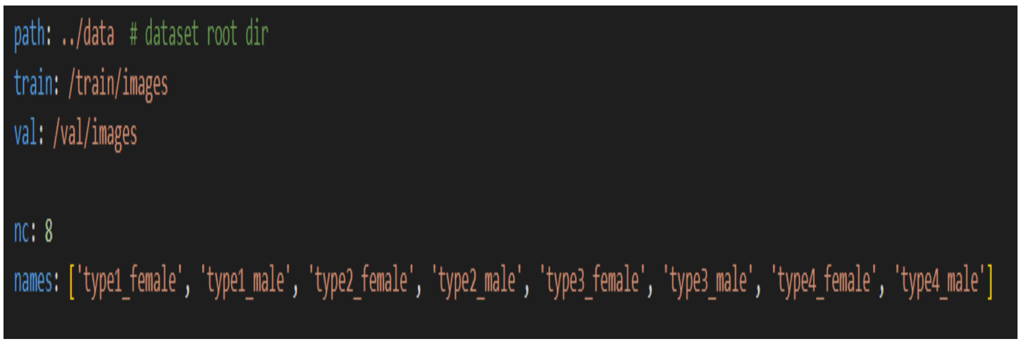
Figure 3.22: The Structure of the New Folders Created for YOLOv7.

After that, the *y7.yaml* file is configured. The *y7.yaml* file includes the path of the training and validation images, the number of classes, and the name of the classes. Figure 3.23 shows the y7.yaml file after the configuration.

Lastly, the pre-trained YOLOv7 weights are downloaded from https://github.com/WongKinYiu/yolov7/releases/. Similar to YOLOv4, YOLOv4-tiny, and YOLOv5, transfer learning is also applied to YOLOv7. The chosen weight is *yolov7_training.pt*. The training can be started once the pre-trained weights are downloaded. For the training of YOLOv7, the hyperparameters used are the same as YOLOv5; the epochs are 300, and the batch size is 16. The same hyperparameters for YOLOv5 and YOLOv7 are being applied to compare the performances between the two models.

3.4 AI Model Optimization

The AI model optimization only will be done on YOLOv4 and YOLOv4-tiny because the passengers' detection and counting will be deployed at the edge, where gender recognition is just an additional package of the smart application. The model optimization will speed up the model's inference speed, especially when the model is deployed on an edge device such as Raspberry Pi.



```
path: ../data  # dataset root dir
train: /train/images
val: /val/images

nc: 8
names: ['type1_female', 'type1_male', 'type2_female', 'type2_male', 'type3_female', 'type3_male', 'type4_female', 'type4_male']
```

Figure 3.23: *y7.yaml* File After Configuration.

The AI model optimization will be done using Open Visual Inference and Neural Network Optimization (OpenVINO). OpenVINO is an open-source toolkit providing faster inference for deploying deep learning models developed by Intel. The developers can develop cost-effective and robust computer vision applications by using OpenVINO.

Figure 3.24 shows the overall optimization process using OpenVINO. After the training of YOLOv4 and YOLOv4-tiny is completed, the output weights will be saved. After that, the output weights will be converted into intermediate representation (IR) format by running the model optimizer of OpenVINO. The IR format consists of two files which are *.xml* and *.bin*. After that, the IR format can be fed into the inference engine to check the model's compatibility.

3.5 Implementation of DeepSORT

In this project, the DeepSORT is used for tracking the passenger. After the YOLOv4 and YOLOv4-tiny detect the passenger, the algorithm will pass it to DeepSORT for tracking. The DeepSORT will be deployed together with YOLOv4 and YOLOv4-tiny in IR format after the conversion by OpenVINO.

The deployment of YOLOv4 and YOLOv4-tiny in OpenVINO format with DeepSORT is referred from Mateusz (2021). In addition, the codes are modified for passenger counting. The overall process of the passenger counting application after modification is shown in Figure 3.25.

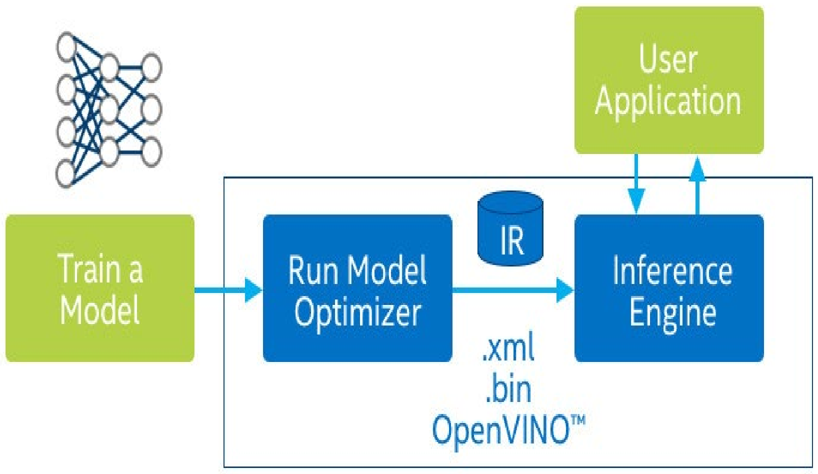


Figure 3.24: AI Model Optimization Process using OpenVINO.

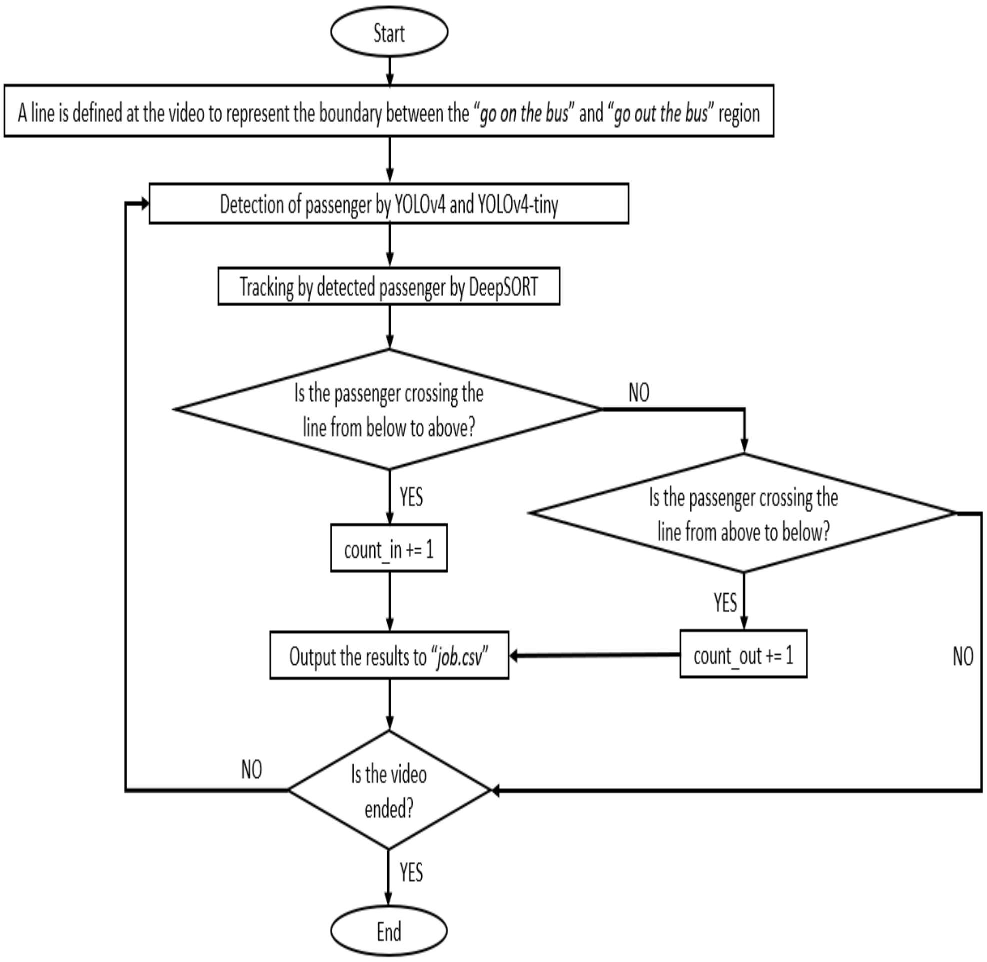

Start
A line is defined at the video to represent the boundary between the "go on the bus" and "go out the bus" region
Detection of passenger by YOLOv4 and YOLOv4-tiny
Tracking by detected passenger by DeepSORT
Is the passenger crossing the line from below to above?
NO
YES
count_in += 1
Is the passenger crossing the line from above to below?
YES
count_out += 1
NO
Output the results to "job.csv"
NO
Is the video ended?
YES
End

Figure 3.25: Overall Process for Passenger Counting Application.

When DeepSORT tracks any passenger, it will check the direction of the passenger crossing the line. If the passenger crosses the line from below to above indicates the passenger is going into the bus. If the passenger crosses the line from above to below indicates the passenger is going out from the bus. After that, the number of passengers going in and out of the bus will be output to "*job.csv*".

3.6 Edge AI Testbed

Figure 3.26 shows the edge AI testbed of this project. R3 will run the smart application on a UTAR bus. Besides, R1, R2, R3, and R4 will act as LoRa nodes and be put in Bandar Sungai Long's surrounding areas. In addition, the N10 and N8 will only have Wi-Fi connectivity. Moreover, as mentioned earlier, N4 will have both LoRa and Wi-Fi connectivity.

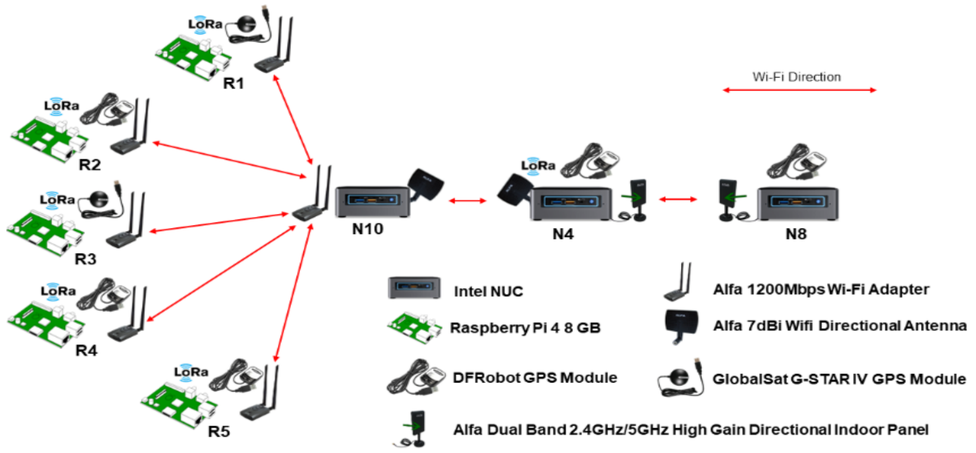

LoRa
R1
R2
R3
R4
R5
N10
N4
N8
Wi-Fi Direction
Intel NUC
Raspberry Pi 4 8 GB
DFRobot GPS Module
Alfa 1200Mbps Wi-Fi Adapter
Alfa 7dBi Wifi Directional Antenna
GlobalSat G-STAR IV GPS Module
Alfa Dual Band 2.4GHz/5GHz High Gain Directional Indoor Panel

Figure 3.26: Edge AI Testbed.

3.7 Edge AI LoRa Deployment

Figure 3.27 shows the deployment of the edge AI LoRa. As mentioned earlier, R3 is the Raspberry Pi that will run the smart application on a UTAR bus and act as a publisher. R1, R2, R4, R5, and N4 will act as LoRa nodes and subscribers in this configuration.

There are five LoRa subscribers and one publisher in this scenario. When the publisher publishes data, the nearest subscriber node will receive the data and relay the data to further subscriber nodes. As mentioned, this scenario will divide one minute into six time slots since LoRa works in TDMA. In this scenario, the LoRa nodes can only transmit the data in their designated time slots, which last for 10 seconds. If the sent packet does not receive any acknowledgement from the other nodes, it will only resend the data for one time. After that, the data will be discarded.

The LoRa sending in this deployment is designed at five data packets sent per bus trip. The bus locations are the five points shown in Figure 3.27. The mechanism to trigger the LoRa sending after each bus stop is shown in Figure 3.28.

When the bus starts to move, the smart application will be triggered. The number of passengers going in and out of the bus will be updated to the "*job.csv*" if the smart application detects any of them. At the same time, the GPS of R3 will read the current location and the speed of the bus. If the distance between the current locations and the designated locations listed in

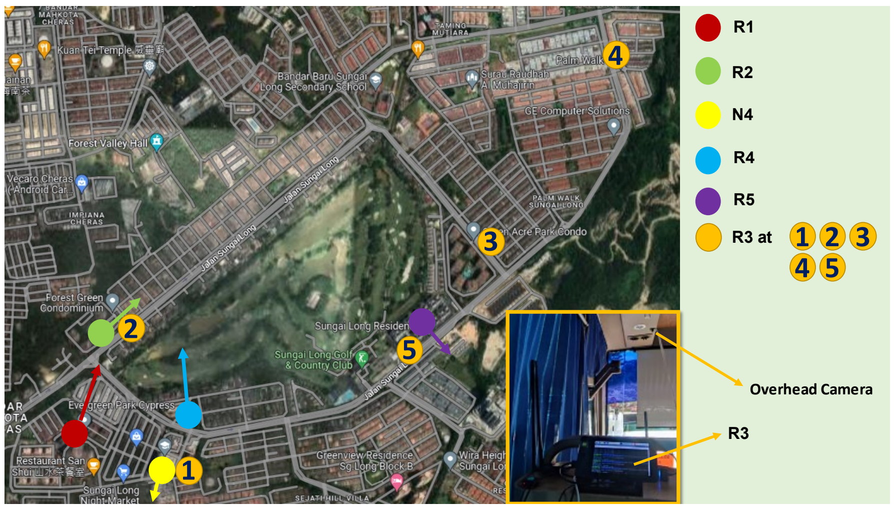


Figure 3.27: Edge AI LoRa Deployment.

