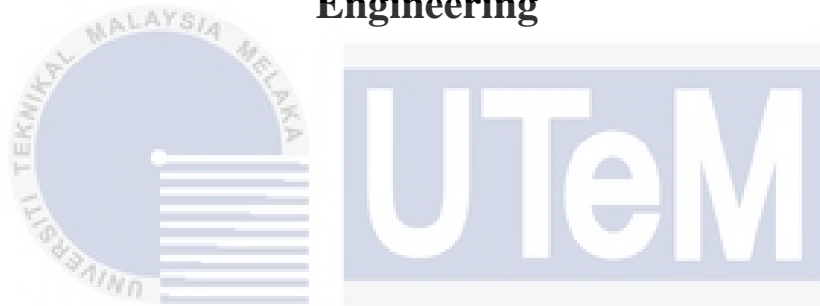




**Faculty of Electronics & Computer Technology and
Engineering**



**DEVELOPMENT OF PIPES COUNTING APPLICATION FOR
WAREHOUSE INVENTORY BASED ON ANDROID PLATFORM**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

NOR FATIN SYAHIRAH BINTI ZAIRUL AKHBAR

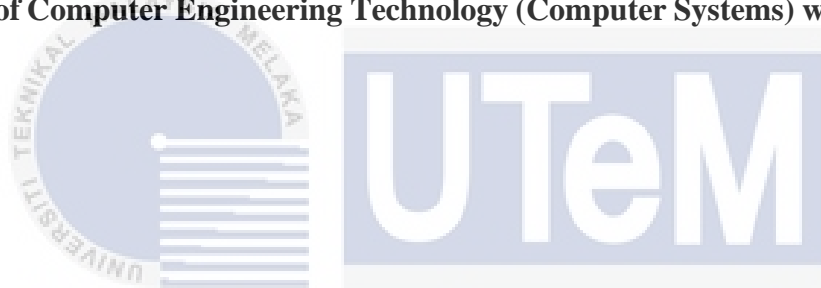
Bachelor of Computer Engineering Technology (Computer Systems) with Honours

2024

**DEVELOPMENT OF PIPES COUNTING APPLICATION FOR WAREHOUSE
INVENTORY BASED ON ANDROID PLATFORM**

NOR FATIN SYAHIRAH BINTI ZAIRUL AKHBAR

**A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Computer Engineering Technology (Computer Systems) with Honours**



Faculty of Electronics & Computer Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

**BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II**

Tajuk Projek : **DEVELOPMENT OF PIPES COUNTING APPLICATION FOR WAREHOUSE INVENTORY BASED ON ANDROID PLATFORM**

Sesi Pengajian: **2023/2024**

Saya **NOR FATIN SYAHIRAH BINTI ZAIRUL AKHBAR** mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

SULIT*

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD*

(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

TIDAK TERHAD

Disahkan oleh:



(TANDATANGAN PENULIS)

Alamat Tetap: **NO. 13 JALAN LAGONG
27/84 TAMAN ALAM MEGAH
SEKSYEN 27 40400 SHAH ALAM
SELANGOR**



(COP DAN TANDATANGAN PENYELIA)
DR. SUHAILA BINTI MOHD NAJIB
PENSYARAH KANAN
JABATAN TEKNOLOGI KEJURUTERAAN
FAKULTI TEKNOLOGI KEJURUTERAAN
ELEKTRONIK DAN KOMPUTER
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Tarikh: 9/2/2024

Tarikh: 12/2/2024

DECLARATION

I declare that this project report entitled “Development of Pipes Counting Application for Warehouse Inventory Based on Android Platform” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : 

Student Name : NOR FATIN SYAHIRAH BINTI ZAIRUL AKHBAR

Date : 9/2/2024



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Engineering Technology (Computer Systems) with Honours.

Signature :

Supervisor Name :

DR. SUHALA BINTI MOHD NAJIB

Date :

12/2/2024

Signature :

Co-Supervisor :

Name (if any)

Date :

DEDICATION

To my beloved mother, Nor Zakiah Binti Zakaria, and father, Zairul Akhbar bin Jahroni, who always supported me through this completion of this research. Thank you for the given advice, financial and moral support to motivate me to conduct this research until now. Thanks also to my Supervisor, Family, and my friends for always supporting and helping me in this research.



ABSTRACT

The project aims at the development of a pipes counting application designed for warehouse inventory based on the Android platform. This study focuses on pipes, which are crucial elements in construction materials at construction sites and employs deep learning technology to automate the pipe counting process. The design of a mobile application is a part of the development of pipe counting applications which enables warehouse workers to effectively count and manage the stock of pipes in its inventory. Inaccuracy of inventory records appears to be a widespread issue, particularly in industrial and residential sectors. While manual counting is a widespread practice at the warehouse, it is impractical, challenging, and time-consuming. The process of counting the pipes manually often takes several hours for workers. Thus, human error can take place due to the manual counting process. To address the challenge, a pre-trained YOLOv5 model based on the PyTorch framework is used to prepare the data model training. The REST API server with the framework of Flask to expose the data model training has been integrated into the Flutter framework deployed on mobile applications for testing. The application prompted users to input the image of a captured pipe. The response will be processed from the YOLOv5 REST API in the Android app to return the detected objects and their coordinates. The detected objects and their counts will be displayed on the Android app's user interface.

ABSTRAK

Projek ini bertujuan untuk membangunkan aplikasi mengira paip yang direka untuk inventori gudang berdasarkan platform Android. Kajian ini memberi tumpuan kepada paip, yang merupakan elemen penting dalam bahan binaan di tapak pembinaan, dan menggunakan teknologi pembelajaran mendalam untuk mengautomasikan proses pengiraan paip. Reka bentuk aplikasi mudah alih adalah sebahagian daripada pembangunan aplikasi mengira paip yang membolehkan pekerja gudang mengira dan mengurus stok paip dalam inventornya dengan berkesan. Ketidaktepatan rekod inventori nampaknya menjadi isu yang meluas, terutamanya dalam sektor perindustrian dan kediaman. Walaupun pengiraan manual adalah amalan yang meluas di gudang, ia tidak praktikal, mencabar dan memakan masa. Proses mengira paip secara manual selalunya mengambil masa beberapa jam untuk pekerja. Oleh itu, kesilapan manusia boleh berlaku kerana proses pengiraan manual. Untuk menangani cabaran tersebut, model YOLOv5 yang telah dilatih berdasarkan rangka kerja PyTorch digunakan untuk menyediakan latihan model data. RESTful API dengan rangka kerja Flask untuk mendedahkan latihan model data telah disepadukan ke dalam rangka kerja Flutter yang digunakan pada aplikasi mudah alih untuk ujian. Aplikasi ini membenarkan pengguna untuk memuat naik imej paip yang ditangkap. Respons akan diproses daripada YOLOv5 REST API dalam apl Android untuk mengembalikan objek yang dikesan dan koordinatnya. Objek yang dikesan dan kiraan akan dipaparkan pada antara muka pengguna Android.

ACKNOWLEDGEMENTS

In the name of Allah, Most Beneficent and Most Merciful, Praise to Allah S.W.T for providing me with great health, strength, and emotional support in completing this project for the title of “Development of Pipes Counting Application for Warehouse Inventory based on Android Platform”. First and foremost, I would like to express my gratitude to my supervisor, Dr. Suhaila Binti Mohd Najib for her precious guidance, words of wisdom and patience throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for the support through providing the resources and environment essential which enables me to accomplish the project. Not forgetting my fellow colleague, Syafiq for the willingness of sharing his thoughts and ideas regarding the project.

My highest appreciation goes to my parents, and family members for their love and prayer during the period of my study. An honorable mention also goes to my course mate for all the motivation and understanding.

Finally, I would like to thank all the staff at the UTeM, fellow colleagues and classmates, the faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

	PAGE
DECLARATION	
APPROVAL	
DEDICATIONS	
ABSTRACT	i
ABSTRAK	ii
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
LIST OF ABBREVIATIONS	ix
CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Pipe Detection and Counting Based on Safety Hazard Issue	2
1.3 Problem Statement	3
1.4 Project Objective	4
1.5 Scope of Project	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Introduction	5
2.2 Overview of Image Processing Technique for Object Detection and Image Counting	6
2.3 Overview of Neural Network for Object Detection and Image Counting	7
2.3.1 Convolutional Neural Network (CNNs)	8
2.3.2 You Only Look Once (YOLO)	9
2.4 Previous work of object detection and counting for multipurposed industry	11
2.5 Review of the Comparison for Object Detection and Image Counting Table	20
2.6 Summary	20
CHAPTER 3 METHODOLOGY	21
3.1 Introduction	21
3.2 Methodology	21
3.3 Python backend	26
3.3.1 Process flow for training	26
3.3.2 Process flowchart for testing	28
3.3.3 Round-shaped pipe detection	29

3.3.4	Roboflow software for annotating image dataset	30
3.3.5	Generate dataset in Roboflow software	31
3.3.6	Model Training Procedure using YOLOv5 PyTorch	32
3.4	Python backend	33
3.4.1	Flask Server	33
3.4.2	Integrating YOLOv5 PyTorch Model with Flutter through Flask	33
3.5	Firebase Services	36
3.6	Hardware and Software Development Tools	37
3.7	Libraries and framework setup	40
3.8	Limitation of Proposed Methodology	42
3.9	Project Planning	43
3.10	Summary	45
CHAPTER 4	RESULTS AND DISCUSSIONS	46
4.1	Introduction	46
4.2	System Operation and Testing	47
4.2.1	User Interface Design	47
4.3	System performance	50
4.4	Summary	55
CHAPTER 5	CONCLUSION AND RECOMMENDATIONS	56
5.1	Conclusion	56
5.2	Potential for commercialization	57
5.3	Future Works	58
REFERENCES		59

اوتیور سیتی تیکنیکل ملیسیا ملاک

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1	The comparison of the object detection and image counting in multipurposed industry	11
Table 3.1	Major tasks (use case) and possible actors of the system	25
Table 3.2	Gantt Chart Process for Final Year Project I	43
Table 3.4	Gantt Chart Process for Final Year Project II	44
Table 4.1	System performance testing in pipe counting application	50



LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 3.1	System Architecture Diagram	22
Figure 3.2	Use Case Diagram of the System	24
Figure 3.3	Training model process flow	26
Figure 3.4	Process flowchart for testing	28
Figure 3.5	Round-shaped pipes annotation for preparing datas	30
Figure 3.6	Generate dataset pipe counting in Roboflow.	31
Figure 3.7	Python script model train.py.	32
Figure 3.8	Weights folder	32
Figure 3.9	Load the pre-trained model with specified weights and processing the image	34
Figure 3.10	Body function to allow input	34
Figure 3.11	Ngrok access	35
Figure 3.12	Send HTTP post request to Flask server	35
Figure 3.12	Firebase name project	36
Figure 3.13	Firebase name project	36
Figure 3.14	PyCharm Software Edition 2023.1.2	38
Figure 3.15	Google Firebase cloud	39
Figure 3.16	Visual Studio Code	39
Figure 3.17	Roboflow	40
Figure 3.18	Ngrok	41
Figure 4.1	Mobile application named 'Pipe Quant' on edge device	47
Figure 4.2	Pipes detect and count	48
Figure 4.3	Historical Data Review	49

LIST OF ABBREVIATIONS

<i>CNNs</i>	-	Convolutional Neural Network
<i>YOLO</i>	-	You Only Look Once
<i>API</i>	-	Application Programming Interface
<i>mAP</i>	-	mean average precision
<i>NNs</i>	-	Neural Network



CHAPTER 1

INTRODUCTION

1.1 Background

A warehouse is a storage area where goods are received, stocked, and dispatched [1]. Warehousing is the process of organizing, storing, and managing inventory to ensure that everything operates efficiently within the warehouse. In addition, the main tasks and duties of a warehouse are to keep the right products in stock, manage new inventory coming into the facility, pack and ship orders, track and improve storage overall and arrange the delivery of finished goods to end customers. This proves that warehousing is the crucial component of the supply chain. It is not only necessary in the supply chain, but it is also significant for logistic operations and makes a major difference to supply chain speed and cost [1]. With the rapid advancements in computer vision, artificial intelligence and deep learning techniques have found their way in providing the improvement of digital warehouse to streamline and automate warehouse operations. This can improve speed, accuracy, and visibility throughout the supply chain process flow. Warehouse computer vision systems gather visual data from the warehouse environment using cameras, sensors, and other imaging equipment. These devices provide data that is used as an input for future analysis [2]. Other than that, the warehouse computer vision relies heavily on object detection. Convolutional Neural Networks (CNNs) are used to detect and locate objects in the captured images by being trained on large datasets to recognize patterns and features correspond to specific objects. This enables the algorithms to accurately identify objects in real-time.