"*gps_corrdinate.csv*" is less than 50 m, the application will check the speed of the bus. If the bus speed is continuously less than 10 km/h for 20 s, the application will check if the speed increases from below 10 km/h to above 30 km/h. If yes, the LoRa sending mechanism will be triggered, and the LoRa will send the last row of data in "job.csv" to N4, the only LoRa node located at UTAR. The smart application will stop if it detects that the bus has reached all the designated locations listed in "*job.csv*".

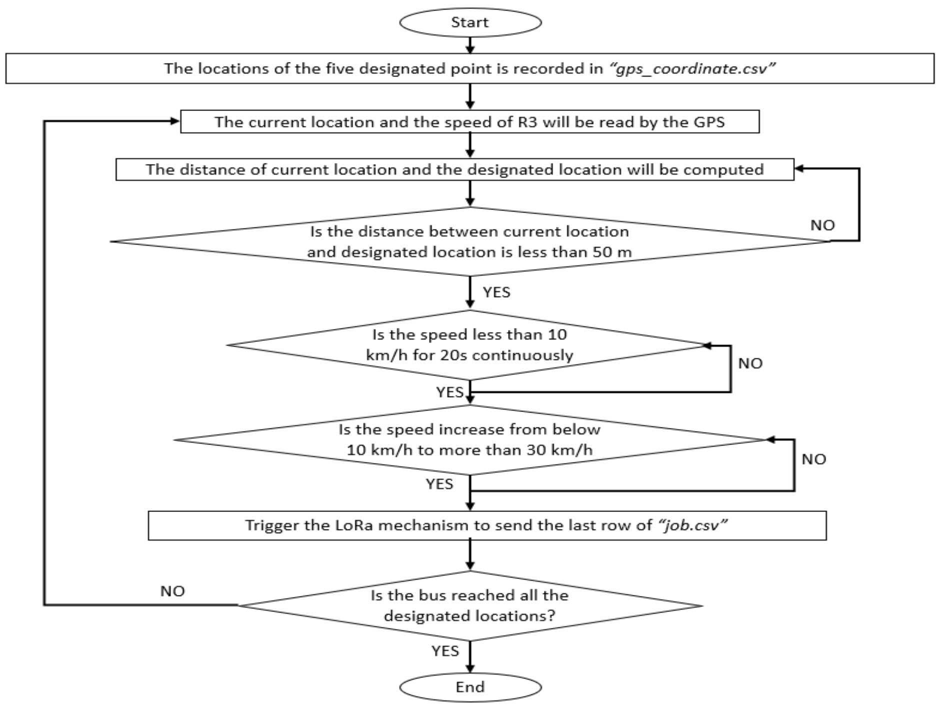

Start
The locations of the five designated point is recorded in "gps_coordinate.csv"
The current location and the speed of R3 will be read by the GPS
The distance of current location and the designated location will be computed
Is the distance between current location and designated location is less than 50 m
NO
YES
Is the speed less than 10 km/h for 20s continuously
NO
YES
Is the speed increase from below 10 km/h to more than 30 km/h
NO
YES
Trigger the LoRa mechanism to send the last row of "job.csv"
NO
Is the bus reached all the designated locations?
YES
End

Figure 3.28: LoRa Triggering Mechanism.

3.8 Edge AI Wi-Fi Deployment

Figure 3.29 shows the deployment of the edge AI Wi-Fi. As mentioned earlier, R3 have both LoRa and Wi-Fi connectivity. N0 will be placed at KB ground floor pantry, N4 at KB 8th floor pantry, and N8 at KB 8th floor office. In this configuration, the R3 will send the number of passengers of each stop to N10, thus synchronising to N4 and N8 after R3 reaches the UTAR bus stop.

Unlike in the LoRa sending mechanism, where only five packets of data will be sent to N4, R3 will send every updated row in "*job.csv*" to N10.

3.9 Evaluation of the Results

In this section, the metrics used to evaluate the results of this project will be introduced.

3.9.1 AI Models

3.9.1.1 Mean Average Precision (mAP)

Average precision is a widely used evaluation metric to evaluate the object detectors' accuracy, considering both precision and recall of the predictions.

Precision, which is defined as the ratio of true positives to the total number of positive predictions made by the model, is a key component of this metric. Its formula is shown in equation (3.2).

$$Precision = \frac{TP}{TP+FP} \qquad (3.2)$$

where

TP = True Positives

FP = False Positives

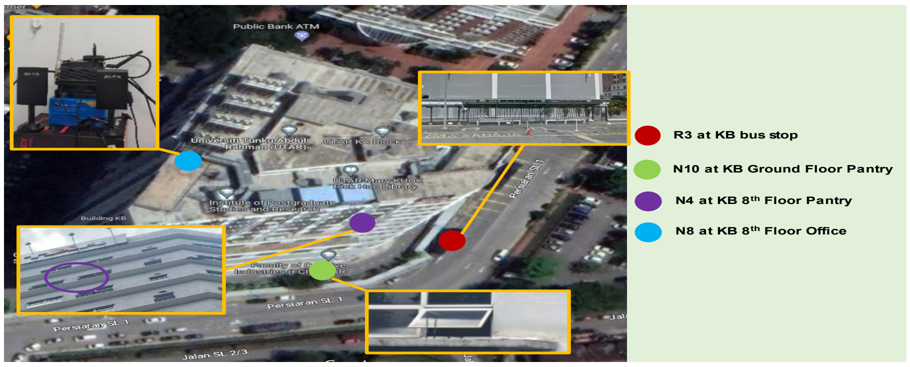

Public Bank ATM
Universiti Tunku Abdul Rahman (UTAR)
UTAR KA Block
UTAR Mary Kuok Pick Hoon Library
Institute of Postgraduate Studies and Research
Building KB
Persiaran SL 1
Jalan SL 2/3
● R3 at KB bus stop
● N10 at KB Ground Floor Pantry
● N4 at KB 8th Floor Pantry
● N8 at KB 8th Floor Office

Figure 3.29: Edge AI Wi-Fi Deployment.

On the other hand, the recall metric measures the ability of an object detector to identify all positive instances in the dataset. It is calculated as the ratio of true positives to the total number of positive instances, and its formula is presented as (3.3).

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

where
TP = True Positives
FN = False Negatives

In short, To summarize, precision evaluates the accuracy of the model's predictions, while recall evaluates the model's ability to detect all positive instances.

Intersection over union (IoU) is related to mAP. IoU refers to the measurements of the overlap between two boundaries. IoU quantifies the degree of overlap between the predicted boundary and the ground truth and is computed using the formula (3.4). The illustration of poor, good and excellent IoU is shown in Figure 3.30.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} \tag{3.4}$$

When IoU varies within a range, the precision and recall will change dynamically, and the precision-recall (PR) curve is constructed (Yin et al., 2021). The average precision (AP) can be viewed as the area under the PR curves by setting the IoU to different values. In this project, the evaluations of mAP will be carried out at IoU = 0.50. The formula of average precision is shown as (3.5). After the computation of AP is done for each class, the mean of the AP is computed. The formula of mAP is shown as (3.6).

$$AP = \int_0^1 p(r)\, dr \tag{3.5}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{3.6}$$

where

N = number of classes

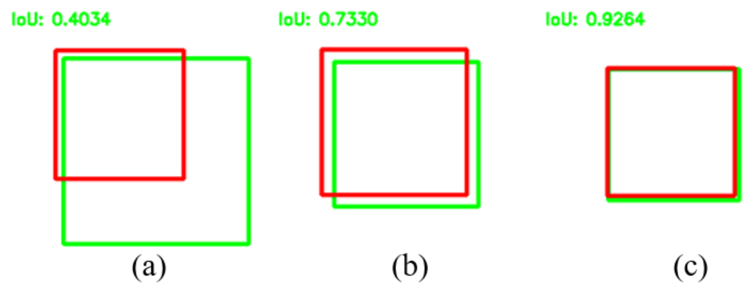


Figure 3.30: Illustration of (a) Poor, (b) Good, and (c) Excellent IoU.

3.9.1.2 Frames Per Second (FPS)

FPS measures how many images or frames can be processed in one second. FPS is an important metric when dealing with the real-time application. The formula of FPS is shown as (3.7).

$$FPS = \frac{total\ number\ of\ frames}{total\ processing\ time} \tag{3.7}$$

3.9.1.3 Accuracy

The accuracy formula for the AI models is shown as (3.8). Accuracy is the ratio of how many correct predictions are made by the detector to the ground truth value.

$$Accuracy\ (\%) = \frac{Total\ Number\ of\ Predictions}{Ground\ Truth\ Value} \times 100\ \% \tag{3.8}$$

3.9.2 LoRa

3.9.2.1 Packet Delivery Ratio (PDR)

The PDR, or Packet Delivery Ratio, is determined by dividing the number of received packets by the total number of packets transmitted by the publisher. The formula for PDR is displayed as (3.9).

$$PDR = \frac{Number\ of\ Packets\ Received}{Total\ Number\ of\ Packets\ Sent} \tag{3.9}$$

3.9.2.2 Hop Count

Hop count refers to how often the LoRa packet is relayed between the LoRa nodes in the network. When the packet is relayed by one node, the hop count will increase.

3.9.2.3 Received Signal Strength Indicator (RSSI)

RSSI is a metric that quantifies the signal strength of the transmitted data received by the subscriber node from the publisher. The unit of RSSI is dBm. The higher value of RSSI indicates that the signal is stronger and vice versa.

3.9.2.4 Time on Air

Time on air refers to the time the packets reach the subscriber nodes from the publisher node. The formula of time on air is shown as (3.10).

$$time\ on\ air = Initiation\ Time\ from\ Publisher - Received\ Time\ of\ Subscriber \quad (3.10)$$

3.9.3 Wi-Fi

3.9.3.1 Total Packets Received

The formula of total packets received is shown as (3.11). It is the ratio of packets received to the number of packets sent.

$$Total\ Packets\ Received\ (\%) = \frac{Number\ of\ Packets\ Received}{Total\ Number\ of\ Packages\ Sent} \times 100\% \quad (3.11)$$

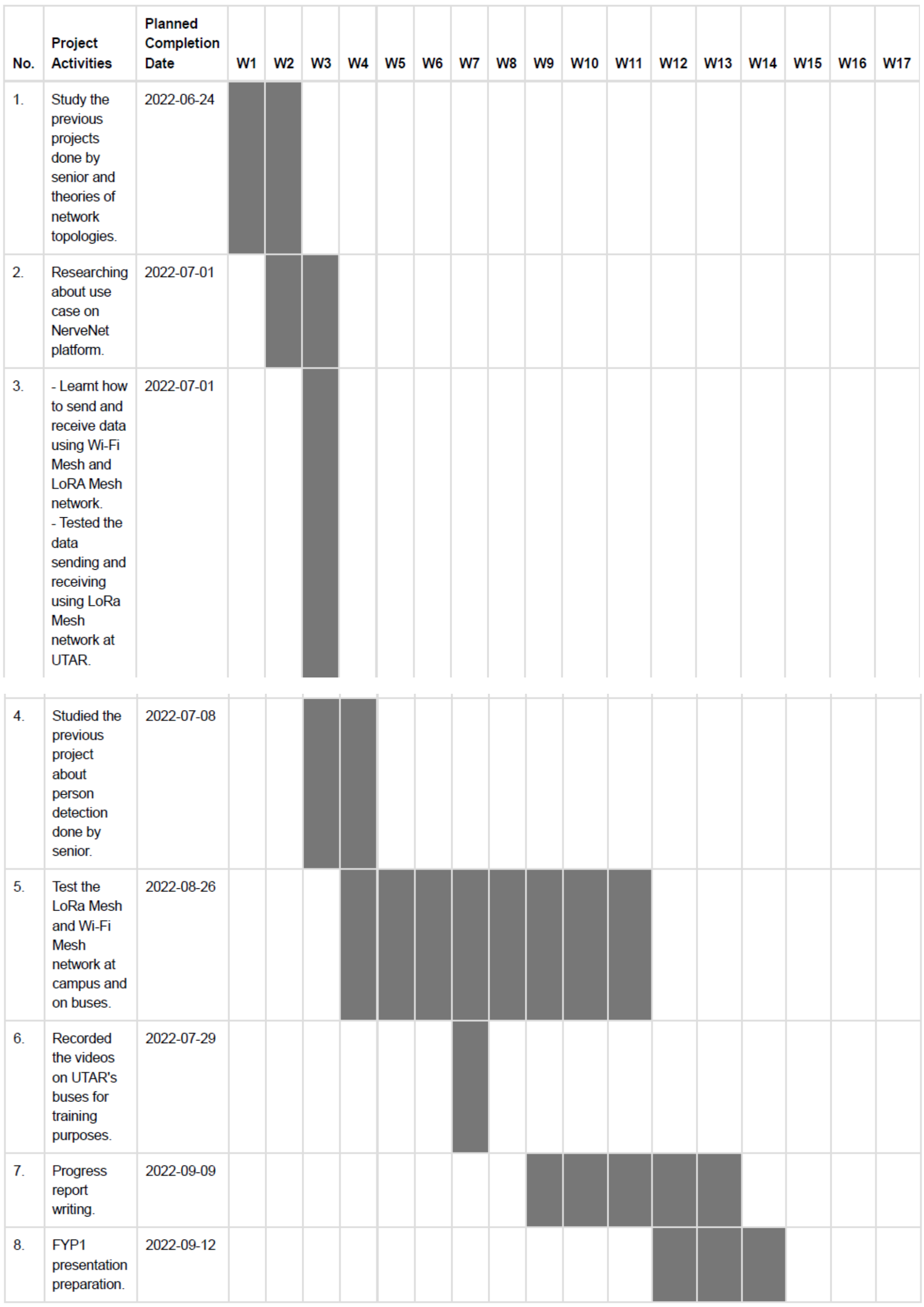
3.9.3.2 Latency

Latency refers to the time it takes for data to travel from its source to its destination in a network. The formula of latency in this project is shown as (3.12).

$$latency = time_{sync} - time_{update} \quad (3.12)$$

3.10 Work Plan

The Gantt Chart of this project is shown in this section. Figure 3.31 shows the Gantt Chart for Part 1, and Figure 3.32 shows the Gantt Chart for Part 2.