This study focuses on pipes, which are crucial elements in construction materials at construction sites and employs deep learning technology to automate pipe counting process. The design of a mobile application is a part of the development of pipe counting applications which enables warehouse workers to effectively count and keep track of inventory stock in their inventory based on timestamp. In general, the creation of an Android-based application using deep learning technique for pipe counting in warehouse inventory seeks to improve accuracy, streamline inventory management, and boost operational effectiveness.

1.2 Pipe Detection and Counting Based on Safety Hazard Issue

In the industrial sector, worker safety is of the greatest significance. Potential risks can be discovered by pipe detection and counting, which enables the implementation of the necessary safety measures. One such incident in California where a packer in a manufacturing facility dies when he is struck by a pipe that rolled off a storage rack highlights the implementation of technologies to prevent such tragedies [3]. The primary feature of pipe detection and counting is the capability to automate counting and inventory management of pipe inventories. The danger of accidents can be particularly reduced by ensuring the storage racks are not overcrowded and the pipes are appropriately structured. Automated systems help to keep the environment structured and clear of hazards and provide a safer workplace for manufacturing workers.

Additionally, pipe detection and counting increases employees' knowledge of the existence of pipes and the related safety risks. Therefore, the staff may be briefed on safety practices to follow when working near or with handling pipes and informed of the hazards associated. In conjunction, training programs with technical solutions form a synergistic approach to risk mitigation and assuring the well-being of industrial personnel. This

understanding encourages a safety culture beneath the company and decreases accidents. This leads us to the conclusion that pipe detection and counting are crucial in reducing injury to create a more secure workplace.

1.3 Problem Statement

One of the features of having a good inventory management system is to have a high accuracy in which the difference of variances between the physical and inventory are small. Therefore, inaccuracy of inventory record appears to be a widespread issue particularly in industrial and residential sectors [4]. Inaccurate inventory data is a result of incorrect product counts. Inaccurate inventory information leads to ineffective replenishment choices, which also raises inventory costs and lowers service levels. This is a significant problem influencing the effectiveness of the supply chain performance in manufacturing, distribution, and retail environments.

The warehouse may contain thousands or millions of pipes. While manual counting is a widespread practice at the warehouse, it is impractical, challenging, and time-consuming. One survey from PT. Barata Indonesia (Persero) Turbine Components Division of the manufacturing companies in Indonesia proves that inventory of goods that is done manually can be said to be less efficient and effective. The process of counting the stock manually often takes several hours for workers [5]. Thus, human error can take place due to the manual counting process and causes losses to the company.

The input data of stocks will be updated into the inventory system. Paper sheets for data stock counting are viewed as the characteristic of traditional warehouse which leads to several drawbacks including lacks real-time updates. The time lag might be quite significant between the actual stock and the representation of the inventory records since

the data must be manually input and processed. This labor-intensive operation may generate delays which can cause discrepancies of quantity due to lack of real-time updates.

To overcome these issues, pipe detection and counting based on computer vision technique on the Android platform is introduced. The real-time interface from the mobile application ensures that inventory records are instantly synchronized with the database. Furthermore, utilizing computer vision to address the challenge of material counting holds significant importance in terms of cost savings and enhanced work efficiency.

1.4 Project Objective

The main aim of this project is to develop a mobile application that facilitates the counting and management of pipes in a warehouse inventory. Specifically, the objectives are as follows:

- a) To detect and count pipe using pre-trained model based on YOLOv5.
- b) To provide a real time interface based on Android application to automate the warehouse stock pipe counting.

1.5 Scope of Project

Here is the description of the project:

- a) The system limitation is that the dataset is only trained based on round shaped pipe.
- b) The dataset of the pipe is in one view perspective to increase the possibility of pipe detection.
- c) The prepared data model for training used is YOLOv5 and system framework used is PyTorch. For testing data model and framework used Python as backend for processing output from the YOLOv5 model to integrate into Flutter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Most inventory warehouses management following a manual process uses headcount, resulting in inaccurate data due to missed inventory items during the manual count. Instead of the fully manual operation, the company's productivity and accessibility to inventory information did the study of object counters using inventory management. It is labelled as time-consuming and less efficient. With the advancement of current technologies, pipe detection and counting provides precise and effective solutions. The solutions to these problems have been offered in a variety of ways over the years, with recent developments in computer vision technique and machine learning leading to considerable breakthroughs. The automatic extraction of data from images is known as computer vision. Anything from 3D models, camera placement, object recognition, and picture content search may be used [6]. It also uses massive image algorithms to reduce the need for large datasets, which has resulted in significant performance improvements. The most important step in inventory control is object counting, which determines how readily available stock items are. It compromises the ability to locate instances of predefined stock templates regardless of scale, rotation, or partial occlusion using a vision interface.

2.2 Overview of Image Processing Technique for Object Detection and Image Counting

Image processing is a technique to convert a physical image into digital form and applying multiple processes to produce an improved image or extract some relevant information [7]. This technique was proposed as a form of signal distribution where the input is an image, such as a video frame or a photograph, and the output can either be another image or its related properties.

There are several research that use image processing techniques to detect object and image counting. Ramil Lumauag [8] conducted a method to capture images of fish population by using blob analysis and Euclidean filtering. The computation matrix is chosen to record the detection and counting accuracy. A blob is a group of adjacent pixels with the same logical state. Blob analysis is a technique to find blobs in images in taking specific measurements of these blobs. The objects' distance travelled was calculated using the centroid in accordance with the Euclidean distance formula. The pixel positions of the moving object at initial stage to the final stage were utilized as variables. The result shows that the average counting accuracy of ten test images is 91% and detection accuracy is 94%.

Another research from Jianjun Ni et al. [9] which proposed an algorithm to detect and count overlapping objects with circular shape by using Circular Hough Transform. The proposed approach carried out on a test image that is circular and non-circular objects which is overlapped with each other. In the proposed approach, the contour detection technique is used to count objects irrespective of its shapes. Experimental result shows the counting task in the image has been successfully detected, segmented, counted, and sorted efficiently.

Kaushiki Roy et al. [10] discussed an automated segmentation and counting of platelets Android based mobile app by using image processing technique. The characteristics of the platelets are that they are in irregular shapes and some of them are very small, and some are very large. Additionally, the color-based segmentation technique is used to segment platelets. Next, the dataset images were integrated to the Android application to be used as the input. Experimental results show that the accuracy obtained by doing the comparison between manual count and count given by the app reaches 98.8%.

2.3 Overview of Neural Network for Object Detection and Image Counting

In recent years with the update and iteration of high-performance GPUs, object detection algorithms based on numerous large-scale datasets such as COCO and Neural Network (NNs) based on CNNs have become the norm in current object detection technology. The NNs model is widely used to address a real-world issue in business, education, economics, and variety aspects of life areas. This deep learning based performed an excellent rate for object detection.

This technique can be categorized into one-stage and two-stage algorithms. The one-stage algorithms are said to be more concise since it does not required to generate a region proposal [11] while the two-stage algorithm focuses on the complex architecture. The one- stage method involves YOLO and SSD algorithm meanwhile in two-stage algorithms include Faster R-CNN.

Meanwhile, the typical approach for supervised learning includes two stages: network training and network evaluation. During the training period, the network processes images and ground truths, such as density maps or the total headcount, derives gradients, and then backpropagates the loss. The objective function that the network is minimizing is

known as the loss function. A metric function is used to evaluate the model after one or a few epochs.

2.3.1 Convolutional Neural Network (CNNs)

Throughout the years, the CNNs has become one of the most popular models for computer vision. This common depth of NNs extended via shared weights. It is divided into three primary types of layers which are convolutional, pooling and fully connected layer. CNNs are a particular kind of neural network that are proficient in modelling and identifying spatial association between adjacent pixels in an image. The visual domain is where CNNs clearly outperform feed-forward neural networks. Research has found that CNNs execute an excellent performance in machine learning tasks. They are good at capturing the spatial and temporal dependencies of a picture, unlike the other neural networks. This is achieved by utilizing the appropriate filters. The CNNs design enables a better fitting of the picture dataset due to its limited fitting of the image dataset [8].

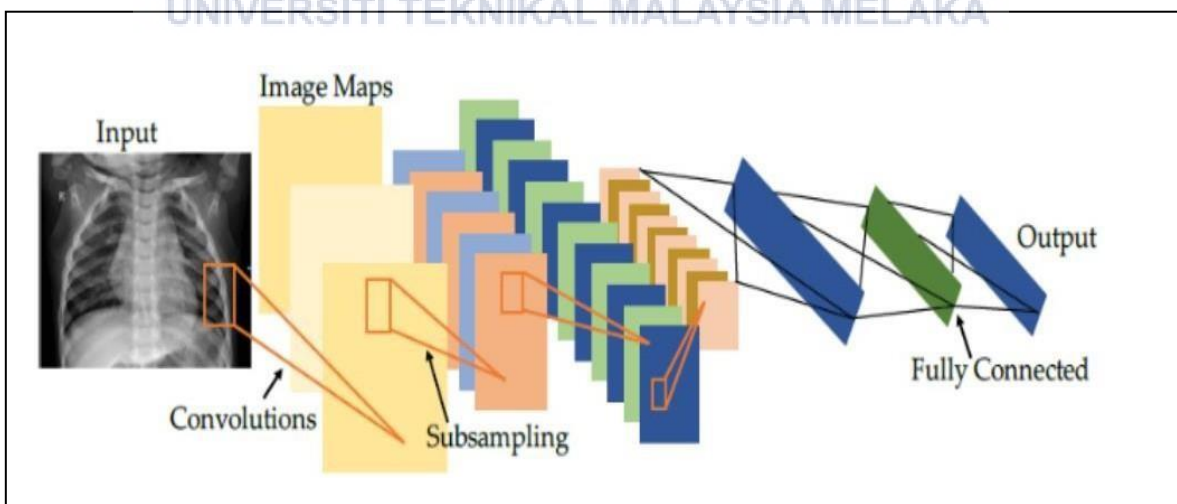


Figure 2.1 Architectural overview of CNN [12]

2.3.2 You Only Look Once (YOLO)

The YOLO series is the enhancement of R-CNN. The algorithm may quickly identify objects by referring to CNNs since it uses a single convolutional network to predict bounding boxes with class probability [13]. In addition, YOLO enhances detection performance by training on entire images. Over the years, YOLO has developed different versions such as YOLOv3, YOLOv4 and YOLOv5.