No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17
1.	Study the previous projects done by senior and theories of network topologies.	2022-06-24	■	■															
2.	Researching about use case on NerveNet platform.	2022-07-01		■	■														
3.	- Learnt how to send and receive data using Wi-Fi Mesh and LoRA Mesh network. - Tested the data sending and receiving using LoRa Mesh network at UTAR.	2022-07-01			■														
4.	Studied the previous project about person detection done by senior.	2022-07-08			■	■													
5.	Test the LoRa Mesh and Wi-Fi Mesh network at campus and on buses.	2022-08-26				■	■	■	■	■	■	■	■						
6.	Recorded the videos on UTAR's buses for training purposes.	2022-07-29							■										
7.	Progress report writing.	2022-09-09									■	■	■	■	■				
8.	FYP1 presentation preparation.	2022-09-12												■	■	■			

Figure 3.31: Gantt Chart for Part 1.

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17
1.	- Research about Yolov5. - Research about Yolov7.	2023-02-10	■	■															
2.	- Collect dataset from various online sources. - Annotate the images into the required format.	2023-02-18	■	■	■														
3.	- Training for Yolov5. - Training for Yolov7.	2023-02-24			■	■													
4.	- Record videos on UTAR bus for testing purpose. - Convert the AI weights into IR format of OpenVINO.	2023-03-11					■	■											
5.	- Retrain YOLOv5 algorithm. - Retrain YOLOv7 algorithm. - Evaluate the performance of YOLOv5 and YOLOv7.	2023-03-25							■	■									
6.	Training YOLOv4 for overhead passenger counting.	2023-04-01								■	■								
7.	Prepare poster for competition.	2023-04-15								■	■	■							
8.	Evaluation of network deployment	2023-04-22											■	■					
9.	Finalizing the project.	2023-04-29										■	■	■					
10.	Final Report Writing	2023-04-29										■	■	■	■				
11.	Final Presentation Preparation.	2023-04-30											■	■	■	■			

Figure 3.32: Gantt Chart for Part 2.

3.11 Summary

This chapter discussed this project's overall methodology and work plan, including the hardware required, AI training, AI model optimization, implementation of DeepSORT, the edge AI testbed, the edge AI LoRa deployment, and the edge Wi-Fi deployment. Besides, the metrics used in evaluations in this project were introduced. In addition, the Gantt Chart of the work plan for this project is also shown in this chapter.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter will tabulate the results obtained in this project. In addition, discussions based on the results will be provided.

4.2 Passenger Counting Application

4.2.1 Training Results

The training results of the passenger counting application will be compared between YOLOv4 and YOLOv4-tiny. As mentioned in Chapter 3, both models will run a training of 6,000 iterations. Figure 4.1 shows the mAP at a threshold of 0.5 against the number of iterations for YOLOv4. Figure 4.2 shows the mAP at a threshold of 0.5 against the number of iterations for YOLOv4-tiny.

For YOLOv4, the highest mAP at a threshold of 0.5 is 95.81 %, and for YOLOv4-tiny is 93.43 %. These two weights will be chosen for testing in the next section. The training performance of YOLOv4 is higher than YOLOv4-tiny because YOLOv4 were developed from 137 convolutional layers while YOLOv4-tiny only developed from 29 convolutional layers. Besides, the weights of YOLOv4 are also heavier than YOLOv4.

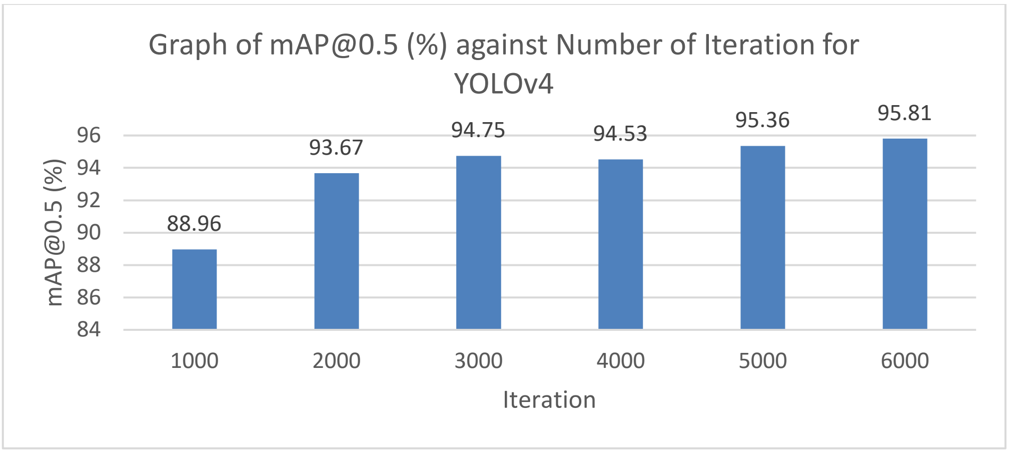


Figure 4.1: Graph of mAP@0.5 (%) Against Number of Iteration for YOLOv4.

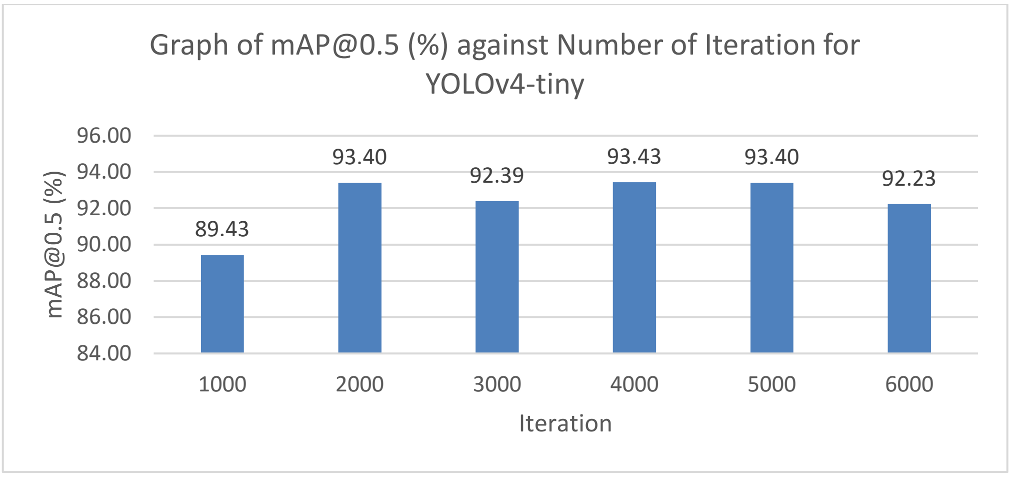


Figure 4.2: Graph of mAP@0.5 (%) Against Number of Iteration for YOLOv4-tiny.

In addition, the best weights of YOLOv4 and YOLOv4-tiny were used to run on three different videos to obtain both models' performance in terms of frames per second (FPS). Table 4.1 shows the FPS obtained by YOLOv4 and YOLOv4-tiny after running the three videos. The average FPS for YOLOv4 is 59.27, and for YOLOv4-tiny is 256.0. The results show that YOLOv4-tiny is 4.3 times faster than YOLOv4 when running on the Intel NUC with an external NVIDIA GeForce GTX 1080 Ti GPU.

4.2.2 Testing Results

The best weights for YOLOv4 and YOLOv4-tiny were used for testing in this section. A total of 16 images were tested where the image might consist of more than one passenger. Table 4.2 shows the testing accuracy on 16 images for YOLOv4 and YOLOv4-tiny. The results show that YOLOv4 has slightly higher accuracy than YOLOv4-tiny. The difference between the two models is only 1.09 %.

Table 4.1: FPS of YOLOv4 and YOLOv4-tiny.

Video	FPS of YOLOv4	FPS of YOLOv4-tiny
Video 1	58.10	247.0
Video 2	59.50	259.4
Video 3	60.20	261.7

Table 4.2: Accuracy of YOLOv4 and YOLOv4-tiny.

Model	Accuracy (%)
YOLOv4	84.10
YOLOv4-tiny	83.01

However, the accuracy obtained is not a satisfactory result. Hence, further analysis was done on the results obtained. All the images tested were further analysed to find out the reason that caused the unsatisfactory results. After the analysis, there was a huge difference in accuracy when the images tested were in good and poor lighting conditions.

Figure 4.3 shows the ground truth and predicted result for YOLOv4 under good lighting conditions, while Figure 4.4 shows the ground truth and predicted result for YOLOv4 under poor lighting conditions. In addition, Figure 4.5 shows the ground truth and predicted result for YOLOv4-tiny under good lighting conditions, while Figure 4.6 shows the ground truth and predicted result for YOLOv4-tiny under poor lighting conditions.

From the four images mentioned above, it is clear that both YOLOv4 and YOLOv4-tiny perform much better under good lighting conditions. The accuracy obtained above was further divided into good and poor lighting conditions. The results of the comparison are shown in Figure 4.7. From Figure 4.7, the results show that the accuracy of both YOLOv4 and YOLOv4-tiny is above 94 % under good lighting conditions. However, both models' accuracy is slightly above 70 % under poor lighting conditions. Hence, it can be concluded that the lighting conditions of the bus are important to determine the accuracy of the smart application.

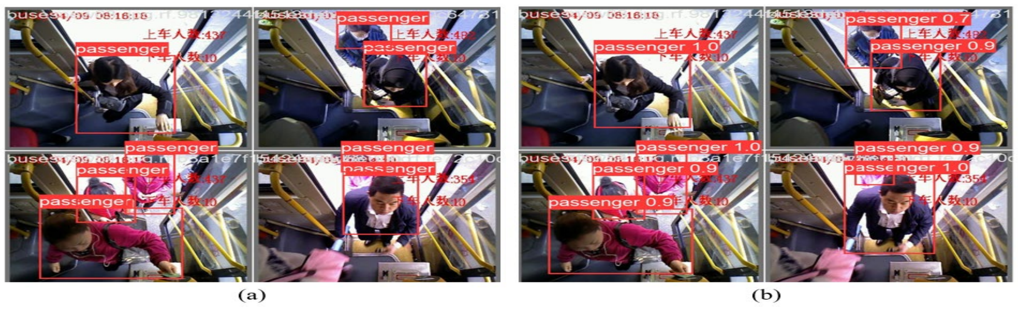


Figure 4.3: (a) Ground Truth and (b) Predicted Result for YOLOv4 Under Good Lighting Conditions.

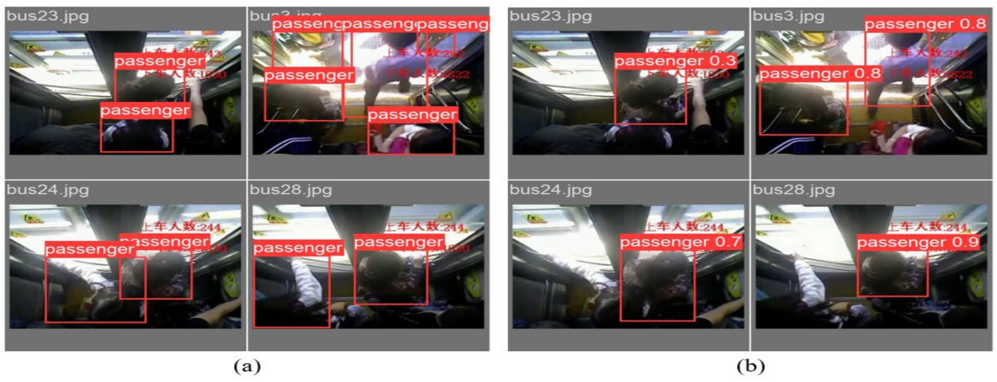


Figure 4.4: (a) Ground Truth and (b) Predicted Result for YOLOv4 Under Poor Lighting Conditions.

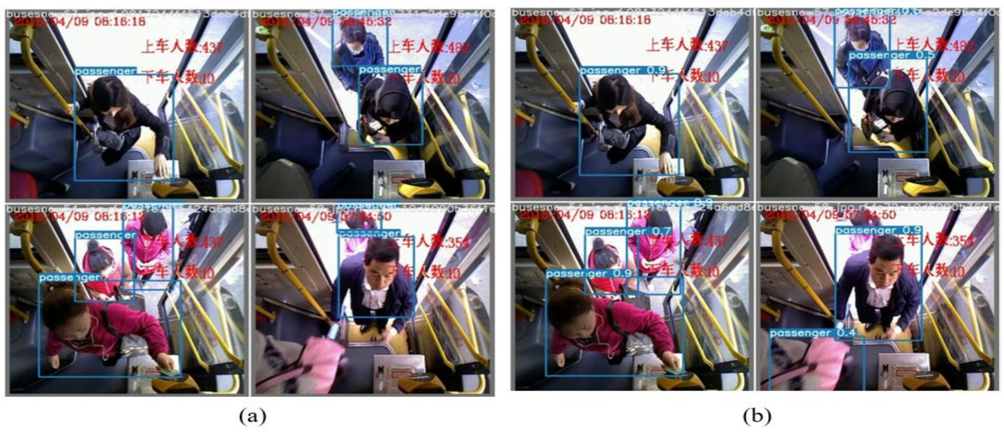


Figure 4.5: (a) Ground Truth and (b) Predicted Result for YOLOv4-tiny Under Good Lighting Conditions.