Wen-Sheng Wu and Zhe-Ming Lu [14] introduced to implement YOLOv3 network to detect and counting cups on warehouse shelves to update into inventory system. The improvement has been made to the network such as resetting a detection area, discard supporting frameworks and feature maps that irrelevant to any final detection outcomes. The result shows that detection outcome in mean average precision (mAP) is 96.82%.

J. Dela Cruz [15] proposed YOLOv5 method for object detection and inventory stock counting. The total of 1623 images have been collected and split in Roboflow application into three parts based on training set (70%), validate set (20%) and testing set (10%). The outcome shows that the subject in image which to detect cell phone boxes has reached an average accuracy of 0.93. The proposed method has been proven to identify and calculate the target objects and update inventory stock counting.

A.Cogato et al. [16] evaluated different versions of YOLO which consists of YOLOv3, YOLOv4 and YOLOv5 for real-time bunch detection and grape bunch counting. The training for YOLOv3 and YOLOv4 is based on Darknet framework, while YOLOv5 based on PyTorch framework. The dataset used for the model training was composed of 2931 images, which the dataset divided into two sets of data which is train and test data. The collected dataset images were annotated to label and drawing bounding boxes on each bunch in the image. These versions of YOLO model were contrasted and trained for various tasks. The fastest detection was achieved by YOLOv4 while best mAp obtained

by YOLOv5. Figure 2.2 shows the comparison of performance between different versions of YOLO based on its accuracy and speed of detection.

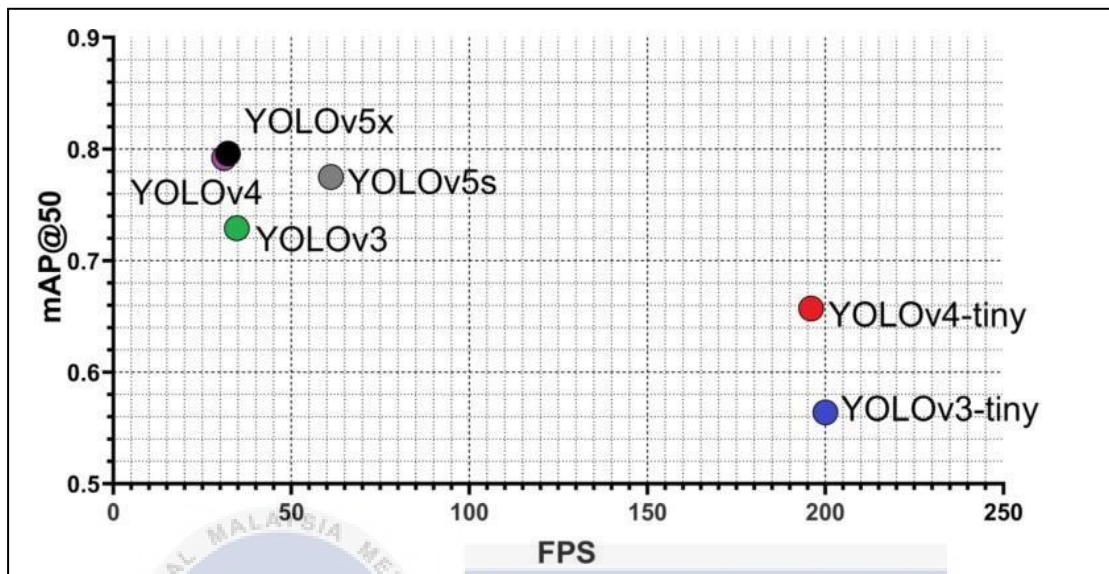
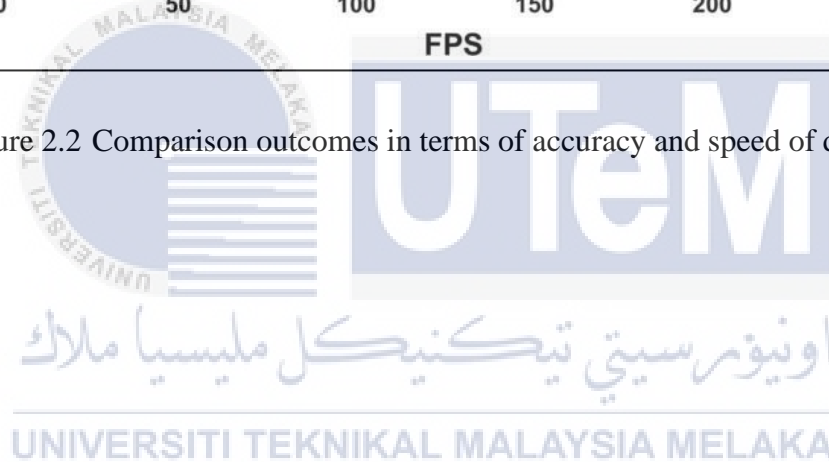


Figure 2.2 Comparison outcomes in terms of accuracy and speed of detection [16]



2.4 Previous work of object detection and counting for multipurposed industry

Table 2.1 The comparison of the object detection and image counting in multipurposed industry

Year	Author	Title	Methodology	Discussion
2019	Krishna Sai, B. N. Sasikala, T.	Object Detection and Count of Objects in Image using Tensor Flow Object Detection API	<ol style="list-style-type: none"> 1. Faster RCNN Inception B2 model <ul style="list-style-type: none"> • To train images with multiple CNNs for all region of an image. 2. Csv file (binary format) <ul style="list-style-type: none"> • Store data 3. Label map <ul style="list-style-type: none"> • Gives unique id. 4. TensorFlow <ul style="list-style-type: none"> • 2D coordinate system and a 3D coordinate system that are both being used. 	<ul style="list-style-type: none"> • Tensor flow Object Detection API was used to train the model, and the Faster R-CNN technique was employed for implementation, making it easier to identify potentially dangerous objects.

2022	<p>Fanca, Alexandra Pop Puscasiu, Adela Gota, Dan Ioan Giurgiu, Flavius Madalin Santa, Maria Magdalena Valean, Honoriu Domuta, Claudiu Miclea, Liviu</p>	<p>Romanian Coins Recognition and Sum Counting System from Image Using TensorFlow and Keras</p>	<ol style="list-style-type: none"> 1. "google_images_download" library. <ul style="list-style-type: none"> • To analyze the accuracy of new images and to evaluate the model. 2. TensorFlow <ul style="list-style-type: none"> • Open-source library 3. Keras <ul style="list-style-type: none"> • API gives easy access to machine learning platform, TensorFlow. 4. CNN <ul style="list-style-type: none"> • Dataset to train the model for coin image recognition according to its accuracy. • For coin value prediction 5. "Canny edge detector" <ul style="list-style-type: none"> • Algorithm to identify each coin in the image 	<ul style="list-style-type: none"> • The paper highlights the importance of feature extraction and classification methods using (CNNs) and picture pre-processing methods like image enhancement and segmentation. • The paper summarizes the usage of deep learning algorithms for coin recognition and counting tasks. • The total sum of the coins is then determined using the categories into which they are categorized. The findings following testing are in line with what is expected from a CNN-based application where the database was manually constructed.
------	--	---	--	--

2021	Lu, Shenglian Song, Zhen Chen, Wenkang Qian, Tingting Zhang, Yingyu Chen, Ming Li, Guo	Counting dense leaves under natural environments via an improved deep-learning-based object detection algorithm	<ol style="list-style-type: none"> 1. Detectron2 deep-learning platform based on programming language Python. <ul style="list-style-type: none"> • Performed on an Intel Core i7-9850H CPU and a Quadro P4000 GPU, with CUDA 10.1 and CUDNN 7.6.5. 2. Labelling 3. Data Augmentation 4. Dataset 5. Improved CenterNet <ul style="list-style-type: none"> • Training frameworks 	The enhanced CenterNet detector suggested in this study successfully identified leaves in a variety of development stages, lighting environments, and sizes with the accuracy of 96%.
2021	Hong, Suk Ju Nam, Il Kim, Sang Yeon Kim, Eungchan Lee, Chang Hyup Ahn, Sebeom Park, Il Kwon	Automatic pest counting from pheromone trap images using deep learning object detectors for <i>Matsucoccus thunbergianae</i> monitoring	<ol style="list-style-type: none"> 1. Object detection models <ul style="list-style-type: none"> • Faster R-CNN • RetinaNet • SSD 2. Image cropping: original images cropped into smaller images by size 6x4 and 12x8. 3. Model evaluation; gives number of false-positive and false-negative 	<ul style="list-style-type: none"> • Faster R-CNN and RetinaNet outperformed the other eight deep learning-based object detection models that were examined. • The cropping size and the chosen object identification model have an impact on the models' performance.

	Kim, Ghiseok		detections	
2020	Lanang Afkaar Ar, Muhammad Muzakki Adytia S, Sulthan Nugraha, Yudhistira Rizka R, Farizah Ernesto, Andy Intan Kanggrawan, Juan Suherman, Alex L.	A computer vision-based object detection and counting for COVID-19 protocol compliance: A case study of Jakarta	<ul style="list-style-type: none"> • Python programming language and artificial intelligence technology • Physical distance and mask detection are included in the system. • Object counting method. • 5th gen i5 processor NVIDIA graphics card with minimum calculation capability of 6.0 effect less time in take. 	The implementation of artificial intelligence technology through CCTV develops monitoring system and surveillance.

2021	Gupta, Pooja Sharma, Varsha Varma, Sunita	People detection and counting using YOLOv3 and SSD models	<ol style="list-style-type: none"> 1. YOLOv3 2. SSD 3. Collection of image and video datasets to compare the performance of the two algorithms. 	<ul style="list-style-type: none"> • Object counting is a crucial task in industries. • CNN increased the precision of object recognition and counting. • The SSD approach according to the authors, may be used to both object identification and counting tasks.
2021	Kumar, Shailender Vishal Sharma, Pranav Pal, Nitin	Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow	<ol style="list-style-type: none"> 1. Object detection <ol style="list-style-type: none"> a) The YOLOv4 2. Dataset <ol style="list-style-type: none"> b) COCO 3. Object tracking <ol style="list-style-type: none"> c) The DeepSORT algorithm 4. Kalman Filters – motion prediction 	Object detection has potential applications in various domains such as reduce heavy traffic. It uses the Kalman Filters and SORT algorithm for object tracking and evaluation using MOT-A score.

2021	Knura, Martin Kluger, Florian Zahtila, Moris Schiewe, Jochen Rosenhahn, Bodo Burghardt, Dirk	Using object detection on social media images for urban bicycle infrastructure planning: A case study of dresden	<ol style="list-style-type: none"> 1. Images from social media posts 2. Dataset – YFCC100m 3. CNN and COCO dataset – object detection algorithm 4. Moving bicycle 	d) New method detected
2022	Nguyen, Quoc Toan	Detrimental Starfish Detection on Embedded System: A Case Study of YOLOv5 Deep Learning Algorithm and TensorFlow Lite framework	<ol style="list-style-type: none"> 1. YOLOv5 2. TensorFlow Lite 	The result shows that a deep CNN can be employed to complete the task and produce promising outcomes. However, there is issue on limitation on embedded system setup.
2022	Babila, Isaiah Francis E. Villasor, Shawn Anthonie E. Dela Cruz,	Object Detection for Inventory Stock Counting Using YOLOv5	1. YOLOv5	The YOLOv5 model has trained the model and applied successfully to detect and count the objects. The average accuracy is up to 0.96 for detection and 0.93 for counting.

	Jennifer C.			
2021	Shin, Yoonsoo Heo, Sekojae Han, Sehee Kim, Junhee Na, Seunguk	An image-based steel rebar size estimation and counting method using a convolutional neural network combined with homography	<ol style="list-style-type: none"> 1. CNN architecture 2. Training dataset 3. Segmentation image 4. Homography image processing 5. Histogram and Gaussian distribution for analysis 	Although the performance of the approach suggested in this study is adequate, it is recommended that the error rate in estimating the size and counting the quantity of steel rebars be reduced for use in complex real-world building sites.
2018	Lumauag, Ramil Nava, Marevic	Fish Tracking and counting using image processing	<ol style="list-style-type: none"> 1. Blob analysis 2. Euclidean filtering 	Certain objects have been discovered as false detections that result in excess counting which leads to overcounting. The average counting reaches 91% meanwhile detection accuracy is 94%.
2016	Ni, Jianjun Khan, Zubair Wang, Shihao	Automatic detection and counting of circular shaped overlapped	<ol style="list-style-type: none"> 1. Circular Hough Transform 2. Contour detection 3. OpenCV library 	All items in the image are identified, designated by number, and sorted successfully in the presented technique. All

	Wang, Kang Haider, Syed Kamran	objects using circular hough transform and contour detection		random shaped objects that attached to circular shaped objects can be solved by using segmentation technique.
2016	Roy, Kaushiki Dey, Ratnadeep Bhattacharjee, Debotosh Nasipuri, Mita Ghosh, Pramit	A smart phone-based app for automated segmentation and counting of platelets	<ol style="list-style-type: none"> 1. Image segmentation technique 2. Android Software Development Kit (SDK) 	It is helpful to do platelet counting by using software to get rid of results from manual examination. Additionally, it would delay the rush to the laboratory, saving time and thus minimizing mistake.
2022	A Real-Time Cup-Detection Method Based on YOLOv3 for Inventory Management	A Real-Time Cup-Detection Method Based on YOLOv3 for Inventory Management	<ol style="list-style-type: none"> 1. YOLOv3 	The purposed method has been discovered to solve the issue of manual counting. The counting work can be easily completed by a worker. The result shows the accuracy reaches until 96.82% which proves that this can be implemented in counting inventory
2022	Sozzi, Marco Cantalamessa, Silvia Cogato, Alessia Kayad, Ahmed Marinello,	Automatic Bunch Detection in White Grape Varieties Using YOLOv3, YOLOv4, and YOLOv5 Deep Learning Algorithms	<ol style="list-style-type: none"> 1. YOLOv3 2. YOLOv4 3. YOLOv5 	The comparison of three different versions of YOLO has been made to compare the accuracy and speed detection. The validation result shows that the better performance is reached by YOLOv5 and YOLOv4 compared to YOLOv3, The

	Francesco			YOLOv5 shows the best in detect bunches accurately and YOLOv4 reaches good results but with a lower detection speed.
--	-----------	--	--	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.5 Review of the Comparison for Object Detection and Image Counting Table

Table 2.1 summarized data from 15 previous works on object detection and images counting based on various techniques. The YOLOv5 achieved a good performance and the most accurate neural network based. The proposed method is getting approved by J. Dela Cruz [15] where the YOLOv5 has achieved its result to detect and count cell phones into inventory stock counting. Based on the results, it also indicates that the neural network provides real-time updates to do stock counting directly to the database. It was agreed that YOLOv5 achieved better accuracy by A. Cogato et al. [16] where the comparison of different YOLO version is compared.

2.6 Summary

Any object can be detected through devices like mobile phones. This includes computer vision techniques to be implemented into the system. There are various object detection and image counting that can be applied to build the system. The YOLO version is compared to observe the accuracy rate for the better operation system. For this project, the YOLOv5 model is chosen based on its higher accuracy compared to YOLOv3 and YOLOV4.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The field of computer vision encompasses techniques for capturing, pre-processing, analyzing, and understanding images. The goal of this field's advancement has been to electronically see and comprehend images to duplicate the abilities of human vision. By comprehending computer vision and machine learning, the techniques section outlines the specific algorithms, resources, and datasets utilized to handle the problem.

Throughout this part will discuss on creating the system relates to object detection and counting with the aids and resources of framework system requirements. The framework will extract the description of each component and basic understanding of the technique used in each component. The research intends to develop an object detection and counting system using neural network.

3.2 Methodology

The idea of using image capture to count the number of objects has been proposed as a new way of detection and counting approach by using camera as the sensor equipment [17]. The camera allows the user to insert other kinds of input, as such an image file has its own format to count the number of objects. Therefore, an image processing technique can fulfil the requirement thus the users can reduce the cost of investment since it does not require users to install much equipment.

The developed methodology in this case has four main steps. Data collection, model construction, and model training are the subsequent stages. A brief explanation of the various libraries and tools utilized in this project is provided along with the technique. In this part, the most significant procedures among the employed tools and technologies are covered. The YOLOv5 train model is proposed for object detection of the pipes based on its circular shape at the back end. The dataset of the images will be annotated by using Roboflow as an annotation tool for computer vision datasets.

The system architecture diagram illustrated in Figure 3.1 is a crucial component in the methodology of developing the system.

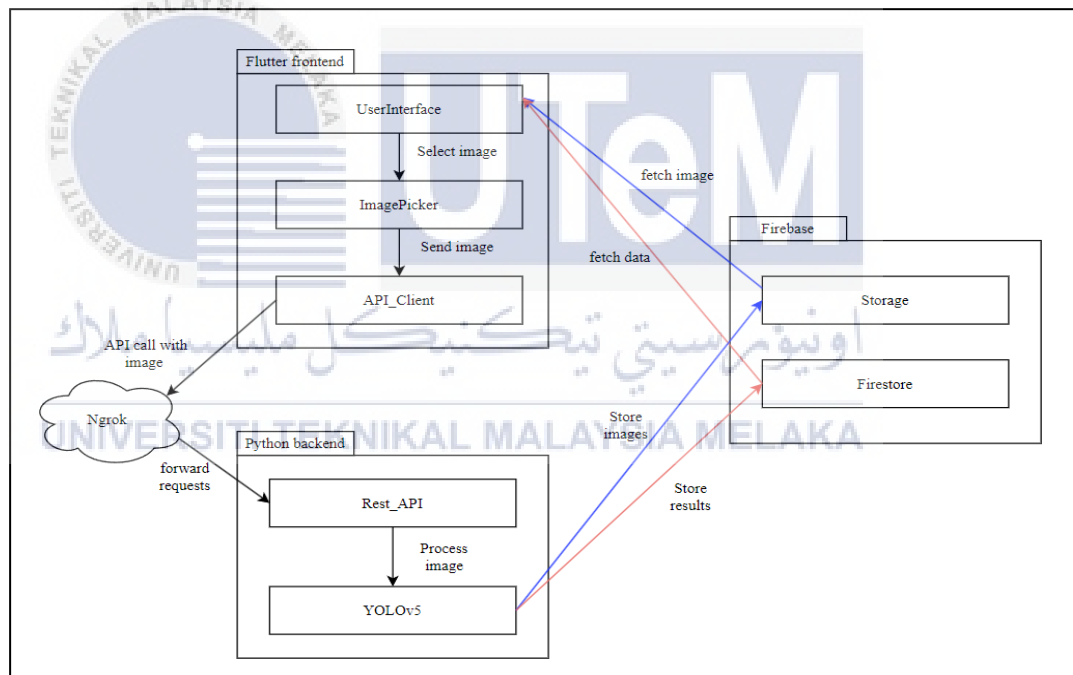


Figure 3.1 System Architecture Diagram

Below are the detailed explanations of how each component interact with each other:

- 1) *Flutter frontend*: The user interface of the mobile application is developed with the Flutter framework. Flutter is an open-source SDK for that develops high-

- performance. It is a reliable mobile application for operating systems such as Android.
- 2) *UserInterface*: The users communicate with the system of mobile application within this component. It encompasses all displays, buttons, and inputs that the user sees and interacts with.
 - 3) *ImagePicker*: A feature in the app that requires users to select images from device's gallery.
 - 4) *API_Client*: This component manages sending requests to and handling replies from backend server by dealing with network communication, send data (images) and receive processing results.
 - 5) *Ngrok*: Ngrok is a free and open-source cloud service to expose local servers to the public internet over secure tunnels. It enables public IP communication with a web server.
 - 6) *Python backend*: These components perform image processing tasks by receiving requests from the Flutter frontend, integrates and replies with the information.
 - 7) *REST_API*: It specifies endpoints that accept HTTP requests (upload images) and transmit HTTP responses (processing results). REST APIs are often used for its scalability and simplicity.
 - 8) *YOLOv5*: The pre-trained model for image processing. The pre-trained deep learning model will process the image for object detection and image counting.
 - 9) *Firebase*: A Google established framework for mobile and online applications. It offers a range of services, including storage options, real-time databases, and authentication. Firebase serves as the system's backend-as-a-service, managing data storage and server-side functionality.

- 10) *Firestore*: A modular and scalable database for server, web, and mobile applications. Firestore is a system component that stores processed results and other data. It provides real-time synchronization and quick data retrieval, which is useful for applications that require users to see up-to-date information.
- 11) *Storage*: To store processed and uploaded images by the YOLOv5 model.

Next, the use case diagram in Figure 3.2 illustrates the functional requirements of the system. It plays an integral role in understanding and documenting the actions or services in which the system should carry out from an end-user perspective.

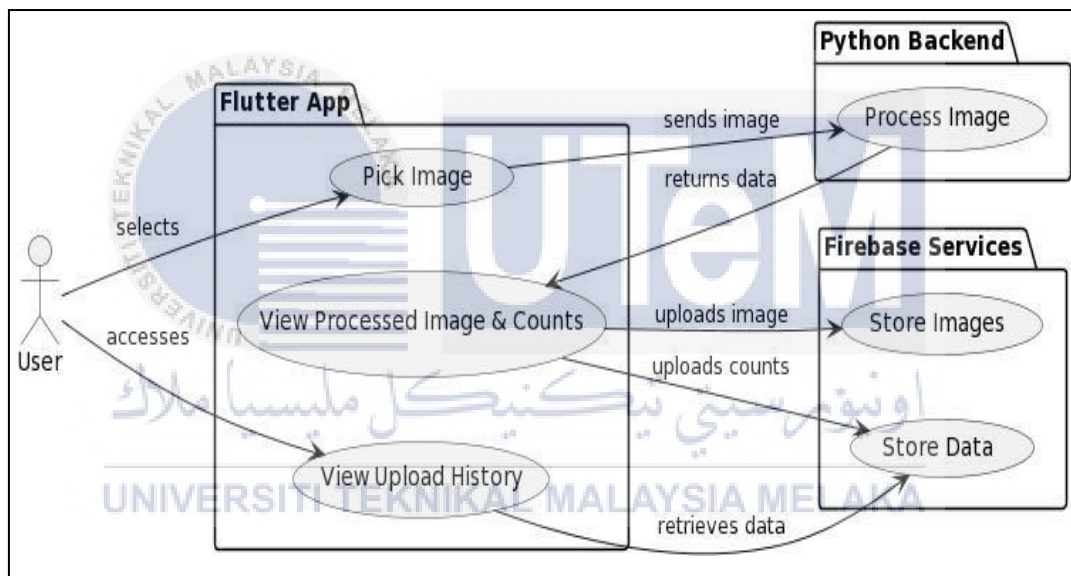


Figure 3.2 Use Case Diagram of the System

Table 3.1 Major tasks (use case) and possible actors of the system

No.	Task	Use Case	Actor
1.	Select image	Pick Image	User
2.	Send image to be processed	Process Image	Flutter app -> Python Backend
3.	View processed image with data	View Processed Image & Counts	Flutter app
4.	Upload processed image	Store Images	Flutter app -> Python Backend
5.	Upload processed data	Store Data	Flutter app -> Firebase Services
6.	View upload history	View Upload History	User

Table 3.1 shows a straightforward illustration of an image processing system by the foundations of Flutter as a frontend, Python as a backend and Firebase services. The process takes place when the user acts as a primary actor selects images within the application by a task defined in the use case as 'Pick Image'. After done select the image, the Flutter application acts as an intermediary, executes the 'Process Image' task use case, where it sends the selected picture to the Python backend for additional analysis. The backend of the system is responsible for image processing by applying the YOLOv5 model to detect and perform image counting. The processed image with its data will be displayed back to the user in the application. The system's architecture considers the use case of 'Store Images' for the persistence of data which involves uploading the processed image to the Python backend. Additionally, the application interacts with Firebase services to manage the use case of 'Store Data' which archives the information or outcomes of the image processing task, including the number of objects detected found in an image. The

Firestore management allows the user to perform the final use case, "View Upload History", where they can retrieve and view the historical data of processed images and associated results. This seamless interaction between the various components of the system underscores the importance of a well-defined system architecture, ensuring that the application operates efficiently and effectively from image selection to data storage and retrieval.