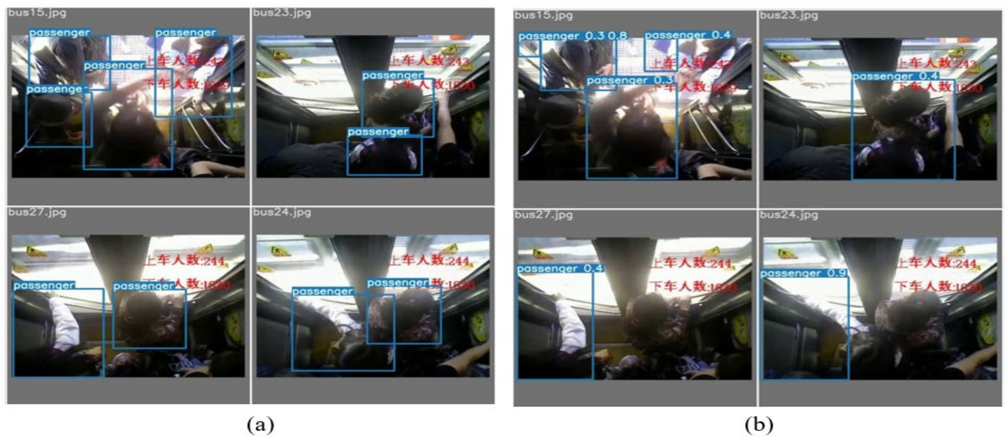


Figure 4.6: (a) Ground Truth and (b) Predicted Result for YOLOv4-tiny Under Poor Lighting Conditions.

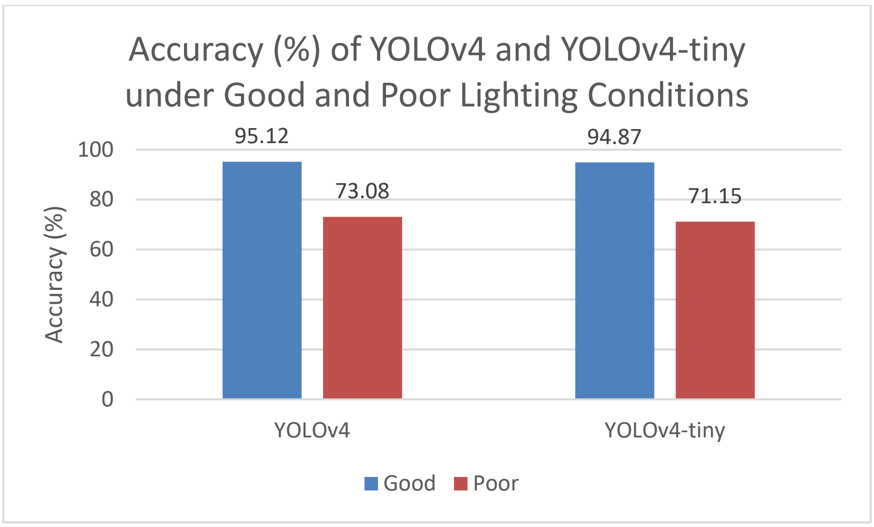


Figure 4.7: Accuracy of YOLOv4 and YOLOv4-tiny under Good and Poor Lighting Conditions.

4.2.3 Deployment Results

After the comparison of YOLOv4 and YOLOv4-tiny was made earlier, the result of the deployment of the person counting application will be tabulated and discussed here.

The YOLOs weights were converted to OpenVINO for AI model optimization. The performances of YOLOv4 and YOLOV4-tiny in IR format in terms of FPS were computed by running the models on the three videos used previously. However, the inferencing was run on R3 and not Intel NUC with an external GPU to test the performance of the deployment on edge devices.

Table 4.3 shows the FPS of YOLOv4 and YOLOv4-tiny for each video run on the edge device. Besides, Figure 4.8 shows the average FPS of both models after the conversions to IR format were done. The results show that YOLOv4-tiny has around ten times faster FPS than YOLOv4.

Table 4.3: FPS of YOLOv4 and YOLOv4-tiny on Edge Device.

Video	FPS of YOLOv4	FPS of YOLOv4-tiny
Video 1	1.011	9.820
Video 2	1.019	10.19
Video 3	1.032	9.910

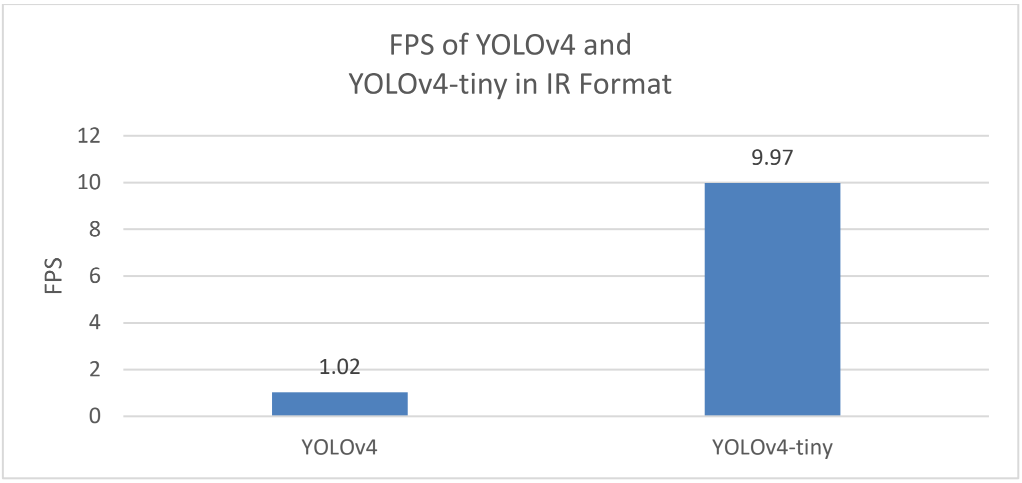


Figure 4.8: Average FPS of YOLOv4 and YOLOv4-tiny on Edge Device.

Hence, YOLOv4-tiny was chosen as the AI model for passenger detection and counting application due to its much better performance in terms of FPS. YOLOv4-tiny was integrated with DeepSORT before deploying the smart application on UTAR's bus. The result of deploying the smart application on UTAR's bus is shown in Figure 4.9. The results show that the application achieved 90 % accuracy for counting the passenger going into the bus and 80 % for the passenger going out. Further analysis was done on the results obtained.

Figure 4.10 shows two results obtained from UTAR's bus under poor lighting conditions. As shown in Figure 4.10, the YOLOv4-tiny algorithm failed to detect the passenger under poor lighting conditions. This caused the tracking not to occur and the passenger to not be counted. Hence, the accuracy of this application is affected.

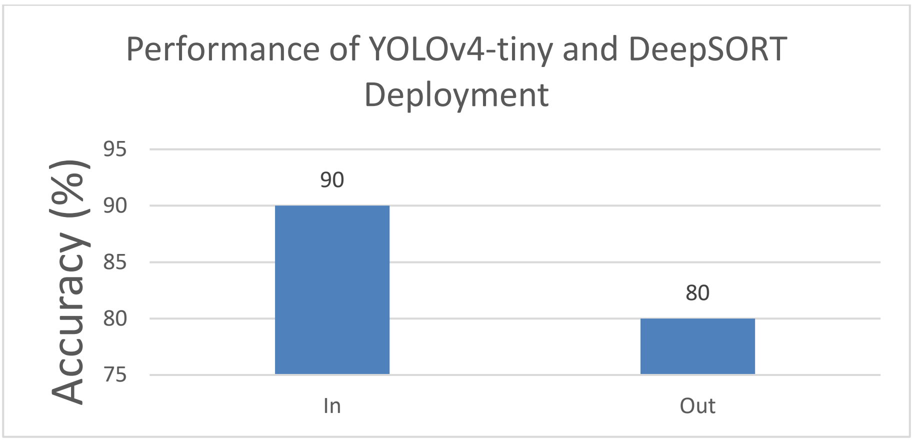


Figure 4.9: Performance of YOLOv4-tiny and DeepSORT Deployment.

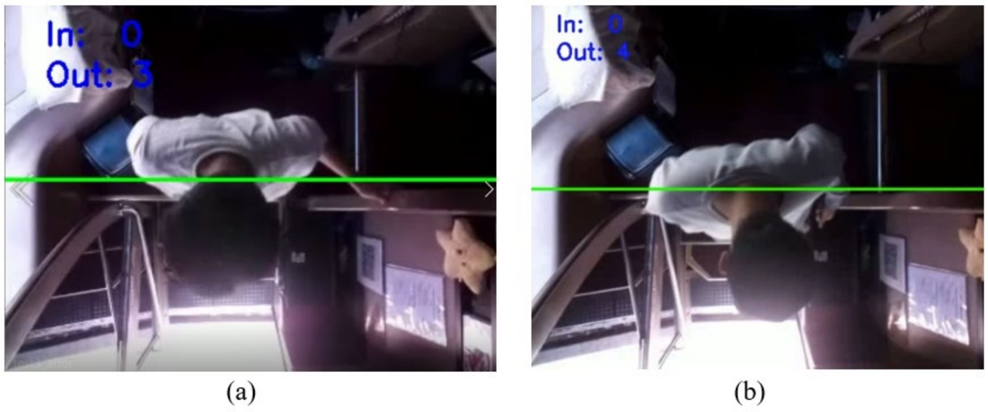


Figure 4.10: Results Obtained from UTAR's Bus under Poor Lighting Condition (a) First Example and (b) Second Example.

4.3 Passenger Gender Recognition

4.3.1 Training Results

The training results of the passenger gender recognition and detect whether the passenger is wearing the mask properly will be compared between YOLOv5 and YOLOv7. As mentioned in Chapter 3, both YOLOv5 and YOLOv7 will run the training for 300 epochs. Hence, the performance between these two models is comparable.

Figure 4.11 shows the performance of YOLOv5 and YOLOv7. YOLOv7 has a higher mAP at the threshold of 0.5 compared to YOLOv5. However, the difference is only 2.21 %, which is acceptable.

Furthermore, the benchmarking of both YOLOv5 and YOLOv7 was done using OpenVINO. The results of the benchmarking are shown in Figure 4.12. YOLOv5 achieves 8.56 FPS, much higher than YOLOv7 with only 1.74 FPS. This result indicates that YOLOv5 is more suitable for real-time application.

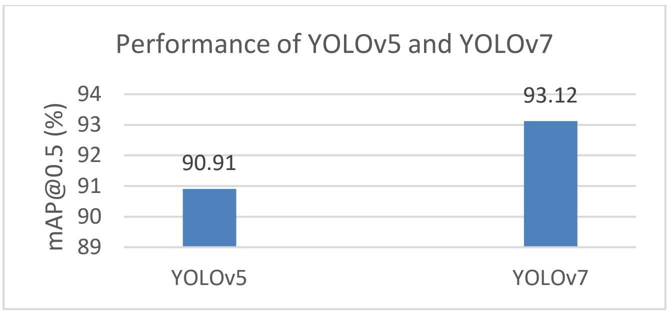


Figure 4.11: Performance of YOLOv5 and YOLOv7.

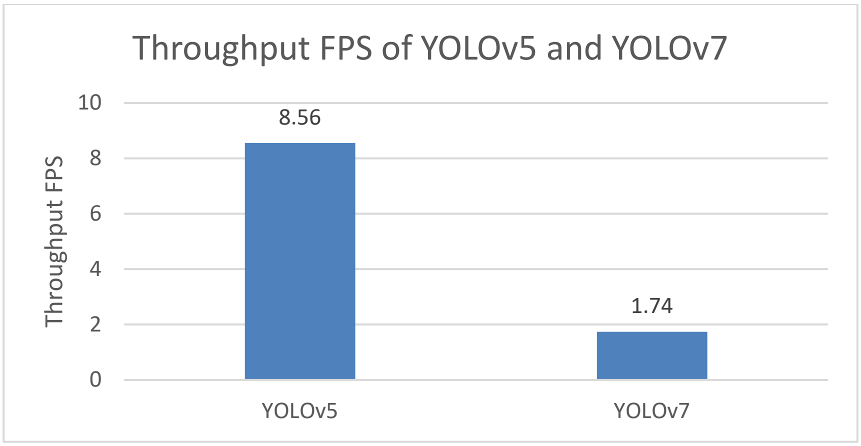


Figure 4.12: Throughput FPS of YOLOv5 and YOLOv7.

4.3.2 Testing Results

YOLOv5 and YOLOv7 were tested on 3 images from each class, a total of 24 images. Figure 4.13 shows part of the images tested for YOLOv5, and Figure 4.14 shows the images tested for YOLOV7.

The testing results for both YOLOv5 and YOLOv7 are impressive. Both models achieved 100 % accuracy, as shown in Figure 4.15 for the testing.