3.3 Python backend

3.3.1 Process flow for training

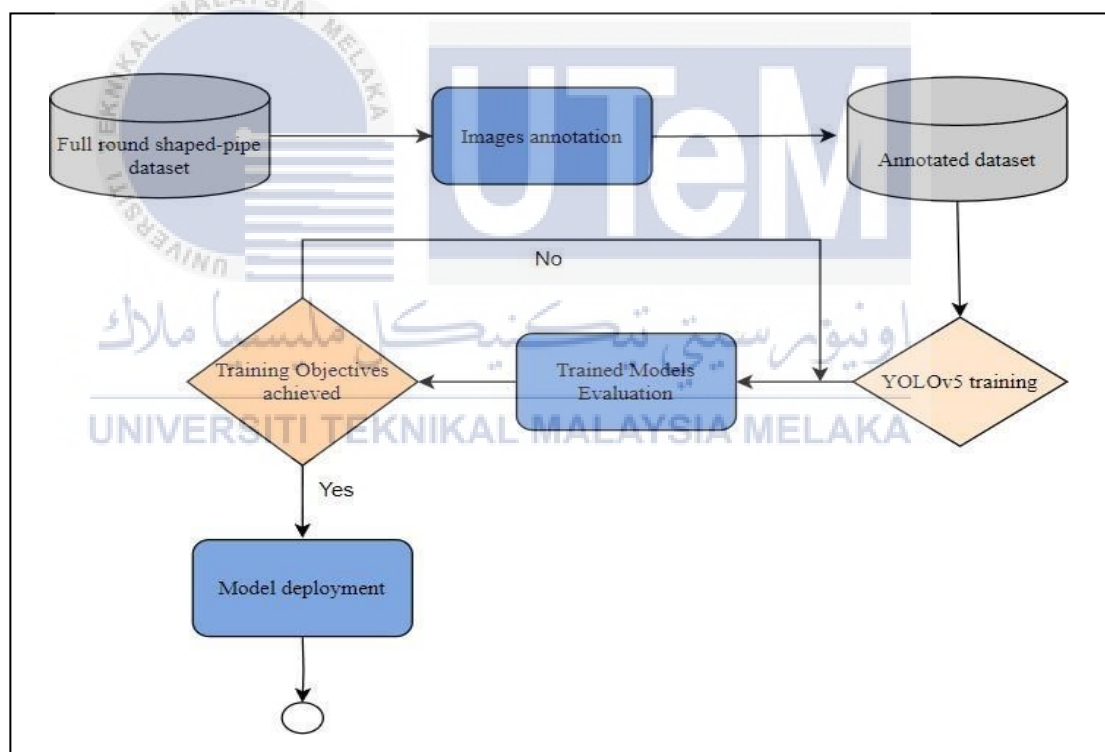


Figure 3.3 Training model process flow

- 1) *Dataset Acquisition*: The dataset is required from round shaped-pipe detection dataset for object detection from Roboflow Universe. The data involves only train

folder. Image dataset contains 250 files which includes the images along with the .txt file.

- 2) *Images Annotation*: The images in data set annotate to give label of objects in image by using bounding box method and annotation tools provided by Roboflow tools.
- 3) *Model Training*: The annotation dataset is being split into two sets which are to train and to validate. To train the object detection model, the annotated images will be used on the training set by using pre-trained model, YOLOv5 based on PyTorch framework.
- 4) *Image Detection*: The images feed the model. The bounding boxes and class labels will be compared to the ground truth annotations and model weights will be updated to reduce the detection error.
- 5) *Model Deployment*: The Python as backend is used to deploy the model on mobile devices such as Android. The model needs to be integrated into a mobile application to process output from the YOLOv5 model.

3.3.2 Process flowchart for testing

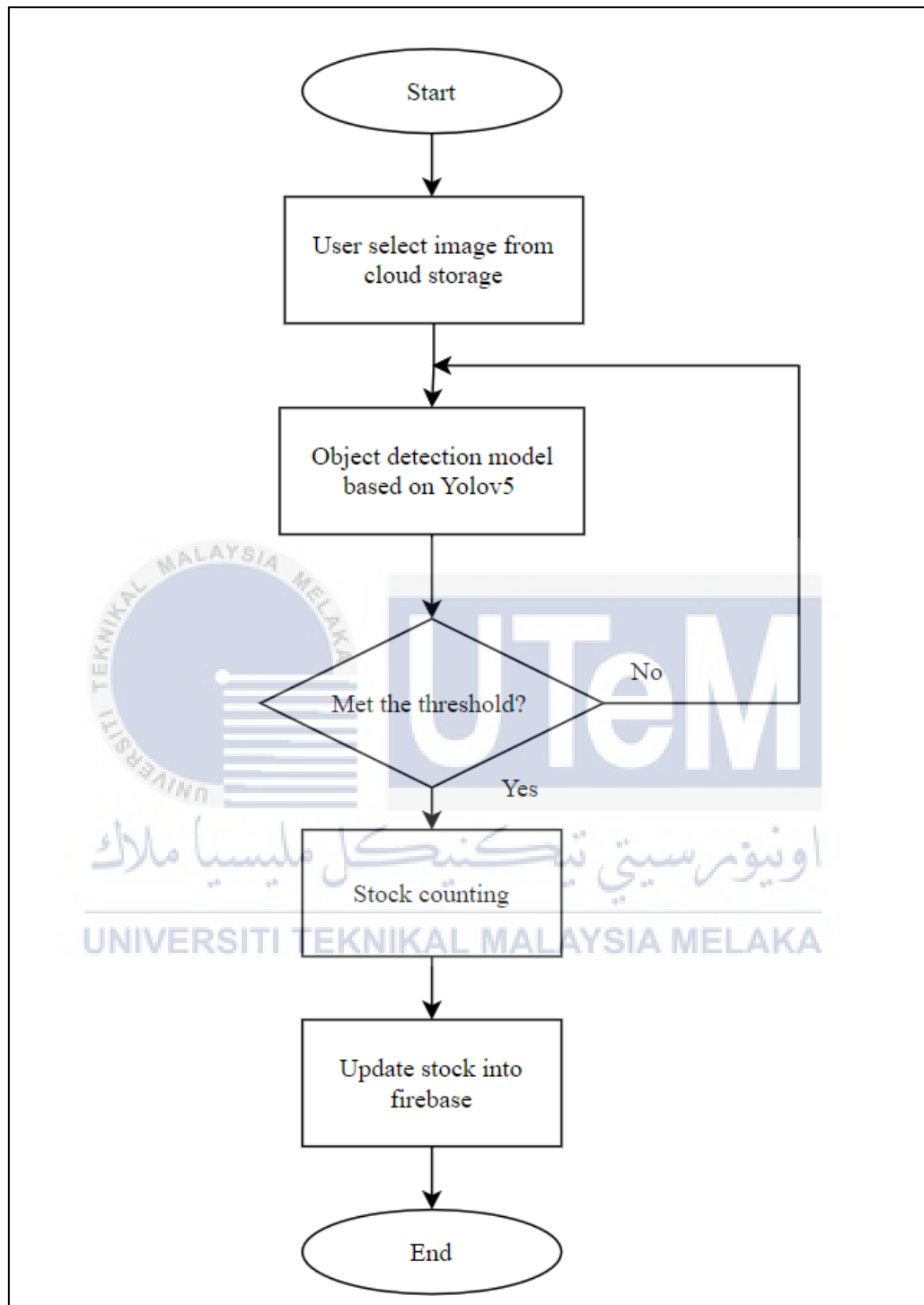


Figure 3.4 Process flowchart for testing

The process flowchart for testing will take place in the Android application. Below is the detailed explanation of each step in the (Figure 3.4):

- 1) *Start*: The process starts to take place.
- 2) *Users select image from cloud storage*: The image of the pipe stock in the warehouse needs to be selected from Android device's gallery.
- 3) *Object detection model based on YOLOv5*: To identify the pipe stock, the round-shaped patterns in the image will be analyzed based on pre-trained model, YOLOv5.
- 4) *Met the threshold?* If the YOLOv5 model tends to meet the threshold, the process goes on. If the YOLOv5 does not meet the requirement, the process will prompt back the user to select image from cloud storage.
- 5) *Stock counting*: The counted pipe stock will be updated by processing image counting.
- 6) *Update stock into firebase*: Firebase fetches the pipe stock and updates the results.
- 7) *End*: The process terminates.

3.3.3 Round-shaped pipe detection

Physical pipes that are kept in a warehouse or inventory environment might be used as test subjects for pipe counting. The pipes will be assigned with the identification number to facilitate accurate identification and do pipe counting during testing process. The pipes stored in racks arrangement and overlap with each other by itself without any other round-shaped object other than pipes. The object detection will detect the subject in its round shape at the back end of the pipes and the image will capture the subject to do object counting through Android device.

3.3.4 Roboflow software for annotating image dataset

The Roboflow software is the advanced software to label and annotate large images datasets. The Roboflow API gives access to a legitimate Python package. Brad Dwyer [18] explains how to use this type of software and its benefits in detail. For the datasets for the object detection model, it is capable of managing, pre-processing, and enhancing, and versioning the datasets. Roboflow capable of importing and exporting picture datasets into any supported formats. The round-shaped pipe dataset will be used for pipe counting. The image set supports object counting and provides the bounding box annotations. A bounding box identifies the objects of different ratios which marks the object's position. These images will be annotated by cropping the bounding boxes and labelling with one class of each image using Roboflow software (Figure 3.5). This software is an advanced platform to label and annotate large image datasets.

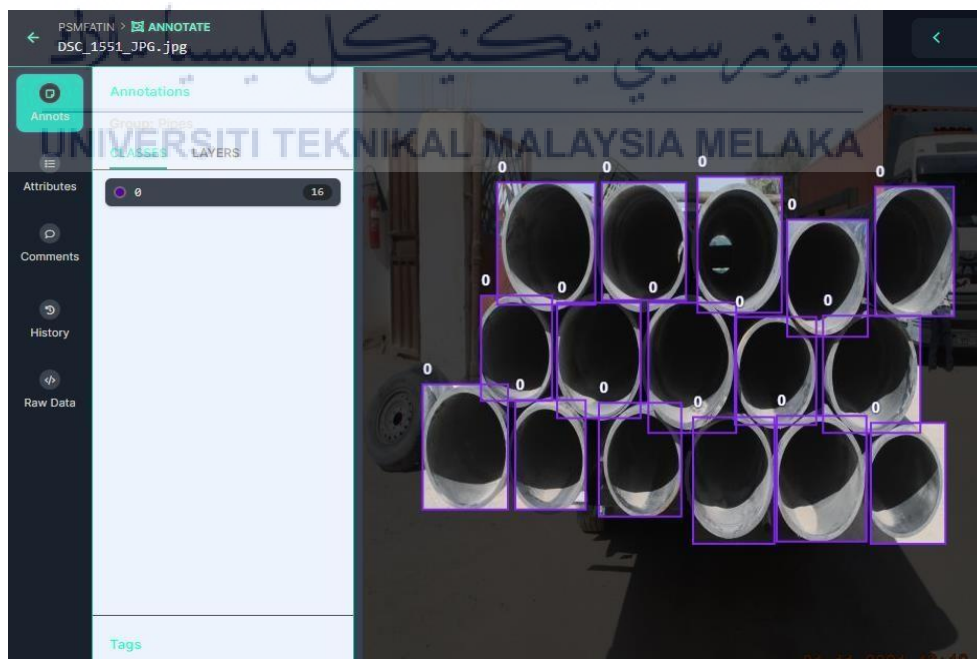


Figure 3.5 Round-shaped pipes annotation for preparing datas

3.3.5 Generate dataset in Roboflow software

The Roboflow software is the suitable platform for image annotation (drawing and image labelling) for object detection. Roboflow was chosen because it is quicker and saves time during annotation and makes it simple to import and export datasets.

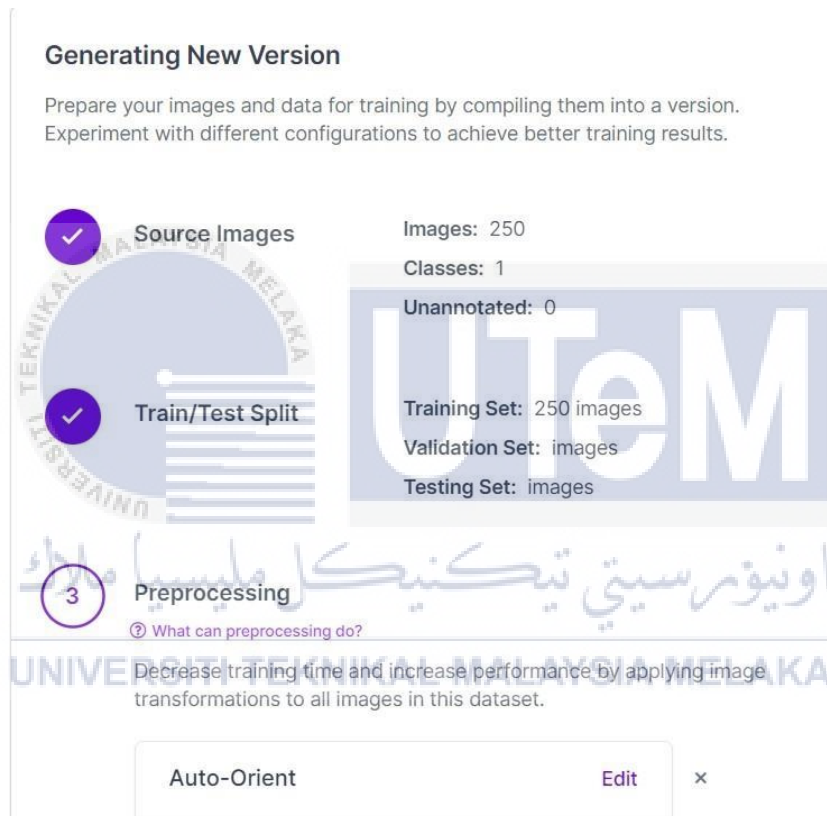
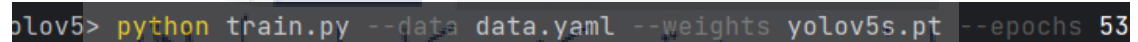


Figure 3.6 Generate dataset pipe counting in Roboflow.

The whole image dataset is divided based on (i) training the dataset, 100%. The testing process is performed through Android devices for better performance pipe detection and counting techniques.

3.3.6 Model Training Procedure using YOLOv5 PyTorch

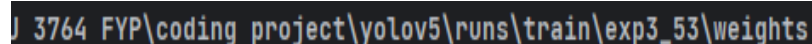
Once the annotated image dataset is ready, the image needs to be trained and tested by using training model of YOLOv5. The version of YOLOv5 source code is using Ultralytics and is available on GitHub. The source code is based on PyTorch framework and requires the installation of the torch and some lightweight Python libraries. For training and validating the YOLOv5 models, the text format set to be used in the code for any input of image which is *.txt. The code uses the YOLOv5 model for .yaml script configuration and to define the model architecture depends on the number of classes, anchors, and layers. Once the image dataset is loaded, classes and config files are defined. Then, the model is trained by passing the following number of arguments. Next is to train the model by using the command (Figure 3.7). The dataset will undergo 53 epochs during training.



```
olov5> python train.py --data data.yaml --weights yolov5s.pt --epochs 53
```

Figure 3.7 Python script model train.py.

Once the training and validation are complete, the trained weights are ready to run inference on any image from the test image dataset. The weights are saved in folder runs/train/exp/weights (Figure 3.8). These weights can be utilized to detect the trained classes in new images via inference.



```
J 3764 FYP\coding project\yolov5\runs\train\exp3_53\weights
```

Figure 3.8 Weights folder

3.4 Python backend

3.4.1 Flask Server

A Python web framework, Flask allows developers to create online apps in a simple way. Flask utilizes the built-in development server that offers to run and test web apps on a local machine. Its features are simple and lightweight, causing this framework is suitable for development and testing purposes. In addition, Flask server offers a sizable and vibrant developer community, which means that a wide range of tools and libraries are accessible to help with development and troubleshooting. Based on this system, the Flask serves as a crucial intermediary between the frontend Flutter application and the backend YOLOv5 PyTorch model for image processing.

3.4.2 Integrating YOLOv5 PyTorch Model with Flutter through Flask

In this setup, the Flask serves as an intermediate crucial between the Flutter application and the YOLOv5 PyTorch model. The images will be sent by the Flutter app to Flask. Then, the images will be processed by using YOLOv5 PyTorch model and return the results back to the Flutter app. The process takes place as below:

a) Load pre-trained model, YOLOv5 in detect.py

Figure 3.9 shows the code snippet of Python script that loads the pre-trained model, YOLOv5 for object detection in mobile applications, The function 'select_device' is for optional hardware processor to execute the model whether it is on CPU or GPU. The variable 'DetectMultiBackend' function responsible to load the model with specific parameters such as

- weights: pre-trained model file
- device: hardware resource to execute the model

```
# Load model
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
```

Figure 3.9 Load the pre-trained model with specified weights and processing the image

b) Select image and send to server in Flutter application (main.dart)

Figure 3.10 shows the code snippet used in Flutter using Dart function to upload image from device's storage. The function '_pickImage()' utilizes the 'ImagePicker' class to allow the user to select an image from the device's gallery. The selected image will be stored into 'image'.

```
void _pickImage() async {
  final ImagePicker _picker = ImagePicker();
  final XFile? image = await _picker.pickImage(source: ImageSource.gallery);
```

Figure 3.10 Body function to allow input

c) Connect with Flask server by HTTP request via Ngrok

Figure 3.11 demonstrates the usage of Ngrok that establishes a safe tunnel to a local server that can be accessed online. The forwarding URL is to connect a public Ngrok URL to a local service that may be operated on specific port.


```
C:\Users\NOR FATIN SYAHIRA x + v
ngrok
Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status      online
Account             fatinira1@gmail.com (Plan: Free)
Update              update available (version 3.5.0, Ctrl-U to update)
Version             3.4.0
Region              Asia Pacific (ap)
Latency             154ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://4d81-42-153-154-227.ngrok-free.app -> http://localhost:5000

Connections
  ttl   opn   rt1   rt5   p50   p90
    0     0    0.00  0.00  0.00  0.00
```

Figure 3.11 Ngrok access

Figure 3.12 shows the Flutter application using Dart HTTP package to perform the HTTP POST request. This allows the utilized public Ngrok URL to send an image file to a Flask server. This is beneficial in mobile apps development to ensure the mobile application able to communicate with backend services.

```
var uri =
  Uri.parse('https://6e85-42-153-154-162.ngrok-free.app/detect_pipes');
var request = http.MultipartRequest('POST', uri)
  ..files.add(await http.MultipartFile.fromPath('file', imageFile.path));
log('Sending');

var response = await request.send();
```

Figure 3.12 Send HTTP post request to Flask server

3.5 Firebase Services

The Flutter framework was created with Firebase authentication. Firebase serves as a crucial component archive the images into certain storage. The same Firebase server will relate to the app to ensure the data access is synchronized. The system has enhanced to further stage in communicate with the user. The name of the project given in Firebase is 'pipe-count' (Figure 3.12).



Figure 3.13 Firebase name project

Figure 3.13 shows the Firebase database interface with a collection named 'images'. The same Firebase server will be connected with the app to ensure the data access is synchronized. The system has enhanced to further stage in communicate with the user.

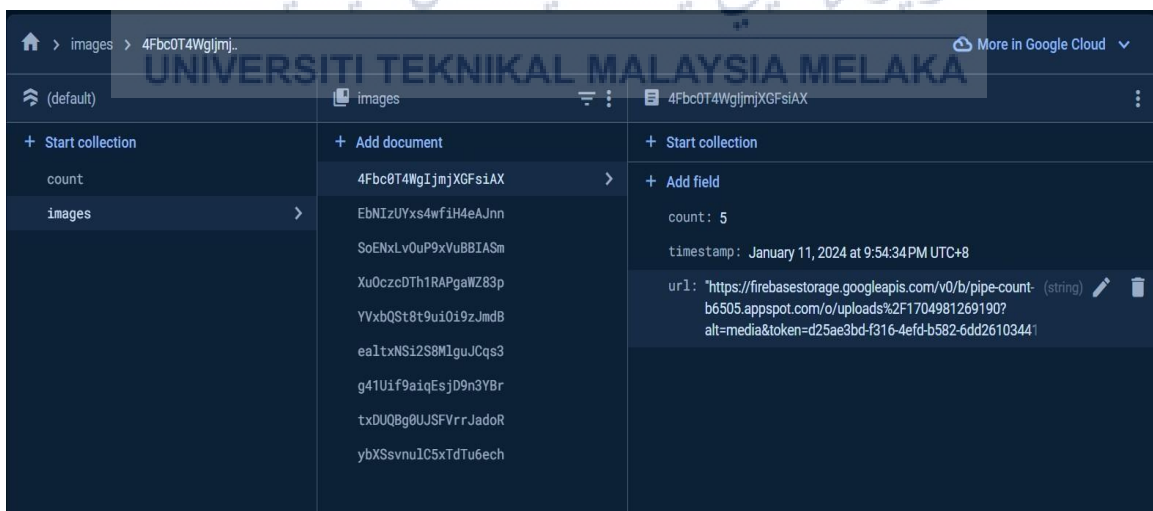


Figure 3.14 Firebase name project

The related fields that store data of image processing task (Figure 3.12) will be explained further below:

- ‘count’: Once the image is uploaded in Flutter, the number of pipes detected in the image is updated.
- ‘timestamp’: The time at which the data image was processed or saved is recorded.
- ‘url’: Indicates the location of the processed picture in Firebase Storage. The picture may be accessed directly with this URL.