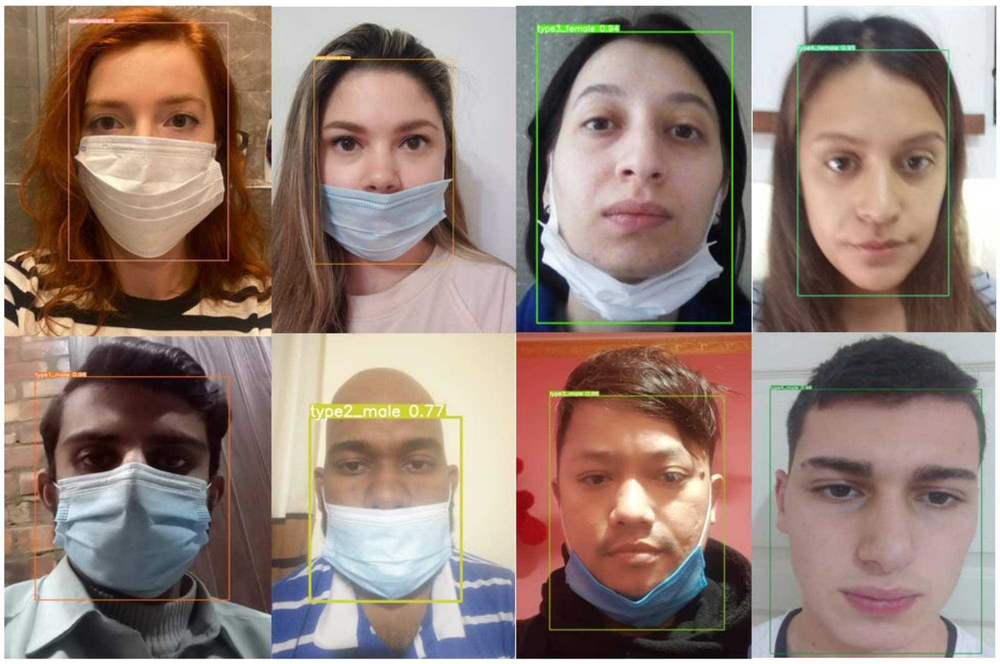


Figure 4.13: Images Tested for YOLOv5 (Kucev, 2021).

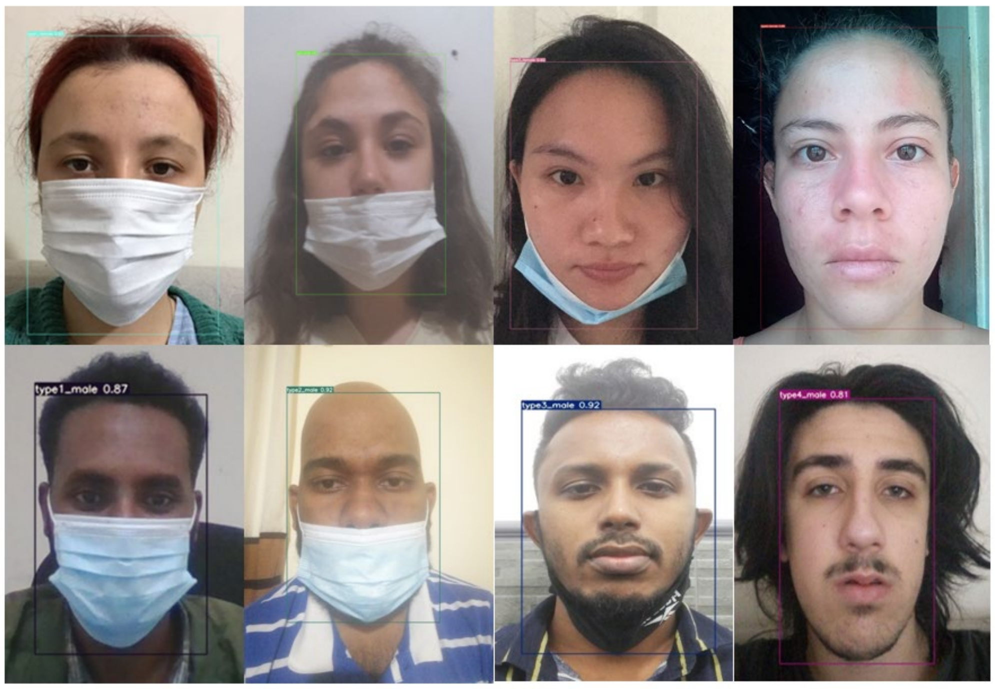


Figure 4.14: Images Tested for YOLOv7 (Kucev, 2021).

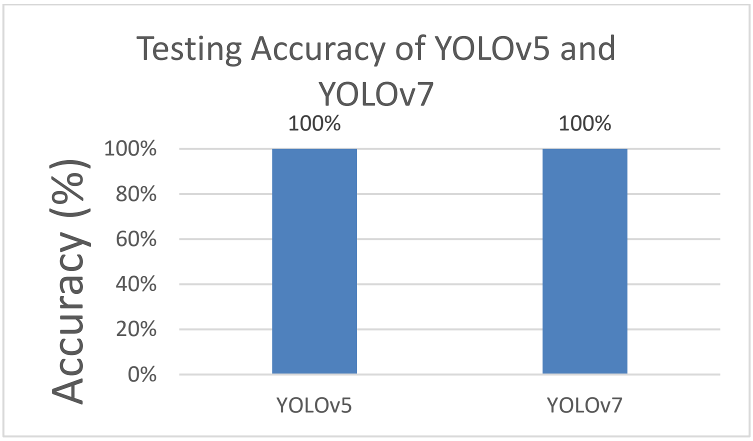


Figure 4.15: Testing Accuracy of YOLOv5 and YOLOv7.

4.4 Wireless Networking (LoRa)

The LoRa wireless networking system was evaluated based on PDR, hop count, RSSI, and time on air.

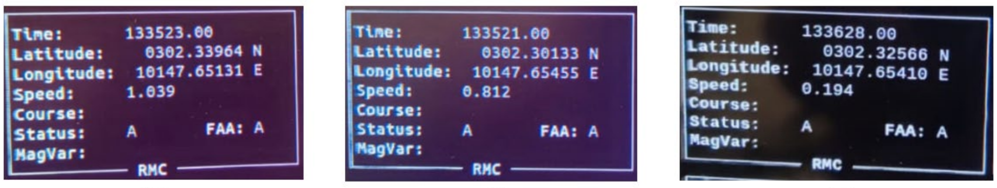
According to Chapter 3, the LoRa mechanism will be triggered based on the current locations and speed of the bus. However, this mechanism failed in this project due to the inaccurate reading of GPS. Figure 4.16 shows the three readings taken using the same GPS module but on different devices. The first and second readings were read on Ubuntu 18.04 LTS OS, while the third was read on Raspberry Pi OS. Although the GPS was put at exactly the same point for the three readings, the values of latitude and longitude were different in each reading. In addition, although the GPS was fixed at the same point, the GPS's speed was not zero. These inaccurate readings of GPS caused the LoRa trigger mechanism to fail in this project. Hence, manual sending of LoRa was applied at each designated location.

Figure 4.17 shows the PDR of LoRa testing in this project. A total of 10 packets of data were sent during the testing. R1 received two packets, R2 received zero packets, R4 and R5 received four, respectively, and N4 received two. Further analysed of the log record of the LoRa was done to find out the reason behind the poor performance of the LoRa network.

Firstly, one of the factors that R2 received zero packets is due to its direction. R2 was located at one of the units at Forest Green Condominium, Sungai Long. The direction of R2 is shown in Figure 3.27 and Figure 4.18. The direction of R2 facing is parallel to the bus route. Unlike R4 and R5, which had the highest PDR, both the LoRa nodes face perpendicular to the bus route.

Besides, when the bus passes R5, no housing area blocks between the publisher and subscriber nodes, as shown in Figure 4.19. However, there is a housing area R2 was facing, as shown in Figure 4.18. The housing area might cause electromagnetic interference to the LoRa network because many devices are connected to other wireless networking systems in the housing area.

In addition to the interference of the housing area, although R4 also faces a housing area, it can receive more packets than R2. The direction of R4 is shown in Figure 4.20. One of the differences between R2 and R4 is that R4 has less electromagnetic interference than R2 because R4 only faces one row of the housing area.



```
Time:        133523.00
Latitude:      0302.33964 N
Longitude:   10147.65131 E
Speed:       1.039
Course:
Status:      A          FAA: A
MagVar:
             RMC
```

(a)

```
Time:        133521.00
Latitude:      0302.30133 N
Longitude:   10147.65455 E
Speed:       0.812
Course:
Status:      A          FAA: A
MagVar:
             RMC
```

(b)

```
Time:        133628.00
Latitude:      0302.32566 N
Longitude:   10147.65410 E
Speed:       0.194
Course:
Status:      A          FAA: A
MagVar:
             RMC
```

(c)

Figure 4.16: (a) First Reading, (b) Second Reading, and (c) Third Reading.

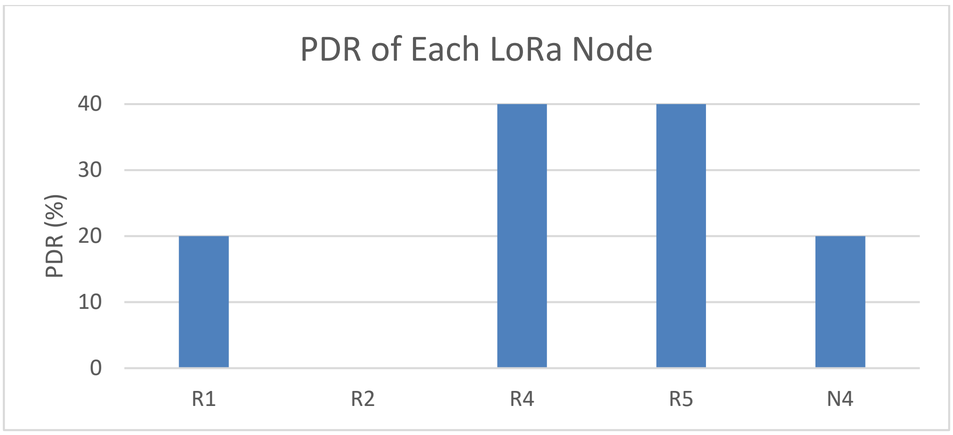


Figure 4.17: PDR of Each LoRa Node.

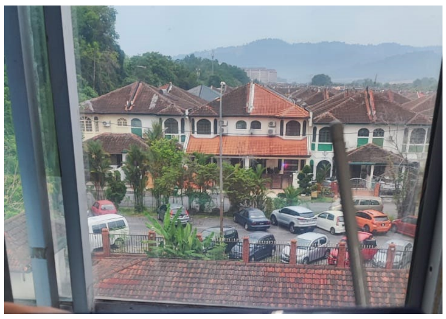


Figure 4.18: The Direction of R2.

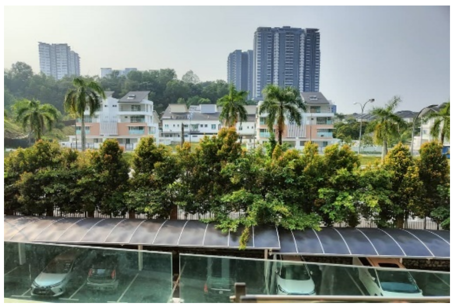


Figure 4.19: The Direction of R5.

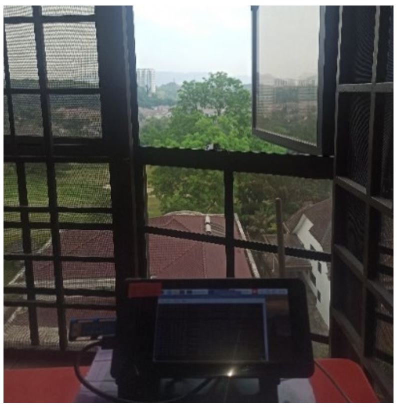


Figure 4.20: The Direction of R4.

Moreover, another significant factor is that the altitude of R4 is higher than R2. According to Google Earth's data, Forest Green Condominium is 56 m above sea level, while Flora Green Condominium is 68 m above sea level. Flora Green Condominium is the location where R4 was placed. Also, R2 was placed on the second floor, while R4 was placed on the fifth floor. According to Choi, Lee, and Lee (2020), the performance of LoRa will drop when the areas consist of many obstacles, such as trees. Most obstacles can be avoided if the LoRa nodes are put at a higher altitude.

For R1, one of the two packets received was via multi-hop, which will be discussed later. For N4, the two packets were received when the UTAR bus was leaving the UTAR bus stop, the first designated location. This is reasonable since N4 was located at KB's 8th-floor pantry, the nearest LoRa node from UTAR. Other than the first location, N4 received nothing from other locations. This is also reasonable since the N4 was not facing the bus route except for the first location, as shown in Figure 3.27 and Figure 4.22. It is difficult for N4 to receive data from other LoRa nodes due to the thick concrete wall of the UTAR campus and substantial electromagnetic interference around that area.

Figure 4.21 shows the hop count of each LoRa node at five designated locations listed in "*gps_coordinate.csv*". The five designated locations follow the UTAR afternoon bus route for Bandar Sungai Long and Palm Walk. The route is shown in Figure 4.22.

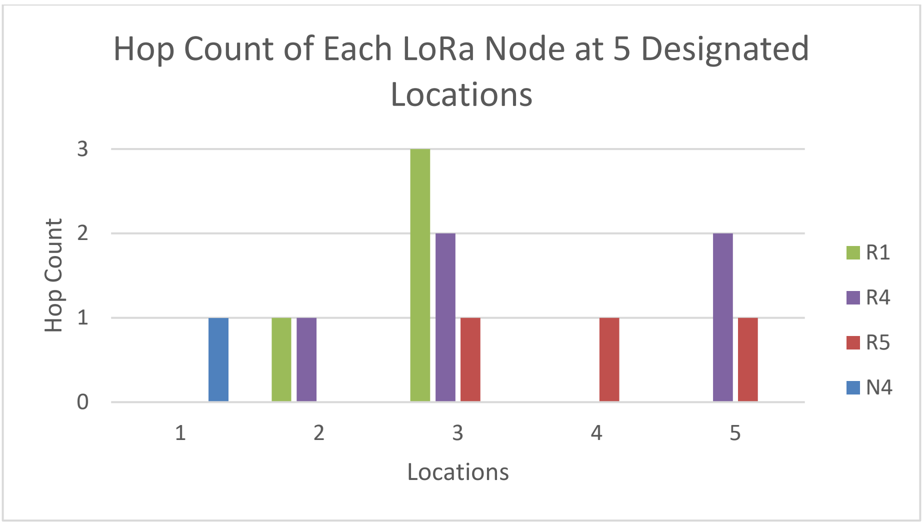


Figure 4.21: Hop Count of Each LoRa Node at 5 Designated Locations.

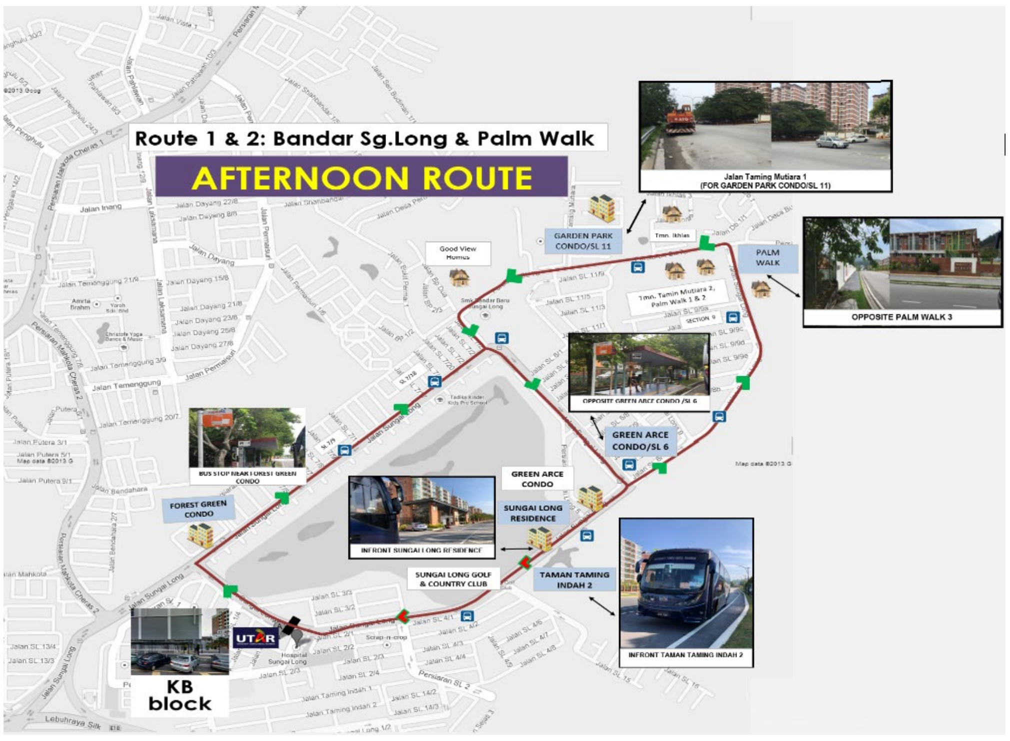

Route 1 & 2: Bandar Sg.Long & Palm Walk
AFTERNOON ROUTE
Jalan Taming Mutiara 1 (FOR GARDEN PARK CONDO/SL 11)
GARDEN PARK CONDO/SL 11
PALM WALK
OPPOSITE PALM WALK 3
OPPOSITE GREEN ARCE CONDO /SL 6
GREEN ARCE CONDO/SL 6
GREEN ARCE CONDO
SUNGAI LONG RESIDENCE
INFRONT SUNGAI LONG RESIDENCE
BUS STOP NEAR FOREST GREEN CONDO
FOREST GREEN CONDO
SUNGAI LONG GOLF & COUNTRY CLUB
TAMAN TAMING INDAH 2
INFRONT TAMAN TAMING INDAH 2
KB block

Figure 4.22: UTAR Bus Route for Bandar Sungai Long and Palm Walk.

As mentioned earlier, the packets at the first location were only received by N4, the nearest LoRa node to UTAR. N4 received the packets from location 1 with a single hop count. For location 2, only R1 and R4 received the packets sent. Both R1 and R4 are the nearest LoRa nodes from location 2. Both of them received the packets directly from R3, which is the subscriber. This indicates that there is only one hop count for both R1 and R4 in location 2.

Next, R1, R4, and R5 received the packets sent by R3 from location 3. Figure 4.21 shows that R1 has three hop counts, R4 has two hop counts, and R5 has one hop count at location 3. This indicates that the packets sent from R3 were received by R5 first. After that, R4 received the packets through R5, and R1 received the packets from R4.

Then, only R5 received the packet sent by R3 in location 4. Location 4 is the furthest location of the bus route. The distance of location 4 to the nearest LoRa node, R5, is 1.35 km, as shown in Figure 4.23. Hence, the distance is the main reason only one packet can be received from location 4. In addition, there are many housing areas and a high condominium building between locations 4 and R5, as shown in Figure 4.23.

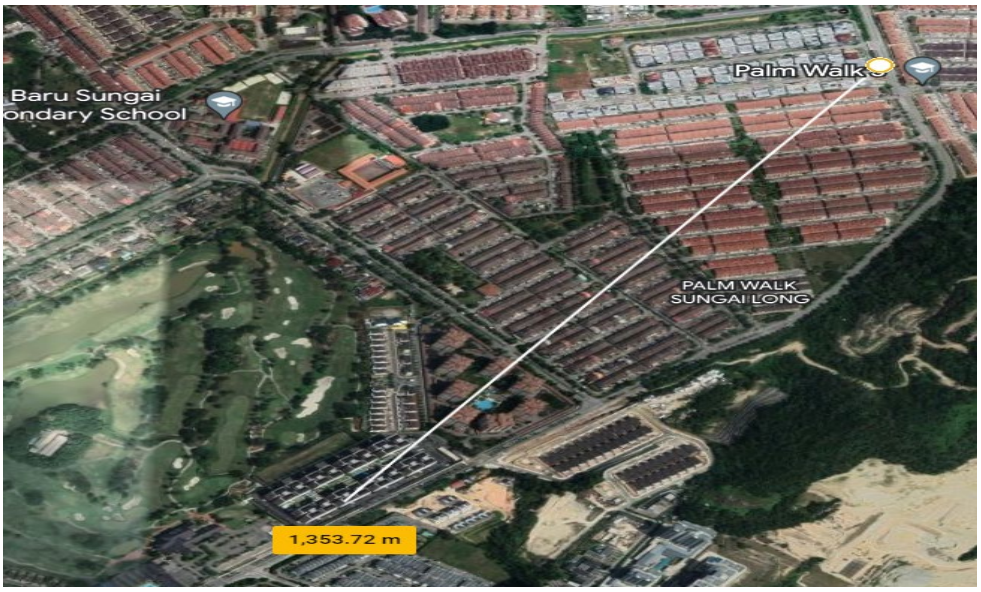
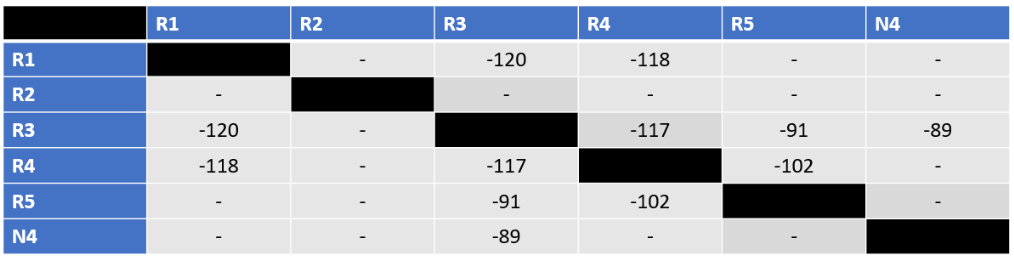


Figure 4.23: Distance between Location 4 and R5.

After that, the packets sent by R3 at location 5 were received by R5 and R4. R5 has one hop count, and R4 has two hop counts. This indicates that the packets sent from R3 were received by R5 first, and then R5 floods the packets to R4. This result is logical because R5 is the nearest LoRa node from location 5, and R4 is the nearest node next to R5.

Figure 4.24 shows the RSSI of the LoRa transmission in this project. The highest RSSI recorded is between N4 and R3. The result shows that the signal between N4 and R3 was the strongest. The second strongest signal existed between R3 and R5. The distance between R3 and R5 is short when the bus passes through R5, and there are not many obstacles blocking the signal, as shown in Figure 4.19, except for some short trees. The weakest signal existed between R3 and R1, where the packet was sent from location 2. The direction of R1 is shown in Figure 4.25. There are some vast trees between R3 and R1, although the distance between the two points is not too far. The vast trees might block the signal of LoRa and cause the weakest signal to exist between R3 and R1. According to Jebril et al. (2018), the blocking of the trees caused a near line of sight. The illustration of line-of-sight is shown in Figure 4.26.



	R1	R2	R3	R4	R5	N4
R1		-	-120	-118	-	-
R2	-		-	-	-	-
R3	-120	-		-117	-91	-89
R4	-118	-	-117		-102	-
R5	-	-	-91	-102		-
N4	-	-	-89	-	-	

Figure 4.24: RSSI or LoRa Transmission.

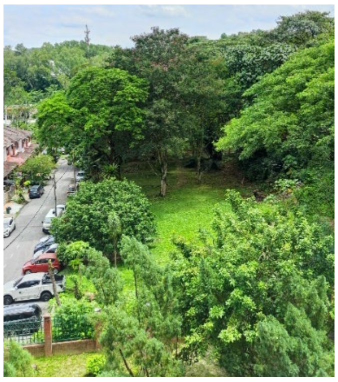


Figure 4.25: The Direction of R1.

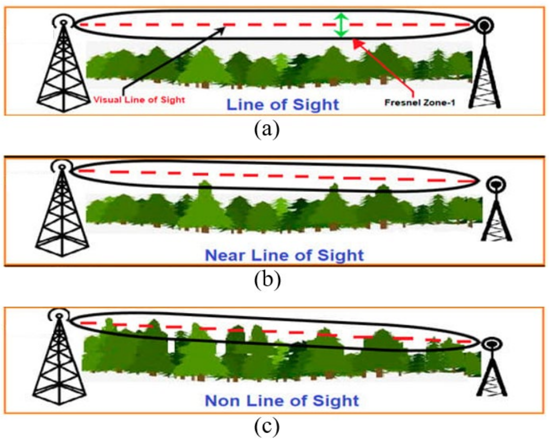


Figure 4.26: Illustration of (a) Line-of-sight, (b) Near Line-of-sight, and (c) Non Line-of-sight (Jebril et al., 2018).

Lastly, the average time on air of the LoRa transmission was computed. Figure 4.27 shows the average time on air of each LoRa node at five designated locations. For location 1, only 68.5 s were taken for R3 to send the packet to N4. For location 2, R3 took 176 s to send the packet to R1 and only 124 s to R4. The time differences might be caused by the vast trees' blockage between R3 and R1. There is no blockage between R3 and R4, as shown in Figure 4.20, due to the high altitude of R4.

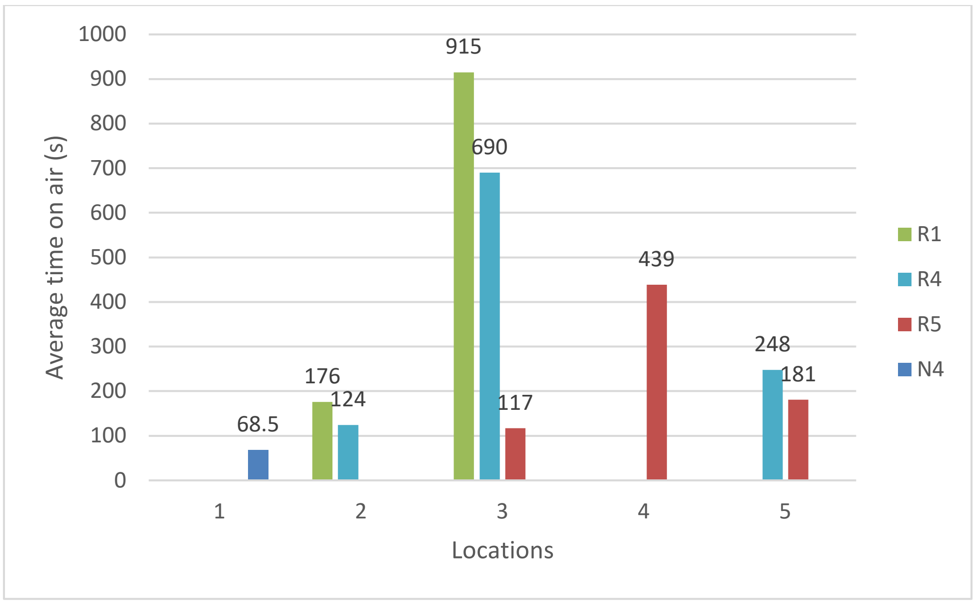


Figure 4.27: Average Time on Air of Each LoRa Node at 5 Designated Locations.

For location 3, R3 took 117 s to send the packet to R5. As mentioned earlier, R4 will receive the packet from R5, and R4 will send the packets to R1 afterwards. Hence, the average time on air of R1 in location 3 is the longest.

For location 4, R5 took 439 s to receive the packet from R3. The long time on air in this location might be due to the R5 actually receiving the packet somewhere else from R3 at the location nearer to R5 instead of the original location of R4. This is because the distance between location 4 and R5 is 1.35 km, as shown in Figure 4.23. According to Microchip Technology (Microchip Technology, 2023), the range of LoRa is up to 15 km in rural areas and more than 2 km in urban areas in an ideal situation. Additionally, there are many housing areas and a high building between locations 4 and R5. Hence, the housing areas might cause electromagnetic interference and blockage of the signal.

For location 5, R3 took 181 s to send the packets to the nearest LoRa node, R5. After that, R5 will send the packets to R4, as mentioned above. Hence, the time on air of R4 was longer than R5 due to this reason.

In short, the LoRa trigger mechanism failed in this project due to the inaccurate reading from the GPS. Then, the LoRa node R2 is useless in this deployment, and the reasons, including the direction and altitude of R2, were discussed.

4.5 Wireless Networking (Wi-Fi)

The Wi-Fi wireless networking system evaluation is based on the percentage of packets received and the latency. Figure 4.28 shows the packets received by each node. Figure 4.29 shows the average latency of each node.