3.6 Hardware and Software Development Tools

Development of this Android apps had a mobile application for the warehouse inventory records. Therefore, the development tools involved mobile application development tools:

- ***Laptop***

The pre-trained weights have learned to recognize many useful features from images to detect new objects by images. Training and validation were performed on a laptop with an AMD Ryzen 7 4800H with Radeon Graphics 2.9Ghz CPU connected to NVIDIA GeForce RTX 3050 Ti GPU.

- ***Android Phone***

Android phones are used that is easily accessible to the public since it is known as inexpensive computing devices. The image capture will be done by using Android devices by using Samsung A13 One UI Core Version 5.1 8GB memory. The image is editable to crop the unwanted region area that can interrupt the accuracy of pipe detection.

- ***PyCharm Community Edition 2023.1.2***

Figure 3.14 is the PyCharm IDE used to run the YOLOv5 algorithm and weights for the pretrained dataset from Roboflow library for one class which is pipe detection. The datasets will apply to do the object detection model based on its circular shape size of pipe. This software is also possible to utilize for building applications for mobile apps.



Figure 3.15 PyCharm Software Edition 2023.1.2

- ***Google Firebase cloud***

Figure 3.15 is Google Firebase as cloud storage to allow for synchronization of data across multiple devices and platforms. The Firebase cloud is a suitable platform in providing a real-time database. It is useful for stock counting, as it enables immediate updates and allows for immediate updates to the most recent stock count since it is very useful. Furthermore, the stock counting data is safely stored.



Figure 3.16 Google Firebase cloud

- **Visual Studio Code**

Figure 3.16 is the Visual Studio Code used to write all the source code for the proposed application. It can link to Git, supports several programming languages, and is free. In addition, there are other extensions available for usage that support a variety of programming languages.

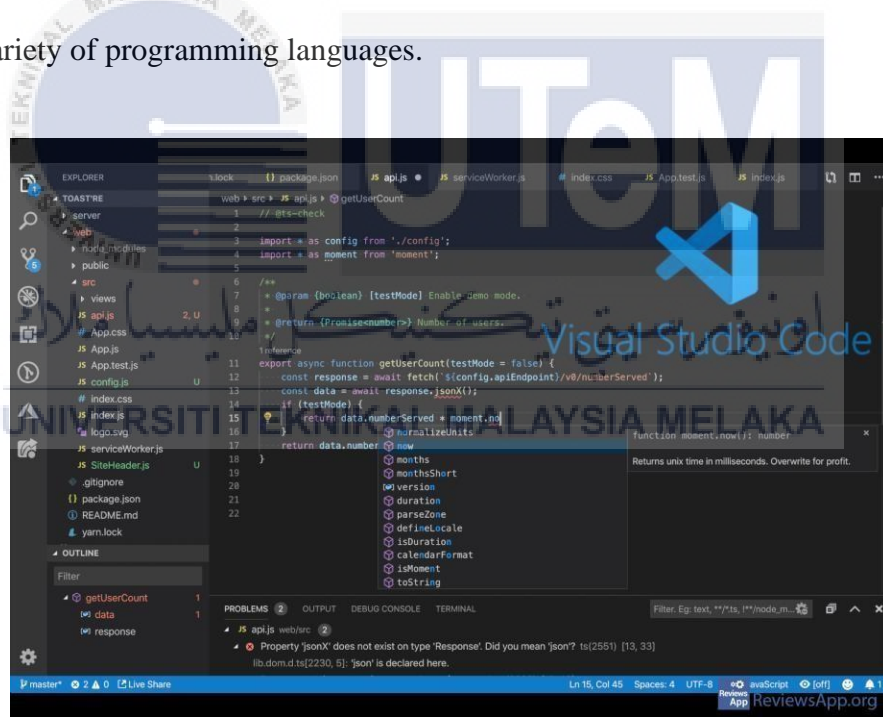


Figure 3.17 Visual Studio Code

3.7 Libraries and framework setup

- ***Roboflow Universe***

Figure 3.17 is the Roboflow framework was utilized to enhance the training set. The images from different sources as well as the own captured images have been collected. All the images of the combined dataset are added by using splitting method to images between train by 100%, valid 0% and test 0%. The prepared dataset link to train YOLOv5 model.



Figure 3.18 Roboflow

- ***Pre-trained model, YOLOv5***

One of the most popular algorithms to date for real-time object detection is YOLO (You Only Look Once), initially proposed by Redmond et.al [19]. This algorithm focuses on object detection in a real-time setting that approaches quick and straightforward because it does not take long to generate region proposals. It is one of the most popular algorithms has emerged in the past few years to tackle the problem. This algorithm focuses on item recognition and speed of detection rather than object location accuracy. Deep Learning is one of the algorithms that has repeatedly demonstrated its importance around object detection. The most recent YOLOv5 implementation created by Ultralytics to carry out an end-to-end object detection project on a custom dataset.

- ***Ngrok***

Developers may use secure tunnels to access their locally hosted servers from the public internet, even when they are protected by firewalls and network address translation (NAT) systems (Figure 3.18). Ngrok is a secure server tunnelling service. Developers may share their web apps, APIs, and webhooks with others for testing and development without having to set up port forwarding or firewall limitations. Ngrok provides both free and premium plans, each with unique features and functionalities. Additional tunnel endpoints, reserved IP addresses, and custom domains are among the advanced capabilities available in the subscription tiers. Developers who need to test their apps in a setting like production before putting them on a live server will find this tool especially helpful.

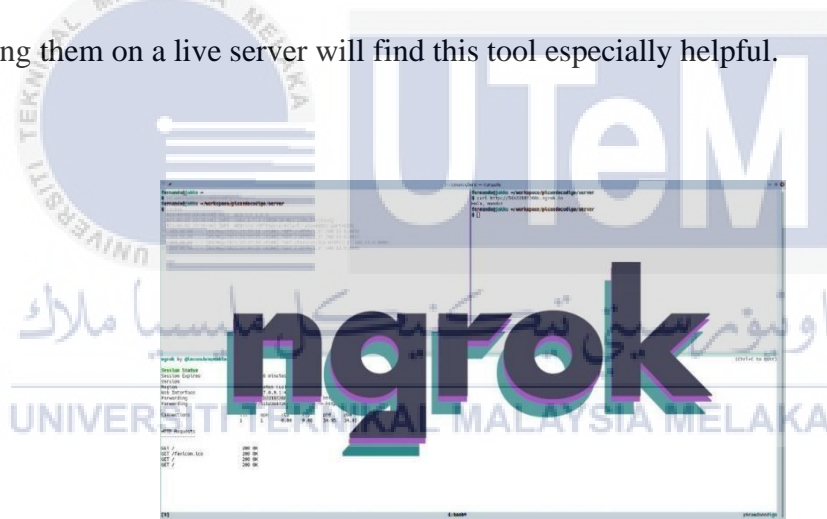


Figure 3.19 Ngrok

- ***Flutter***

In 2017, Google began developing Flutter with Dart as the development programming language. It can be delivered on multiple platforms such as Android and iOS. Over the years, Flutter has gained its popularity since it is relatively new and easy to use by facilitate the application development in Windows, Mac, and Linux.

3.8 Limitation of Proposed Methodology

The creation and usage of pipe counting systems in a warehouse may be constrained in several ways. The system limitation is explained in detail:

- 1) The dataset is only trained based on round-shaped pipe. Uneven round-shaped pipes will reduce the accuracy of object detection. Pipes with non-uniform shapes or deformities require additional techniques to ensure an accurate counting.
- 2) The image capture only contains pipes as a subject. The system may find out the other round-shaped subject as a pipe as well.



3.9 Project Planning

To execute project planning and schedule, a Gantt chart introduced to visualize the progress of a project. The Gantt chart provides a bar chart that helps in scheduling and planning. The essential scheduling deadlines must be followed to ensure that no procedure is missed.

Table 3.2 Gantt Chart Process for Final Year Project I

No.	Project activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Introduction & Project Plan	█													
2	Literature Review		█	█	█		█				█	█	█		
3	Methodology				█	█	█	█	█	█					
4	Preliminary test/investigation results													█	
5	Report, writing and presentation														█

Table 3.3 Gantt Chart Process for Final Year Project II

No.	Project activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Develop System Algorithm to implement deep learning technique and Flutter														
2	Research methodology														
3	Result and Discussion														
4	Poster preparation														
5	Report, writing and presentation														

3.10 Summary

The proposed methodology for pipes counting system in a warehouse typically involves a combination of technology and processes to accurately count and track the pipes. To precisely count and track the pipes, the proposed technique for a pipe counting system in a warehouse often combines technology and processes. A pipe counting system that aims to this methodology attempts to offer an automated and trustworthy solution for precisely tracking and counting pipes in a warehouse setting, enhancing inventory management and increase in operational effectiveness.



CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

Accurate detection and counting of pipes are crucial tasks in various industries, including manufacturing and construction. With the development of computer vision and deep learning, the ability to automate pipes has been demonstrated to be highly effective. This chapter presents results and analysis of the pipe detection and counting pipes by using the pre-trained model, YOLOv5 based on PyTorch framework. The application named 'Pipe Quant' aims to streamline the process of counting and tracking pipes in a warehouse, improve overall efficiency and reduce human error in terms of manual counting. The development process involved the pre-trained YOLOv5 model and the PyTorch framework for training performance. For testing data model and framework performance used Python as backend for processing output from the YOLOv5 model to integrate into Flutter.

4.2 System Operation and Testing

4.2.1 User Interface Design

Figure 4.1 shows the home screen of a mobile device with an application named 'Pipe Quant'. The 'Pipe Quant' is developed and deployed into edge device by compiled an Android Package Kit (APK) file. This mobile application focuses on performing inventory or stock management through image processing and counting.



Figure 4.1 Mobile application named 'Pipe Quant' on edge device

Figure 4.2 shows the interface for detecting and counting pipes. The app is currently displaying the status message, "No processed image available". There were no pipes found. This implies that the app contains a feature for processing photos and detecting the existence of pipes inside them, but no images have been processed, or no pipes have been discovered in the most recently processed image. To upload, press the camera icon button at the bottom of the screen. This confirms that users may take pictures directly from the app's UI. The pictures will be transferred to a cloud service for further processing.



Figure 4.2 Pipes detect and count

Figure 4.3 shows the upload history from the 'Pipe Quant' application.

The elements involved on this screen are:

- Upload History: This suggests that the screen lists a history of images that have been uploaded to the application for processing.
- Search Bar: At the top, there is a search bar, which likely allows users to filter or search through the upload history based on specific criteria or keywords.
- List of Uploaded Images: The pictures uploaded will be displayed based on the timestamps, pipe counting results and thumbnails of the uploaded photos with bounding box of image detection.