In Wi-Fi networks, ten packets of data were sent from R3 to N10 from the UTAR bus stop, as shown in Figure 3.29. After that, the packets will be automatically synchronized across N4 and N8. N4 was located at KB block 8th-floor pantry, and N8 was at KB 8th-floor office, as shown in Figure 3.29. N4, N8, and N10 received all the ten packets sent to them, as shown in Figure 4.28.

Figure 4.29 shows that N4 has the highest average latency, followed by N8, and N10 has the lowest average latency. These results are logic based on the configuration of the Wi-Fi network. Based on the configuration, R3 will send the packets to N10 first. Hence, N10 will have the lowest average latency. After that, N10 will synchronize the data to N4, and N4 will synchronize to N8 based on the configuration. Hence, the results matched the configurations of the Wi-Fi network.

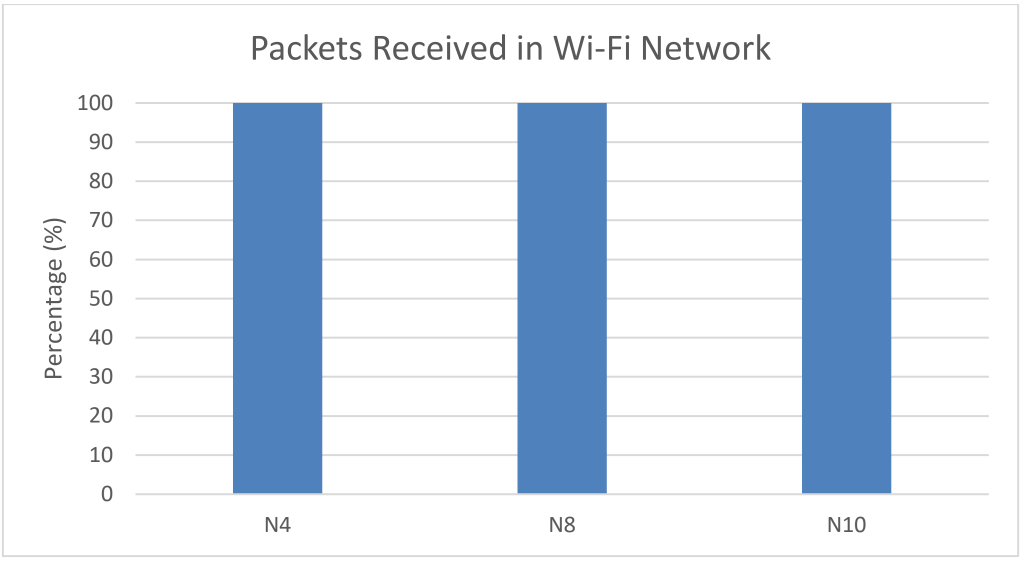


Figure 4.28: Packets Received by Each Node in Wi-Fi Network.

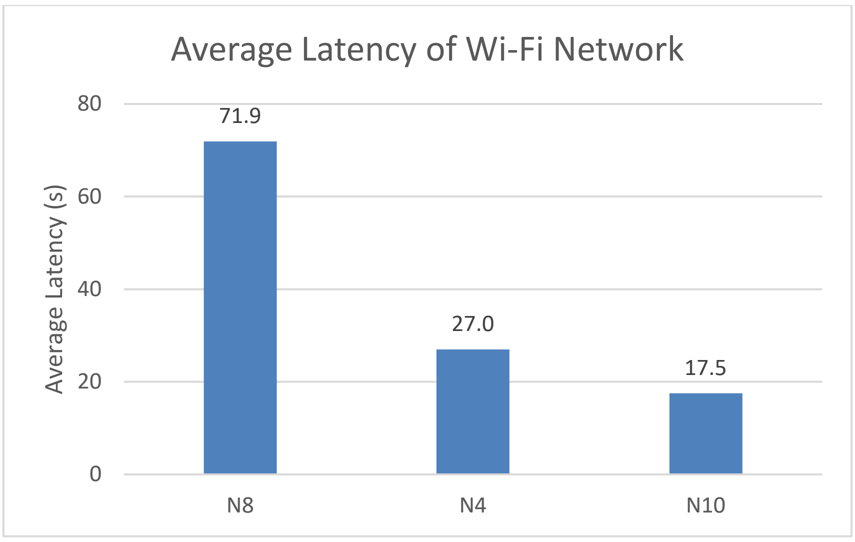


Figure 4.29: Average Latency of Wi-Fi Network.

4.6 Summary

In this chapter, the results obtained from this project are presented. In addition, discussions on the results obtained were provided.

For passenger counting applications, three main parts of results are presented: training, testing, and deployment. For passenger gender recognition applications, only two main parts of results were presented: training results and testing results. This is an additional package provided, and no deployment results were tested.

Besides, the LoRa results tabulated include PDR, hop count, RSSI, and time on air. The total packets received and average latency were presented for Wi-Fi wireless networking.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In this project, a smart application for passenger counting on UTAR buses was developed. Additionally, a passenger gender recognition algorithm was provided as an additional package to the smart application. After that, a NerveNet-based mesh network testbed with LoRa and Wi-Fi was deployed. Then, the smart application was integrated into the wireless network testbed. Finally, the experimental performance of the smart application and wireless network system was evaluated.

For the passenger counting application, the performance of YOLOv4 and YOLOv4-tiny were compared. YOLOv4 has higher mAP than YOLOv4-tiny in the training stage. However, YOLOv4-tiny has 4.3 times higher frames per second (FPS) than YOLOv4. For the testing results, YOLOv4 achieved 84.10 % accuracy, while YOLOv4-tiny achieved 83.01 %. Further analysis was done on the testing results obtained due to unsatisfactory results. The analysis shows that the unsatisfactory results are due to the poor lighting condition. YOLOv4 can achieve an accuracy of 95.12 % under good lighting conditions but only 73.08 % under poor lighting conditions. Besides, YOLOv4-tiny can achieve an accuracy of 94.87 % under good lighting conditions but only 71.15 % under poor lighting conditions. After that, YOLOv4 and YOLOV4-tiny were tested on the edge device to obtain the FPS. The results show that YOLOv4-tiny has almost ten times faster than YOLOv4 in terms of FPS. Hence, YOLOv4-tiny was chosen to deploy this application due to its better performance in terms of FPS. In the deployment stage, the average accuracy of the person counting application deployment is 85 %, and the unsatisfying performance is due to the poor lighting conditions of the UTAR bus.

For passenger gender recognition, YOLOv7 has a higher mAP than YOLOv5 in the testing stage. However, YOLOv5 has almost five times higher FPS than YOLOv7. In the testing stage, both YOLOv5 and YOLOv7 achieved

100 % accuracy. However, the FPS of YOLOv5 is almost five times faster than YOLOv7.

For the LoRa deployment, the LoRa triggering mechanism using the data from GPS failed due to the inaccurate reading of the GPS. The three latitude and longitude readings from the same location are different. Besides, the PDR for all nodes is less than 50 %. In addition, R2 received zero packets and is useless in this project for several reasons, including the directions and altitudes of R2, as explained earlier. Furthermore, the signal strength between R3 and R1 is the weakest. This result is caused by blocking trees, which causes the near line of sight. Hence, the overall performance of LoRa still has room for improvement.

For the Wi-Fi deployment, all the packets sent were received. Besides, the latency of every node is acceptable following the configuration. N10 recorded the lowest latency because it was nearest to R3, and N8 recorded the highest latency because it was the furthest node from R3.

5.2 Recommendations for future work

One of the recommendations for future work is to improve the lighting condition of the UTAR bus. Both the testing and deployment results prove that the lighting condition has enormous effects on the accuracy of the object detector. In addition, the analysis of the deployment results also shows that the object detector cannot detect the passenger on the UTAR bus under poor lighting. The lighting system can be improved by installing a new light and a sensor at the bus entrance. By using a sensor, the light can be automatically turned on when the bus's door opens. The automation process will not increase the burden on the bus driver.

Besides, the R2 should be removed from its current location. It can be put at the UTAR KA block to link the N4 and R4. The suggested modification is shown in Figure 5.1. The green point indicates R2, which can be put at the UTAR KA block. With that, the LoRa performance can be improved since the devices can link to each other better than the previous configuration.

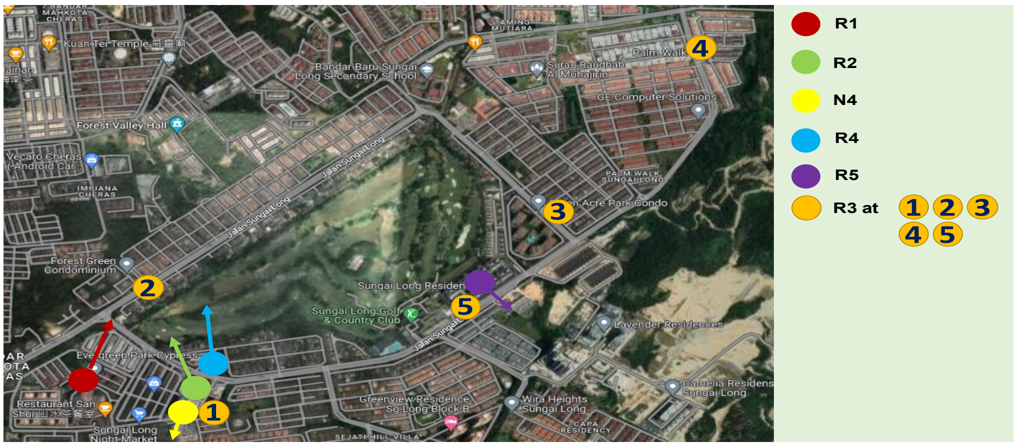


Figure 5.1: Suggestion for R2 Location (Green Point).

In addition, more LoRa nodes can be put around Bandar Sungai Long following the UTAR bus routes. By doing so, the packets can reach with lower time on air because more subscribers nodes can help to relay the packets. The suggested locations for more LoRa nodes are shown in Figure 5.2. The red point indicates the suggested new LoRa nodes.

Furthermore, a more accurate GPS module can be purchased to replace the existing one. An accurate GPS is essential in this project to provide accurate locations and speed readings to the LoRa nodes. If the GPS is accurate, the LoRa trigger mechanism will function well, and the smart application will run at full automation.

Lastly, a more advanced tracking algorithm can be applied in the future. For example, StrongSORT, the upgraded version of DeepSORT, can be the tracking algorithm in the future.

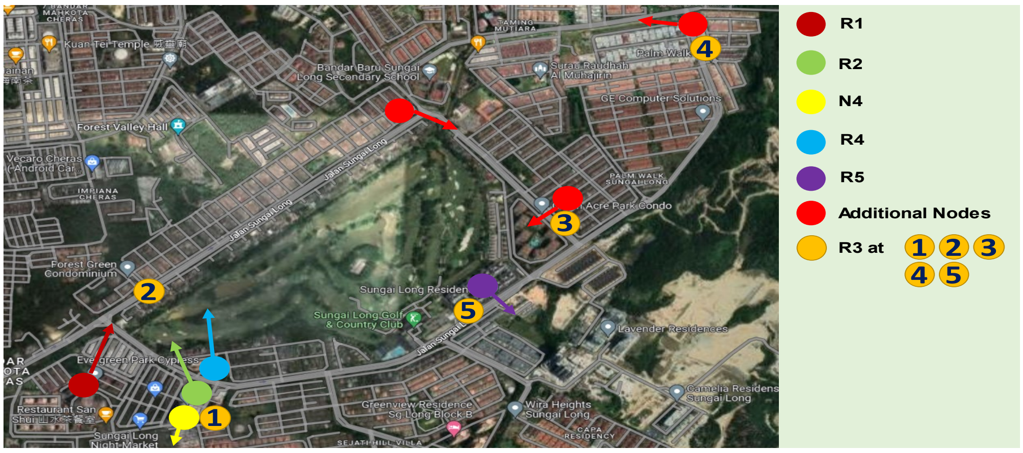


Figure 5.2: Suggested Additional LoRa Nodes (Red Point).

REFERENCES

Ahmad, M., Ahmed, I. and Adnan, A., 2019. Overhead View Person Detection using YOLO. In: *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE. pp.0627–0633. https://doi.org/10.1109/UEMCON47517.2019.8992980.

Ahmed, I. and Adnan, A., 2018. A Robust Algorithm for Detecting People in Overhead Views. *Cluster Computing*, 21(1), pp.633–654. https://doi.org/10.1007/s10586-017-0968-3.

Ali, N.A.A., Adilah, N. and Salimi, I., 2019. Performance of LoRa Network for Environmental Monitoring System in Bidong Island Terengganu, Malaysia. *International Journal of Advanced Computer Science and Applications*, 10(11). https://doi.org/10.14569/IJACSA.2019.0101117.

Anand, R., Das, J. and Sarkar, P., 2021. Comparative Analysis of YOLOv4 and YOLOv4-tiny Techniques towards Face Mask Detection. *2021 International Conference on Computational Performance Evaluation, ComPE 2021*, pp.803–809. https://doi.org/10.1109/COMPE53109.2021.9751880.

Anon. 2023. *LoRa and LoRaWAN: Technical Overview | DEVELOPER PORTAL*. [online] Available at: <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/> [Accessed 17 April 2023].

Arcos-García, Á., Álvarez-García, J.A. and Soria-Morillo, L.M., 2018. Evaluation of Deep Neural Networks for Traffic Sign Detection Systems. *Neurocomputing*, 316, pp.332–344. https://doi.org/10.1016/J.NEUCOM.2018.08.009.

Baduge, S.K., Thilakarathna, S., Perera, J.S., Arashpour, M., Sharafi, P., Teodosio, B., Shringi, A. and Mendis, P., 2022. Artificial Intelligence and Smart Vision for Building and Construction 4.0: Machine and Deep Learning Methods and Applications. *Automation in Construction*, 141, p.104440. https://doi.org/10.1016/J.AUTCON.2022.104440.

BEHRTECH, 2023. *LPWAN Explained | Comparing MYTHINGS, LoRa, NB-IoT, Sigfox*. [online] Available at: <https://behrtech.com/lpwan-technology/> [Accessed 21 April 2023].

Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B., 2016. Simple Online and Realtime Tracking. https://doi.org/10.1109/ICIP.2016.7533003.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. [online] Available at: <https://arxiv.org/abs/2004.10934v1> [Accessed 20 April 2023].

Chamidu, S., 2020. *YOLO v4 or YOLO v5 or PP-YOLO? | Towards Data Science*. [online] Available at: <https://towardsdatascience.com/yolo-v4-or-yolo-v5-or-pp-yolo-dad8e40f7109> [Accessed 20 April 2023].

Chen, W., Lu, S., Liu, B., Li, G. and Qian, T., 2020. Detecting Citrus in Orchard Environment by using Improved YOLOv4. *Scientific Programming*, 2020, pp.1–13. https://doi.org/10.1155/2020/8859237.

Chethan Kumar, B., Punitha, R. and Mohana, 2020. YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications. *Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020*, pp.1316–1321. https://doi.org/10.1109/ICSSIT48917.2020.9214094.

Choi, R., Lee, S. and Lee, S., 2020. Reliability Improvement of LoRa with ARQ and Relay Node. *Symmetry*, 12(4), p.552. https://doi.org/10.3390/sym12040552.

Cisco, 2023. *What Is Wi-Fi? - Definition and Types*. [online] Available at: <https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html> [Accessed 21 April 2023].

Degadwala, S., Vyas, D., Chakraborty, U., Dider, A.R. and Biswas, H., 2021. Yolo-v4 Deep Learning Model for Medical Face Mask Detection. *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, pp.209–213. https://doi.org/10.1109/ICAIS50930.2021.9395857.

Diwan, T., Anirudh, G. and Tembhurne, J. V., 2023. Object Detection using YOLO: Challenges, Architectural Successors, Datasets and Applications. *Multimedia Tools and Applications*, 82(6), pp.9243–9275. https://doi.org/10.1007/s11042-022-13644-y.

Do, T., 2021. *Evolution of Yolo Algorithm and Yolov5: The State-of-the-Art Object Detection Algorithm*. Oulu University of Applied Sciences.

Dwyer, B., Nelson, J., Solawetz, J., et. al, (2022). Roboflow (Version 1.0) [Software]. Available from https://roboflow.com. computer vision.

Gaudenz, B., 2023. *YOLOv7: The Fastest Object Detection Algorithm (2023) - viso.ai*. [online] Available at: <https://viso.ai/deep-learning/yolov7-guide/> [Accessed 20 April 2023].

Girshick, R., Donahue, J., Darrell, T., Malik, J., Berkeley, U.C. and Malik, J., 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, [online] 1. https://doi.org/10.1109/CVPR.2014.81.

Inoue, M. and Owada, Y., 2017. NerveNet Architecture and Its Pilot Test in Shirahama for Resilient Social Infrastructure. *IEICE Transactions on*

Communications, E100.B(9), pp.1526–1537. https://doi.org/10.1587/transcom.2016PFI0006.

Jacob, S., 2020. *What is YOLOv5?* [online] Available at: <https://blog.roboflow.com/yolov5-improvements-and-evaluation/> [Accessed 20 April 2023].

Jebril, A., Sali, A., Ismail, A. and Rasid, M., 2018. Overcoming Limitations of LoRa Physical Layer in Image Transmission. *Sensors*, 18(10), p.3257. https://doi.org/10.3390/s18103257.

Jiang, P., Ergu, D., Liu, F., Cai, Y. and Ma, B., 2022. A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 199, pp.1066–1073. https://doi.org/10.1016/J.PROCS.2022.01.135.

Jiang, Z., Zhao, L., Li, S. and Jia, Y., 2020. Real-Time Object Detection Method based on Improved Yolov4-Tiny. *Journal of Network Intelligence*. https://doi.org/10.48550/arXiv.2011.04244.

Joseph, N., 2023. *Research*. [online] Available at: <https://roboflow.com/research> [Accessed 29 April 2023].

Kim, C., Oghaz, M., Fajtl, J., Argyriou, V. and Remagnino, P., 2019. A Comparison of Embedded Deep Learning Methods for Person Detection. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications. pp.459–465. https://doi.org/10.5220/0007386304590465.

Kucev, R., 2021. *500 GB of images for Face Mask Detection. Part 1* . [online] Kaggle. Available at: <https://www.kaggle.com/datasets/tapakah68/medical-masks-part1> [Accessed 23 April 2023].

Kulshreshtha, M., Chandra, S.S., Randhawa, P., Tsaramirsis, G., Khadidos, A. and Khadidos, A.O., 2021. OATCR: Outdoor Autonomous Trash-Collecting Robot Design using YOLOv4-Tiny. *Electronics*, 10(18), p.2292. https://doi.org/10.3390/electronics10182292.

Liang, Q., Shenoy, P. and Irwin, D., 2020. AI on the Edge: Characterizing AI-based IoT Applications Using Specialized Edge Architectures. *Proceedings - 2020 IEEE International Symposium on Workload Characterization, IISWC 2020*, pp.145–156. https://doi.org/10.1109/IISWC50251.2020.00023.

Malaysian Communications and Multimedia Commission, n.d. *Guideline on the Provision of Wireless Local Area Network (WLAN) Service* . [online] Available at: <www.skmm.gov.my> [Accessed 23 April 2023].

Martin, J., 2023. *What is a Smart Campus and the Benefits to College Students and Faculty*. [online] Available at: <https://www.coxblue.com/what-is-a-

smart-campus-and-the-benefits-to-college-students-and-faculty/> [Accessed 29 March 2023].

Masmoudi, M., Ghazzai, H., Frikha, M. and Massoud, Y., 2019. Object Detection Learning Techniques for Autonomous Vehicle Applications. *2019 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2019*. https://doi.org/10.1109/ICVES.2019.8906437.

Mateusz, P., 2021. *GitHub - MatPiech/DeepSORT-YOLOv4-TensorRT-OpenVINO: Object tracking implemented with YOLOv4, DeepSort, and TensorFlow*. [online] Available at: <https://github.com/MatPiech/DeepSORT-YOLOv4-TensorRT-OpenVINO> [Accessed 23 April 2023].

Microchip Technology, 2023. *LoRa® Technology* . [online] Available at: <https://www.microchip.com/en-us/products/wireless-connectivity/lora-technology#> [Accessed 21 April 2023].

MQTT, 2023. *MQTT - The Standard for IoT Messaging*. [online] Available at: <https://mqtt.org/> [Accessed 21 April 2023].

Nico, K., 2023. *Object Tracking in Computer Vision (Complete Guide) - viso.ai*. [online] Available at: <https://viso.ai/deep-learning/object-tracking/> [Accessed 21 April 2023].

O' Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco Hernandez, G., Krpalkova, L., Riordan, D. and Walsh, J., n.d. Deep Learning vs. Traditional Computer Vision. *Journal in Advances in Computer Vision Proceedings of the 2019 Computer Vision Conference (CVC)*, pp.128-144. https://doi.org/10.1007/978-3-030-17795-9.

Owada, Y., Inoue, M. and Ohnishi, M., 2011. Regional Wireless Network Platform for Context-Aware Services and Its Implementation. In: *2011 Tenth International Symposium on Autonomous Decentralized Systems*. IEEE. pp.539–544. https://doi.org/10.1109/ISADS.2011.76.

Parico, A.I.B. and Ahamed, T., 2021. Real Time Pear Fruit Detection and Counting using YOLOv4 Models and Deep SORT. *Sensors*, 21(14), p.4803. https://doi.org/10.3390/s21144803.

Perera, I., Senavirathna, S., Jayarathne, A., Egodawela, S., Godaliyadda, R., Ekanayake, P., Wijayakulasooriya, J., Herath, V. and Sathyaprasad, S., 2021. Vehicle Tracking based on an Improved DeepSORT Algorithm and the YOLOv4 Framework. *2021 10th International Conference on Information and Automation for Sustainability, ICIAfS 2021*, pp.305–309. https://doi.org/10.1109/ICIAFS52090.2021.9606052.

Ramasubramanian, A.K., Mathew, R., Preet, I. and Papakostas, N., 2022. Review and Application of Edge AI Solutions for Mobile Collaborative Robotic Platforms. *Procedia CIRP*, 107, pp.1083–1088. https://doi.org/10.1016/J.PROCIR.2022.05.112.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2015. You Only Look Once: Unified, Real-Time Object Detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. https://doi.org/10.48550/arXiv.1506.02640.

Rupali, R., 2022. *Difference between AI, ML and DL | Towards Data Science*. [online] Available at: <https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c> [Accessed 18 April 2023].

Sambolek, S. and Ivasic-Kos, M., 2021. Automatic Person Detection in Search and Rescue Operations using Deep CNN Detectors. *IEEE Access*, 9, pp.37905–37922. https://doi.org/10.1109/ACCESS.2021.3063681.

Saponara, S., Elhanashi, A. and Zheng, Q., 2022. Developing a Real-Time Social Distancing Detection System based on Yolov4-Tiny and Bird-Eye View for COVID-19. *Journal of Real-Time Image Processing*, [online] 19(3), pp.551–563. https://doi.org/10.1007/S11554-022-01203-5/TABLES/6.

Song, H., Liang, H., Li, H., Dai, Z. and Yun, X., n.d. Vision-Based Vehicle Detection and Counting System using Deep Learning in Highway Scenes. [online] https://doi.org/10.1186/s12544-019-0390-4.

Sun, S., Akhtar, N., Song, H., Zhang, C., Li, J. and Mian, A., 2019. Benchmark Data and Method for Real-Time People Counting in Cluttered Scenes using Depth Sensors. *IEEE Transactions on Intelligent Transportation Systems*, 20(10), pp.3599–3612. https://doi.org/10.1109/TITS.2019.2911128.

Tham, M.-L., Lean, C.H., Lim, W.-S., Leong, R.-C., Owada, Y. and Sein, M.M., 2023. Performance Study of Disaster-Resilient Mesh Networking using NerveNet Wi-Fi and LoRa. In: *2023 6th Conference on Cloud and Internet of Things (CIoT)*. IEEE. pp.219–224. https://doi.org/10.1109/CIoT57267.2023.10084890.

The Things Network, 2023. *What are LoRa and LoRaWAN?* . [online] Available at: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/> [Accessed 21 April 2023].

Vishal, R., 2021. *YOLO v4 explained in full detail | AIGuys*. [online] Available at: <https://medium.com/aiguys/yolo-v4-explained-in-full-detail-5200b77aa825> [Accessed 20 April 2023].

Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y.M., 2022. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. https://doi.org/10.48550/arXiv.2207.02696.

Wang, C.Y., Mark Liao, H.Y., Wu, Y.H., Chen, P.Y., Hsieh, J.W. and Yeh, I.H., 2020. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2020-June, pp.1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203.

Wang, J., Ma, Y., Zhang, L., Gao, R.X. and Wu, D., 2018. Deep Learning for Smart Manufacturing: Methods and Applications. *Journal of Manufacturing Systems*, 48, pp.144–156. https://doi.org/10.1016/J.JMSY.2018.01.003.

Wang, L., Zhou, K., Chu, A., Wang, G. and Wang, L., 2021. An Improved Light-Weight Traffic Sign Recognition Algorithm based on YOLOv4-Tiny. *IEEE Access*, 9, pp.124963–124971. https://doi.org/10.1109/ACCESS.2021.3109798.

Wojke, N., Bewley, A. and Paulus, D., 2017. Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645-3649. https://doi.org/ 10.1109/ICIP.2017.8296962.

Yin, W., Diao, W., Wang, P., Gao, X., Li, Y. and Sun, X., 2021. PCAN—Part-based Context Attention Network for Thermal Power Plant Detection in Remote Sensing Imagery. *Remote Sensing*, 13(7), p.1243. https://doi.org/10.3390/rs13071243.

Ying, B., Xu, Y., Zhang, S., Shi, Y. and Liu, L., 2021. Weed Detection in Images of Carrot Fields based on Improved YOLO v4. *Traitement du Signal*, 38(2), pp.341–348. https://doi.org/10.18280/ts.380211.

Zhang, H., Qin, L., Li, J., Guo, Y., Zhou, Y., Zhang, J. and Xu, Z., 2020. Real-Time Detection Method for Small Traffic Signs based on Yolov3. *IEEE Access*, 8, pp.64145–64156. https://doi.org/10.1109/ACCESS.2020.2984554.

Zhang, Y., Chen, Z. and Wei, B., 2020. A Sport Athlete Object Tracking based on Deep Sort and Yolo V4 in case of Camera Movement. *2020 IEEE 6th International Conference on Computer and Communications, ICCC 2020*, pp.1312–1316. https://doi.org/10.1109/ICCC51575.2020.9345010.

Zhu, X., Lyu, S., Wang, X. and Zhao, Q., 2021. TPH-YOLOv5: Improved YOLOv5 based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios. *Proceedings of the IEEE International Conference on Computer Vision*, 2021-October, pp.2778–2788. https://doi.org/10.1109/ICCVW54120.2021.00312.

APPENDICES

Appendix A: Maximum EIRP and the Frequency Bands.

Item	Frequency Bands	Maximum EIRP	Usage
1.	2400 MHz to 2500 MHz	500 milliWatts	Outdoor/Indoor
2.	5150 MHz to 5350 MHz	1 Watt	Indoor
3.	5470 MHz to 5650 MHz	1 Watt	Outdoor/Indoor
4.	5725 MHz to 5875 MHz	1 Watt	Outdoor/Indoor