Figure 4.3 Historical Data Review

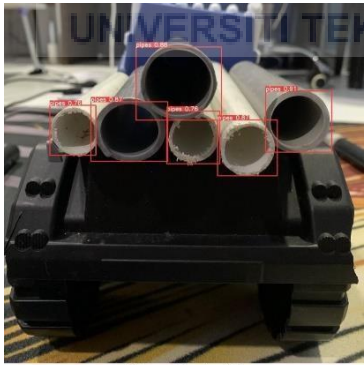
4.3 System performance


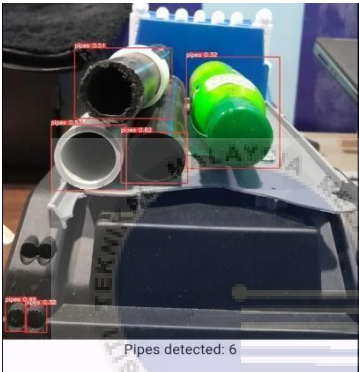

The developed pipe-counting application is tested on three distinct sets to perform object detection accuracy for pipe counting (Table 4.1). The table uses the formula as below:

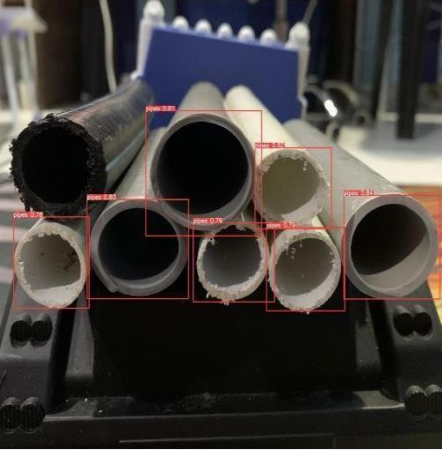

$$\% \text{ error} = \frac{\text{Theoretical} - \text{Experimental}}{\text{Theoretical}} \times 100 \%$$



The percentage error is calculated by comparing the experimental value (app's count) to theoretical value (manual count). The output of pipe detection for different sets are visualized in Table 4.1

Table 4.1 System performance testing in pipe counting application

Set	Testing Image	Theoretical (manual count)	Experimental (app's count)	% error
1	 <p>Pipes detected: 6</p>	6	6	0% fail, 100% success

2	 <p>Pipes detected: 4</p>	4	4	0% fail, 100% success
3	 <p>Pipes detected: 6</p>	3	6	50% fail, 50% success
4	 <p>Pipes detected: 6</p>	6	6	0% fail, 100% success

5	 <p>Pipes detected: 7</p>	8	7	12.5% fail, 87.5% success
6	 <p>Pipes detected: 5</p>	5	5	0% fail, 100% success

7	 <p>Pipes detected: 3</p>	3	3	0% fail, 100% success
8	 <p>Pipes detected: 10</p>	10	10	0% fail, 100% success

<p>9</p>  <p>Pipes detected: 11</p>	<p>9</p>	<p>11</p>	<p>22% fail, 78% success</p>
<p>10</p>  <p>Pipes detected: 9</p>	<p>8</p>	<p>9</p>	<p>12.5% fail, 87.5% success</p>
<p>Mean average accuracy</p>			<p>90.3%</p>

4.4 Summary

Table 4.1 analyses the accuracy of YOLOv5 model to perform object detection accuracy for pipe counting. The test is conducted to see how the application is functioning by testing it with 10 samples. The application is tested to see how the application can detect pipes and count in a static image. According to analysis, the first and second tests yielded a full success rate, and no observed error was found to identify all pipes.

After all the images tested with the application, the accuracy is calculated to see on how the test and results are based on the application. The highest accuracy recorded for the pipes detection and count is 100% whereby all pipes successfully detected in an image on its bounding box region. However, several set images tested do not fulfill the highest accuracy due to the limitations of the training dataset. The lowest accuracy is 50% success whereby all the pipes cannot be detected by the application. On the other hand, several set images contain the pipes detection on the circular shape that is not pipe might be because of the brightness of the image, the quality of the image or the position of the mobile phone to capture the images. Despite this issue, the mean average accuracy obtained across all 10 tests was approximately 90.3%, reflecting the model's potential despite the need for adjustments to improve its generalization capabilities.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The development of pipes counting application for warehouse inventory based on Android platform gives numerous advantages and an effective solution to control for managing inventory in warehouses. This application enables users to precisely count and monitor the quantity of pipes in stock, which optimizes inventory control and streamlines operations. The warehouse staff can accurately assess the pipe quantities in real time updates to the database system by running on their devices including smartphones and tablets. While manual counting for stock checking and stock update are a common technique, it comes with several limitations, which involve human errors and data inaccuracy. The utilization of the pre-trained model, YOLOv5 based on PyTorch framework and integrated with a Python backend can detect and count pipes in an image quickly and accurately to manage the stock of pipes efficiently. With a proven high accuracy rate of 83.33% across various sets of testing showcases it is well-suited for inventory management applications. Python as backend for processing output from the YOLOv5 model to integrate into Flutter framework provides a flexible algorithm for developing the application with a comprehensive set of tools and libraries. It also enables efficient deployment on the Android platform.

5.2 Potential for commercialization

The innovation of pipes counting application has demonstrated significant economic potential due to its unique applications in inventory management, quality control and data collection and analytics. The application of this technology in a commercial setting provides an extensive response to several operational issues that are commonly implied in product handling and warehouse management.

The project offers a significant advance over manual approaches in the inventory management space due to its capacity to precisely track and monitor stock levels in real time. This automated method lowers the likelihood of human mistakes while also increasing productivity, which might result in optimized stock levels and accurate inventory evaluations.

Another crucial area where the initiative is expected to provide value is quality control. Businesses may drastically lower the frequency of flaws and non-compliance by automating the inspection process and making sure that goods fulfill defined standards of quality. This protects the reputation of the business while minimizing waste and raising consumer satisfaction.

Moreover, it is difficult to overstate the project's contribution to data collecting and analytics. Businesses may obtain meaningful insights that help guide strategy and decision-making by collecting and analyzing data on inventory and quality control. This analytical component enables a more in-depth understanding of operational dynamics, resulting in more informed decision-making, better forecasting, and enhanced overall business intelligence.

5.3 Future Works

Several suggestions and modifications can be extended for future development in its usefulness and potential in economics. The focus of extension will target two primary advancements. First, the system is focused on optimizing the object detection algorithm to better differentiate between pipes and other circular projects that are not pipes. This can be done by image segmentation. This image processing technique is carried out by dividing an image into segments to distinguish between pipes and other circular objects by analyzing its features based on its shape, size, and each segment. By integrating this method with YOLOv5 framework, the pipe counting application can compute between pipes and non-pipe circular objects. Thus, the inventory counts will be more accurate and improved the efficiency of warehouse operational settings. Additionally, the enhancement on the system can be done by establishing a reliable connection to the host server, which allow real-time data synchronization and system upgrades. At the same time, this ensures that the warehouse operations are securely managed.

REFERENCES

- [1] D. C. Mumba, "School of Business and Information Technology a Comparison of Automated Warehouse With Traditional Warehouse Management: a Case Study of Dmmu in Zambia," 2020.
- [2] W. Computer *et al.*, "How Does Warehouse Computer Vision Work ?".
- [3] W. Safety and H. Topics, "A packer in a manufacturing facility dies when he is struck by a pipe that rolled off a storage rack . Cause of Death Recommendations / Discussion".
- [4] I. H. Q. D. Mohd Mahzan and K. L. Lee, "Elimination of Misconduct in Manual Counting Process as an Improvement of Inventory Accuracy in A Manufacturing Company," *Int. J. Ind. Manag.*, vol. 10, no. 1, pp. 140–150, 2021, doi: 10.15282/ijim.10.1.2021.6051.
- [5] R. Haerani and P. Desianasari, "the Design of a Stock Taking Inventory Application Based on Android," *JURTEKSI (Jurnal Teknol. dan Sist. Informasi)*, vol. 8, no. 3, pp. 313–320, 2022, doi: 10.33330/jurteksi.v8i3.1529.
- [6] D. A. (University O. C. A. B. Forsyth and J. (University O. I. Ponce, "Computer Vision A Modern Approach (no header footers)," *Education*.
- [7] F. Jalled and I. Voronkov, "Object Detection using Image Processing," pp. 1–6, 2016, [Online]. Available: <http://arxiv.org/abs/1611.07791>
- [8] R. Lumauag and M. Nava, "Fish tracking and counting using image processing," *2018 IEEE 10th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2018*, pp. 1–4, 2019, doi:

10.1109/HNICEM.2018.8666369.

- [9] J. Ni, Z. Khan, S. Wang, K. Wang, and S. K. Haider, "Automatic detection and counting of circular shaped overlapped objects using circular hough transform and contour detection," *Proc. World Congr. Intell. Control Autom.*, vol. 2016-Sept, no. Kyx15 0496, pp. 2902–2906, 2016, doi: 10.1109/WCICA.2016.7578268.
- [10] K. Roy, R. Dey, D. Bhattacharjee, M. Nasipuri, and P. Ghosh, "A smart phone based app for automated segmentation and counting of platelets," *2016 3rd Int. Conf. Recent Adv. Inf. Technol. RAIT 2016*, pp. 434–438, 2016, doi: 10.1109/RAIT.2016.7507941.
- [11] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," *J. Phys. Conf. Ser.*, vol. 1544, no. 1, 2020, doi: 10.1088/1742-6596/1544/1/012033.
- [12] S. Kumar, Vishal, P. Sharma, and N. Pal, "Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow," *Proc. - Int. Conf. Artif. Intell. Smart Syst. ICAIS 2021*, pp. 1017–1022, 2021, doi: 10.1109/ICAIS50930.2021.9395971.
- [13] I. S. Gillani *et al.*, "Yolov5, Yolo-x, Yolo-r, Yolov7 Performance Comparison: A Survey," no. Figure 1, pp. 17–28, 2022, doi: 10.5121/csit.2022.121602.
- [14] W. S. Wu and Z. M. Lu, "A Real-Time Cup-Detection Method Based on YOLOv3 for Inventory Management," *Sensors*, vol. 22, no. 18, 2022, doi: 10.3390/s22186956.
- [15] I. F. E. Babila, S. A. E. Villazor, and J. C. Dela Cruz, "Object Detection for Inventory Stock Counting Using YOLOv5," *2022 IEEE 18th Int. Colloq. Signal Process. Appl. CSPA 2022 - Proceeding*, no. May, pp. 304–309, 2022, doi: 10.1109/CSPA55076.2022.9782028.
- [16] M. Sozzi, S. Cantalamessa, A. Cogato, A. Kayad, and F. Marinello, "Automatic

- Bunch Detection in White Grape Varieties Using YOLOv3, YOLOv4, and YOLOv5 Deep Learning Algorithms,” *Agronomy*, vol. 12, no. 2, 2022, doi: 10.3390/agronomy12020319.
- [17] C. Pornpanomchai, F. Stheitstienchai, and S. Rattanachuen, “Object detection and counting system,” *Proc. - 1st Int. Congr. Image Signal Process. CISP 2008*, vol. 2, pp. 61–65, 2008, doi: 10.1109/CISP.2008.108.
- [18] J. Quispe, “No Titleการบริหารจัดการการบริการที่มคอภาพใน
โรงพยาบาลสงฆ์ กระทรวงสาธารณสุข,” *วารสารวารสารวิชาการมหาวิทยาลัยอีสเทิร์น*
กต วารสารราช เอเชีย, vol. 4,
no. 1, pp. 88–100, 2023.
- [19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.
- 